

## Tema 1. Estándares internacionales para la actividad empresarial y comercial.

1.1. Sistema GS1 (Global Standard One).

1.2. Codificación de productos y mercancías GTIN (Global Trade Item Number).

1.3. Identificación de objetos mediante radiofrecuencia (RFID)  
y tecnologías de Internet.

**1.4. Esquemas XML (uso en procesos de negocio).**



1.5. EDI (intercambio electrónico de datos).

## XML ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

**XML** (*eXtensible Markup Language* - lenguaje de marcas extensible) es un lenguaje de marcas desarrollado por el [World Wide Web Consortium](#) (W3C). Deriva del lenguaje [SGML](#) y **permite definir la gramática de lenguajes específicos para estructurar documentos grandes**. A diferencia de otros lenguajes, XML da soporte a bases de datos y es útil cuando varias aplicaciones se deben comunicar entre sí o integrar información.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)

## XML ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

### Historia (1)

XML proviene de un lenguaje inventado por IBM en los años setenta, llamado [GML](#) (*Generalized Markup Language*), que surgió por la necesidad que tenía la empresa de almacenar grandes cantidades de información. Este lenguaje gustó a la [ISO](#), por lo que en 1986 trabajaron para normalizarlo, creando [SGML](#) (*Standard Generalized Markup Language*), capaz de adaptarse a un gran abanico de problemas. A partir de él se han creado otros sistemas para almacenar información.

En el año 1989 Tim Berners Lee creó la web, y junto con ella el lenguaje HTML. Este lenguaje se definió en el marco de SGML y fue de lejos la aplicación más conocida de este estándar. Los navegadores web sin embargo siempre han puesto pocas exigencias al código HTML que interpretan y así las páginas web son caóticas y no cumplen con la sintaxis. Estas páginas web dependen fuertemente de una forma específica de tratar los errores y las ambigüedades, lo que hace a las páginas más frágiles y a los navegadores más complejos.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)

## XML ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

### Historia (2)

Otra limitación del HTML es que cada documento pertenece a un vocabulario fijo, establecido por la [DTD](#). No se pueden combinar elementos de diferentes vocabularios. Asimismo es imposible para un intérprete (por ejemplo un navegador) analizar el documento sin tener conocimiento de su gramática (del DTD). Por ejemplo, el navegador sabe que antes de una etiqueta <div> debe haberse cerrado cualquier <p> previamente abierto. Los navegadores resolvieron esto incluyendo lógica ad hoc para el HTML, en vez de incluir un analizador genérico. Ambas opciones, de todos modos, son muy complejas para los navegadores.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)

## XML ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

### Historia (3)

Se buscó entonces definir un subconjunto del SGML que permitiera:

- Mezclar elementos de diferentes lenguajes, es decir, que los lenguajes fuesen extensibles.
- La creación de analizadores simples, sin ninguna lógica especial para cada lenguaje.
- Empezar de cero y hacer hincapié en que no se acepte nunca un documento con errores de sintaxis.

Para hacer ésto XML deja de lado muchas características de SGML que estaban pensadas para facilitar la escritura manual de documentos. XML en cambio está orientado a hacer las cosas más sencillas para los programas automáticos que necesiten interpretar el documento.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)

**XML** (<http://www.ibm.com/developerworks/ssa/local/webservices/wa-xml-related-intro/section5.html> )

## Historia (4)

Así, en 1.998, surge el metalenguaje XML.



Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)

## XML ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

### Críticas

XML y sus extensiones han sido regularmente criticadas por su nivel de detalle y complejidad. El mapeo del modelo de árbol básico de XML hacia los sistemas de tipos de los lenguajes de programación o bases de datos puede ser difícil, especialmente cuando se utiliza XML para el intercambio de datos altamente estructurados entre aplicaciones, lo que no era su objetivo primario de diseño. Otras críticas intentan refutar la afirmación de que XML es un lenguaje autodescriptivo (aunque la especificación XML no hace ninguna afirmación de este tipo). Se propone a [JSON](#) y [YAML](#) frecuentemente como alternativas, centrándose ambas en la representación de datos estructurados, en lugar de documentos narrativos.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)

## XML ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

### Ventajas del XML

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML añadiendo nuevas etiquetas, y se puede continuar utilizando sin complicación alguna.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan *bugs* y se acelera el desarrollo de aplicaciones.
- Si un tercero decide usar un documento creado en XML, es fácil entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos Postgres y comunicarla con otra aplicación en Windows y Base de Datos MS-SQL Server.
- Transformamos datos en información, pues se le añade un significado concreto y los asociamos a un contexto, lo que aporta flexibilidad para estructurar documentos.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)



## XML ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

### Estructura de un documento XML (1)

*La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible.* Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de trozos de información. Por ejemplo: un tema musical, que se compone de compases, que están formados a su vez por notas. Estas partes se llaman **elementos**, y se les señala mediante **etiquetas**.

Una **etiqueta** es una marca hecha en un documento para señalar una porción de éste como un elemento, y un **elemento** es un trozo de información que tiene un sentido claro y definido. Formato:

- etiquetas: **<nombre>**, donde **nombre** es el nombre del elemento que se está señalando
- elementos: **<nombre>trozo de información</nombre>**

Puede haber **etiquetas vacías** que señalan **elementos sin contenido**. En este caso sólo se pone la marca fin de elemento (**<nombre/>**).

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)

**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))  
<http://www.desarrolloweb.com/articulos/prologo-xml.html>)

## Estructura de un documento XML (2)

**Elementos** (1).- Los elementos XML pueden tener contenido (más elementos, caracteres, o ambos a la vez) o bien ser elementos vacíos. Elementos con contenido son por ejemplo:

```
<nombre>Fulano Mengáñez</nombre>  
<aviso tipo="emergencia" gravedad="mortal">Que no cunda el pánico</aviso>
```

Siempre empiezan con una **<etiqueta>**, que puede contener atributos o no, y terminan con una **</etiqueta>** que debe tener el mismo nombre. Es obligado "cerrar" todos los elementos.

El símbolo < siempre se interpreta como inicio de una etiqueta XML. Si no es así, el documento no estará bien-formado. Para usar ciertos símbolos se usan las entidades predefinidas, que se explican más adelante.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/prologo-xml.html>

**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))  
(<http://www.desarrolloweb.com/articulos/prologo-xml.html>)

## Estructura de un documento XML (3)

**Elementos (2).**- Un elemento vacío es aquel que no tiene contenido (claro).  
Por ejemplo:

```
<identificador DNI="23123244"/>  
<linea-horizontal/>
```

Al no tener una etiqueta de "cierre" que delimite un contenido, se utiliza la forma **<etiqueta/>**, que puede contener atributos o no.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/prologo-xml.html>

**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))  
(<http://www.desarrolloweb.com/articulos/prologo-xml.html>)

## Estructura de un documento XML (4)

**Atributos** (1) .- Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento.

Los atributos se incluyen en la etiqueta del elemento, no en su texto, y sus valores deben ir entre comillas.

Ejemplo:

Un elemento **chiste** puede tener un atributo **tipo** y un atributo **calidad**, con valores **vascos** y **bueno** respectivamente.

```
<chiste tipo="vascos" calidad="bueno">Esto es un día que Patxi y Josu  
van paseando... </chiste>
```

En la Definición de Tipo de Documento se especifican los atributos que puede tener cada tipo de elemento, así como sus valores y tipos de valor posible.

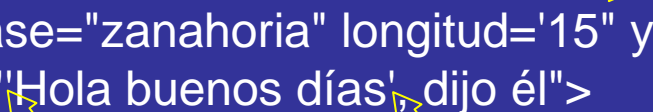
Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/prologo-xml.html>

**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))  
(<http://www.desarrolloweb.com/articulos/prologo-xml.html>)

## Estructura de un documento XML (5)

**Atributos** (2).- Al igual que en otras cadenas literales de XML, los atributos pueden estar marcados entre comillas verticales ( ' ) o dobles ( " ). Cuando se usa una para delimitar el valor del atributo, el otro tipo de comillas se puede usar dentro.

```
<verdura clase="zanahoria" longitud='15" y media'>  
<cita texto="\"Hola buenos días\" dijo él">
```



A veces, un elemento con contenido, puede modelarse como un elemento vacío con atributos. El mismo concepto se puede representar de muy diversas formas, pero una vez elegida una, es aconsejable fijarla en la DTD, y usar siempre la misma consistentemente dentro de un documento XML.

```
<gato><nombre>Micifú</nombre><raza>Persa</raza></gato>  
<gato raza="Persa">Micifú</gato>  
<gato raza="Persa" nombre="Micifú"/>
```

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/prologo-xml.html>

## XML ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

### Estructura de un documento XML (6). Ejemplo de estructura de un documento XML

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

```
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">
```



```
<Edit_Mensaje>
```

```
  <Mensaje>
```

```
    <Remitente>
```

```
      <Nombre>Buenaventura Clares</Nombre>
```

```
      <Mail>bclares@ugr.es</Mail>
```

```
    </Remitente>
```

```
    <Destinatario>
```

```
      <Nombre>Sistemas de Información para Empresas</Nombre>
```

```
      <Mail>sie@correo.etsiit</Mail>
```

```
    </Destinatario>
```

```
    <Asunto>Ejemplo de estructura de un documento XML</Asunto>
```

```
    <Texto>
```

```
      <Parrafo>
```

Este es un documento con una estructura muy sencilla. No contiene atributos ni entidades ...

```
      </Parrafo>
```

```
    </Texto>
```

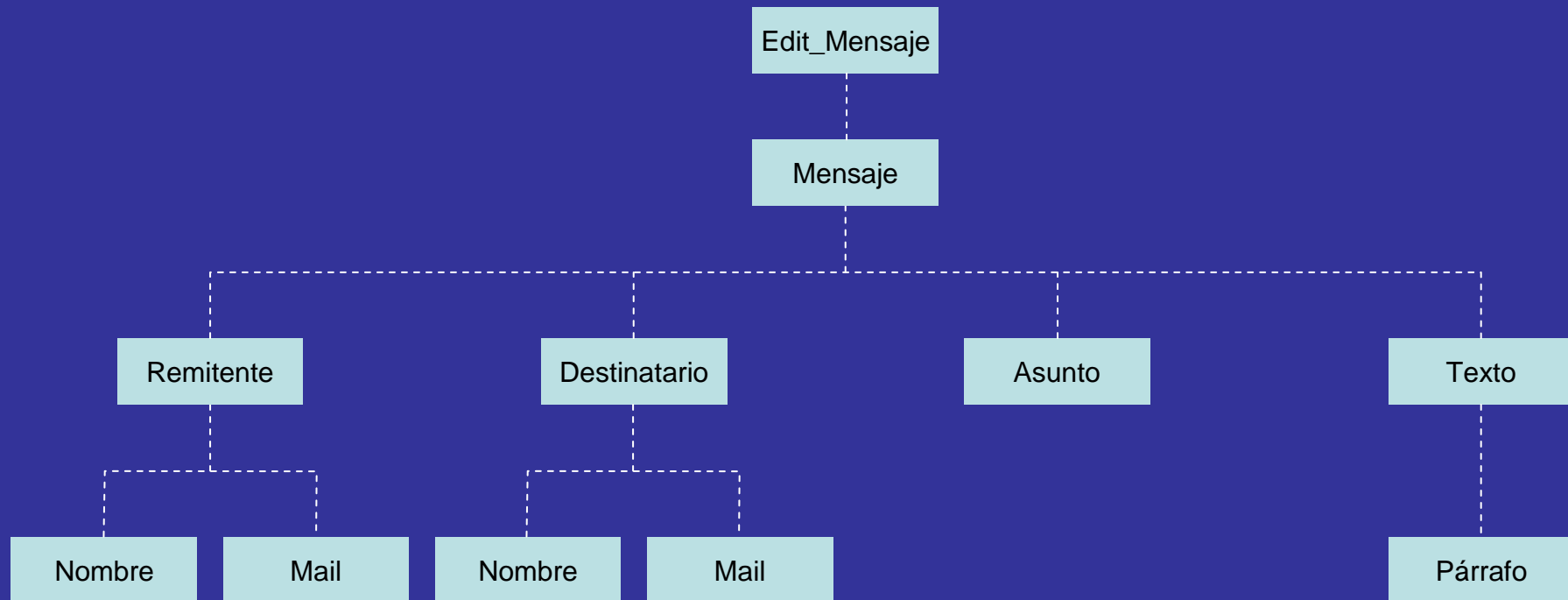
```
  </Mensaje>
```

```
</Edit_Mensaje>
```

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)

**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

**Estructura de un documento XML (7).** Ejemplo de estructura de un documento XML



Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)

**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

## Estructura de un documento XML (8).



### Ejemplo de código de la DTD del documento «Edit\_Mensaje.dtd»

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<!-- Este es el DTD de Edit_Mensaje -->
```

```
<!ELEMENT Mensaje (Remitente, Destinatario, Asunto, Texto)*>
```

```
<!ELEMENT Remitente (Nombre, Mail)>
```

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

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

```
<!ELEMENT Destinatario (Nombre, Mail)>
```

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

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

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

```
<!ELEMENT Texto (Parrafo)>
```

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

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)



**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/documento-xml-bien-formado.html>)

## Documentos XML bien formados y control de errores (1)

Los documentos denominados como «bien formados» son aquellos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, analizarse correctamente por cualquier analizador sintáctico (*parser*) que cumpla con la norma. Esto es independiente del concepto de validez que se explica más adelante.

Para ello:

1. Los documentos han de seguir una estructura estrictamente jerárquica respecto a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar bien anidadas. Los elementos con contenido deben estar correctamente cerrados (`</nombre>`).
2. Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, solo pueden tener un elemento inicial.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/documento-xml-bien-formado.html>

**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/documento-xml-bien-formado.html>)

## Documentos XML bien formados y control de errores (2)

3. Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles (Ejemplo: <Universidad de Granada="http://www.ugr.es"> ).
4. XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados "**espacios en blanco**" o separadores (espacios, tabuladores, retornos de carro, saltos de línea) que hacen más legible el código y que los procesadores XML tratan de forma diferente en el marcado XML. Generalmente los ignoran, aunque, en otros casos, sin embargo, resultan muy significativos para, por ejemplo, separar palabras en un texto, o separar líneas de párrafos diferentes .

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/documento-xml-bien-formado.html>

**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/documento-xml-bien-formado.html>)

## Documentos XML bien formados y control de errores (3)

5. Es necesario asignar nombre a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen algunas características en común. Según la especificación XML 1.0:

- Un nombre empieza por una letra o uno o más signos de puntuación, y continúa con letras, dígitos, guiones, rayas, dos puntos o puntos, denominados de forma global como **caracteres de nombre**. Las letras y rayas se pueden usar en cualquier parte del nombre. También se pueden incluir dígitos, guiones y caracteres de punto, pero no se puede empezar por ninguno de ellos. El resto de caracteres, como algunos símbolos y espacios en blanco, no se pueden usar.
- No se pueden crear nombres que empiecen con la cadena "xml", "xMI", "XML" o cualquier otra variante. Estos nombres se reservan para la estandarización de ésta o de futuras versiones de esta especificación.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/documento-xml-bien-formado.html>

**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/documento-xml-bien-formado.html>)

## Documentos XML bien formados y control de errores (4)

6. Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan **marcas** y son las partes del documento que el procesador XML espera entender:

una marca empieza por < y acaba con >, o bien, en el caso de las referencias de entidad, empieza por & y acaba con ; .

El resto del documento que se encuentra entre marcas es la información “entendible” para las personas.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/documento-xml-bien-formado.html>

## XML ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

### Validez

Que un documento esté «bien formado» se refiere sólo a su estructura sintáctica básica es decir, que se componga de elementos, atributos y comentarios como XML especifica que se escriban. Ahora bien, **cada aplicación de XML**, es decir, cada lenguaje definido con esta tecnología, **necesitará especificar cuál es exactamente la relación que debe verificarse entre los distintos elementos presentes en el documento.**

Esta relación entre elementos se especifica en un documento externo o definición expresada como:

- DTD (***Document Type Definition***, 'Definición de Tipo de Documento'), o como
- XSchema.

**Crear una definición equivale a crear un nuevo lenguaje de marcado, para una aplicación específica.**

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)

## XML ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes))

### Partes de un documento XML (1)

Un documento XML está formado por:

**Prólogo.-** Aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión de XML, el tipo de documento, y otras cosas.

**Cuerpo.-** Es obligatorio. El cuerpo debe contener sólo un elemento raíz, característica indispensable también para que el documento esté bien formado. Es necesaria la adquisición de datos para su buen funcionamiento. Ejemplo:

```
<Edit_Mensaje>
```

```
(...)
```

```
</Edit_Mensaje>
```

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)

**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/prologo-xml.html>)

## Partes de un documento XML (2)

### Prólogo (1)

El prólogo de un documento XML contiene:

1. Una **declaración XML**. Es la sentencia que lo declara como un documento XML.
2. Una **declaración de tipo de documento**. Enlaza el documento con su [DTD](#) (**definición de tipo de documento**). La DTD también puede estar incluida en la propia declaración o ambas cosas al mismo tiempo.
3. Uno o más **comentarios e instrucciones de procesamiento**.

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible\\_Markup\\_Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/prologo-xml.html>

**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/prologo-xml.html>)

## Partes de un documento XML (3)

### Prólogo (2)

La **declaración XML** define la **versión de XML usada** (hasta ahora sólo hay una, la 1.0) Además se especifica **la codificación del documento**, que puede ser, por ejemplo, US-ASCII (7 bits), UTF-8 (código Unicode del que el ASCII es un subconjunto), UCS-2, EUC-JP, Shift\_JIS, Big5, ISO-8859-1 hasta ISO- 8859-7. En general, y para uso con **lenguajes europeos** (incluyendo el juego de caracteres especiales del castellano, **usamos UTF-7 o ISO-8859-1**).

También se puede incluir una declaración de documento autónomo (standalone), que controla qué componentes de la DTD son necesarios para completar el procesamiento del documento

### EJEMPLO:

```
<?xml version="1.0" encoding="UTF-7"?>
```

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/prologo-xml.html>



**XML** ([http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/prologo-xml.html>)

## Partes de un documento XML (4)

### Prólogo (3)

La **declaración de tipo de documento** especifica el tipo del documento que estamos creando para que pueda ser procesado correctamente, ésto es, declara la Definición de Tipo de Documento (DTD – Document Type Definition) que se va a usar para definir, y por tanto validar, los datos que contiene nuestro documento XML.

La declaración puede incluir toda la Definición del Tipo del Documento, una parte de ella, o ninguna parte, siempre que se establezca una referencia a dónde encontrar la información no incluida sobre la definición mediante:

- un identificador público (PUBLIC) que hace referencia a dicha DTD, o
- un Identificador Universal de Recursos (URI) precedido por la palabra SYSTEM.

### Ejemplos:

```
<?xml version="1.0" encoding="UTF-7" standalone="yes"?>
<!DOCTYPE MENSAJE SYSTEM "mensaje.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<!DOCTYPE LABEL SYSTEM "http://www.empresa.com/dtds/label.dtd">
```

Referencia: [http://es.wikipedia.org/w/index.php?title=Extensible Markup Language&printable=yes](http://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&printable=yes)  
<http://www.desarrolloweb.com/articulos/prologo-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/document-type-definitions.html>)

### Partes de un documento XML (5)

#### Definiciones de Tipo de Documento (1)

Crear una definición de tipo de documento (DTD) es como crear nuestro propio lenguaje de marcado para una aplicación específica. Por ejemplo, podríamos crear una DTD que defina una tarjeta de visita. A partir de esa DTD, tendríamos una serie de elementos XML que nos permitirían definir tarjetas de visita.

Una definición de tipo de documento comienza en la primera línea y termina con **]>**. Contiene las definiciones de:

- elementos (comienzan por **<!ELEMENT** ),
- atributos, y
- entidades

permitidos para una DTD y puede expresar algunas limitaciones para combinarlos.

Referencia: <http://www.desarrolloweb.com/articulos/document-type-definitions.html>

## XML (<http://www.desarrolloweb.com/articulos/document-type-definitions.html>)

### Partes de un documento XML (6)

#### Definiciones de Tipo de Documento (2)

Los documentos XML que se ajustan a su DTD se denominan **válidos**. El concepto de "validez" no tiene nada que ver con el de estar "bien-formado":

- un documento "bien-formado" respeta, simplemente, la estructura y sintaxis definidas por la especificación de XML.
- un documento "bien-formado" puede, además, ser "válido" si cumple las reglas de una DTD determinada.
- pueden existir documentos XML sin una DTD asociada, en ese caso no son "válidos", pero tampoco "inválidos"... simplemente son "bien-formados"... o no.

La DTD puede residir en un fichero externo, y ser compartida por otros documentos (pueden ser miles), o bien estar contenida en el propio documento XML, como parte de su declaración de tipo de documento.

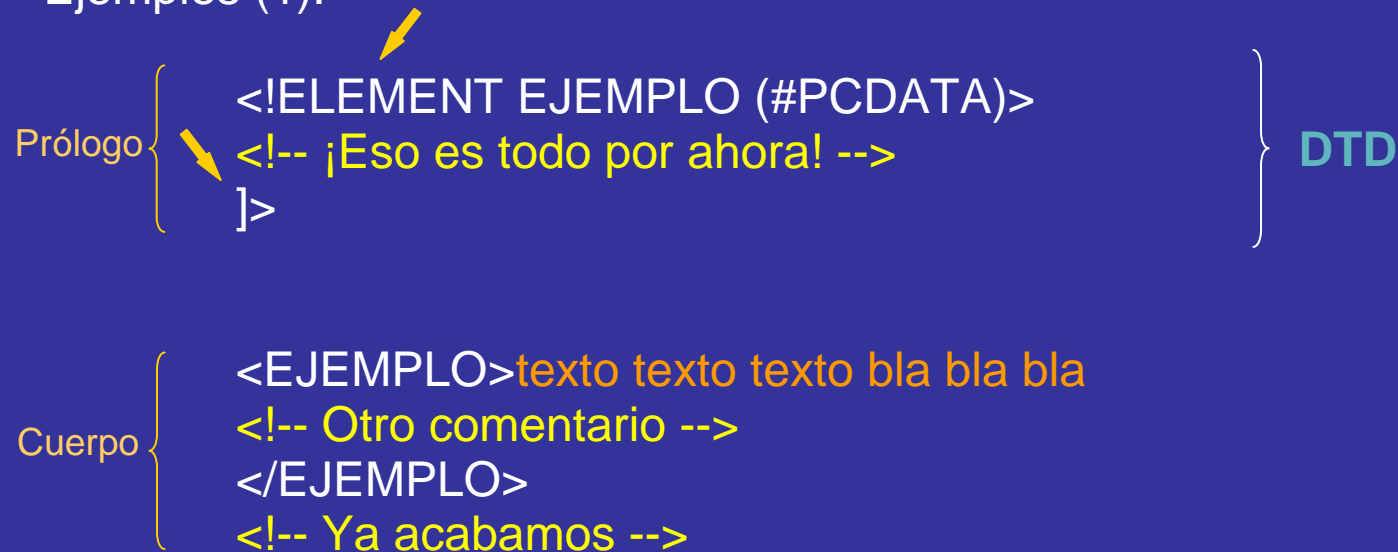
Referencia: <http://www.desarrolloweb.com/articulos/document-type-definitions.html>

## XML (<http://www.desarrolloweb.com/articulos/document-type-definitions.html>)

### Partes de un documento XML (7)

#### Definiciones de Tipo de Documento (3)

Ejemplos (1):



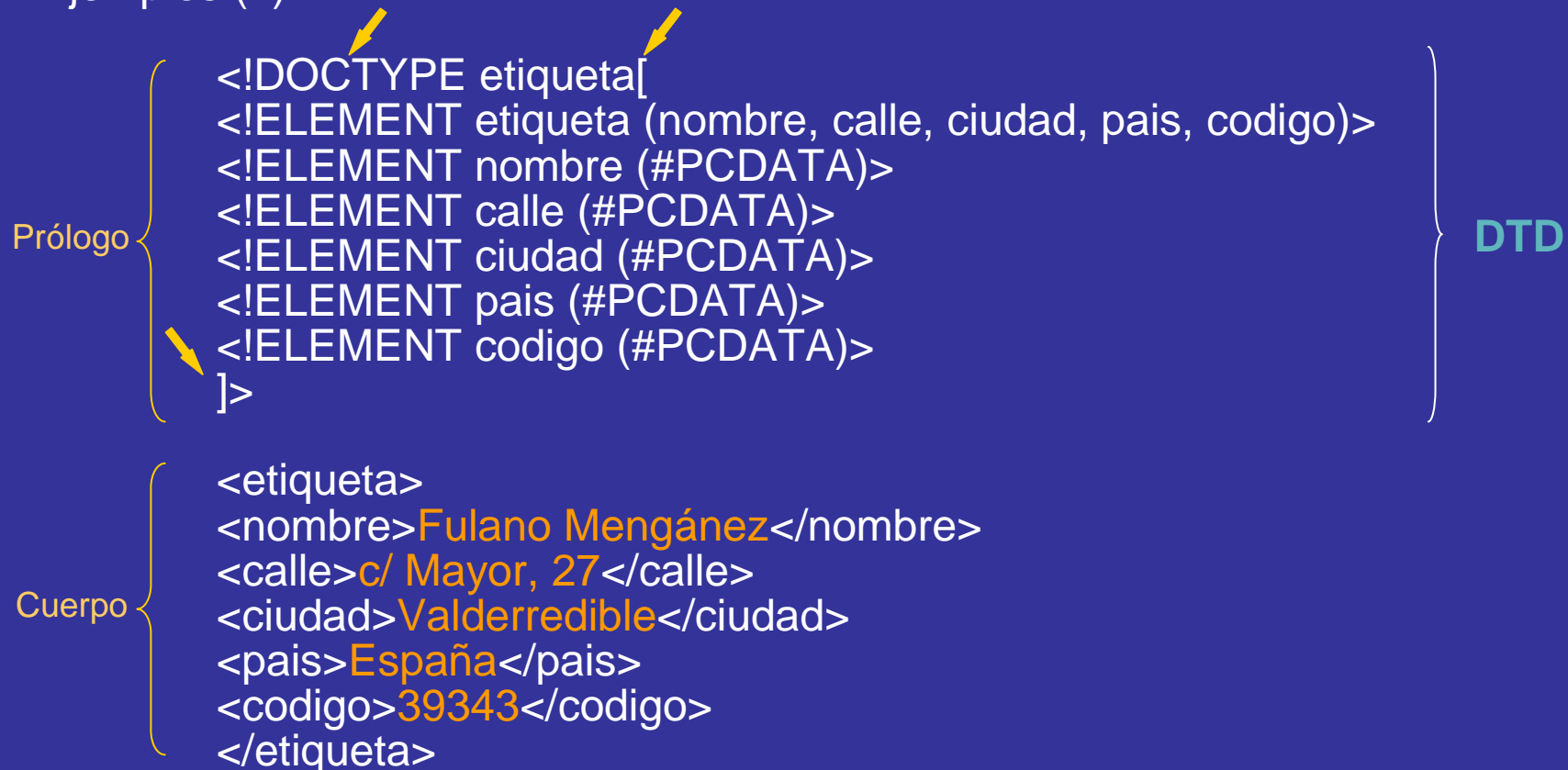
Referencia: <http://www.desarrolloweb.com/articulos/document-type-definitions.html>

## XML (<http://www.desarrolloweb.com/articulos/document-type-definitions.html>)

### Partes de un documento XML (8)

#### Definiciones de Tipo de Documento (4)

Ejemplos (2):



Referencia: <http://www.desarrolloweb.com/articulos/document-type-definitions.html>

## XML (<http://www.desarrolloweb.com/articulos/document-type-definitions.html>)

### Partes de un documento XML (9)

#### Definiciones de Tipo de Documento (5)

En el ejemplo anterior, todas las definiciones de la DTD que definen "etiqueta" residen dentro del documento. Sin embargo, la DTD se puede definir parcial o completamente en otro lugar. Por ejemplo:



Referencia: <http://www.desarrolloweb.com/articulos/document-type-definitions.html>

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>)

### Partes de un documento XML (10)

#### Definiciones de Tipo de Elemento (1)

Los elementos son la base de las marcas XML, y deben ajustarse a un tipo de documento declarado en una DTD para que el documento XML sea considerado válido.

Las definiciones de tipo de elemento deben empezar con **<!ELEMENT** seguidas por el identificador genérico del elemento que se declara y, a continuación, una especificación de su contenido.

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>)

### Partes de un documento XML (11)

#### Definiciones de Tipo de Elemento (2)

Ejemplo:

```
<!ELEMENT receta (titulo, ingredientes, procedimiento)>
```

En este ejemplo, el elemento <receta> puede contener dentro los elementos <titulo>, <ingredientes> y <procedimiento>, que, a su vez, estarán definidos también en la DTD y podrán contener más elementos.

Con esta definición, el siguiente documento XML sería válido:

```
<receta>  
<titulo>Paella</titulo>  
<ingredientes>arroz, tomate, pimiento, sal, ...</ingredientes>  
<procedimiento>Se fríe el ...</procedimiento>  
</receta>
```

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>



## XML (<http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>)

### Partes de un documento XML (12)

#### Definiciones de Tipo de Elemento (3)

La especificación de contenido de un elemento puede ser de cuatro tipos:

1.- EMPTY

2.- ANY

3.- Mixed

4.- Element

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>)

### Partes de un documento XML (13)

#### Definiciones de Tipo de Elemento (4)

##### 1.- EMPTY

Puede no tener contenido. Suele usarse para los atributos.

<!ELEMENT salto-de-pagina **EMPTY**>



##### 2.- ANY

Puede tener cualquier contenido. No se suele utilizar, ya que es conveniente estructurar adecuadamente nuestros documentos XML.

<!ELEMENT batiburrillo **ANY**>



Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>)


### Partes de un documento XML (14)

#### Definiciones de Tipo de Elemento (5)

##### 3.- Mixed

Puede tener caracteres de tipo datos, en cuyo caso el contenido se especifica con **#PCDATA**, o una mezcla de caracteres y sub-elementos especificados en la especificación de contenido mixto. Por ejemplo:

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



donde el primer elemento definido (<enfasis>) puede contener datos de carácter (#PCDATA) y el segundo (<parrafo>) puede contener tanto datos de carácter (#PCDATA) como sub-elementos de tipo <enfasis>.

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>)

### Partes de un documento XML (15)

#### Definiciones de Tipo de Elemento (6)

##### 4.- Element

Sólo puede contener sub-elementos especificados en la especificación de contenido.

<!ELEMENT mensaje (remitente, destinatario, texto)>

Para declarar que un tipo de elemento tenga contenido de elementos se especifica un modelo de contenido en lugar de una especificación de contenido mixto o una de las clases ya descritas.

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-tipo-elemento-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>)

### Partes de un documento XML (16)

#### Definiciones de Tipo de Elemento (7)

#### Modelos de Contenido de Elementos (1)

Un modelo de contenido es un patrón que establece los subelementos aceptados, y el orden en el que se aceptan. Se pueden presentar los siguientes casos:

1) Modelo sencillo: puede tener un sólo tipo de subelemento:

<!ELEMENT aviso (parrafo)>

Esto indica que <aviso> sólo puede contener un sólo <parrafo>.

2) Secuencia de más de un subelemento:

<!ELEMENT aviso (titulo, parrafo)>

La coma, en este caso, denota una secuencia. Es decir, el elemento <aviso> debe contener un <titulo> seguido de un <parrafo>.

Referencia: <http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>)

### Partes de un documento XML (17)

#### Definiciones de Tipo de Elemento (8)

#### Modelos de Contenido de Elementos (2)

#### 3) Subelementos alternativos

<!ELEMENT aviso (parrafo | grafico)>

La barra vertical | indica una alternativa, es decir, <aviso> puede contener o bien un <parrafo> o bien un <grafico>. El número de opciones no está limitado a dos, y se pueden agrupar usando paréntesis:

<!ELEMENT aviso (titulo, (parrafo | grafico))>

En este último caso, el <aviso> debe contener un <titulo> seguido de un <parrafo> o de un <grafico>.

Referencia: <http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>)

### Partes de un documento XML (18)

#### Definiciones de Tipo de Elemento (9)

#### Modelos de Contenido de Elementos (3)

Además, cada partícula de contenido puede llevar un **indicador de frecuencia**, que siguen directamente a un identificador general, una secuencia o una opción, y no pueden ir precedidos por espacios en blanco. Los indicadores de frecuencia son:

- ? Opcional (0 ó 1 vez)
- \* Opcional y repetible (0 ó más veces)
- + Necesario y repetible (1 ó más veces)

Ejemplo:

<!ELEMENT aviso (titulo?, (parrafo+, grafico)\*)>

En este caso, <aviso> puede tener <titulo>, o no (pero sólo uno), y puede tener cero o más conjuntos <parrafo><grafico>, <parrafo><parrafo><grafico>, etc.

Referencia: <http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>)

### Partes de un documento XML (19)

#### Definiciones de Listas de Atributos (1)

Los atributos permiten añadir información adicional a los elementos de un documento.

Diferencias entre elementos y atributos:

- los atributos no pueden contener subatributos: se usan sólo para añadir información corta, sencilla y desestructurada,
- cada atributo se puede especificar una sola vez, y en cualquier orden.

Referencia: <http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>



## XML (<http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>)

### Partes de un documento XML (20)

#### Definiciones de Listas de Atributos (2)

Ejemplo:

Cuerpo { `<mensaje prioridad="urgente">`  
          `<de>Alfredo Reino</de>`  
          `<a>Hans van Parijs</a>`  
          `<texto idioma="holandés">`  
          Hallo Hans, hoe gaat het?  
          ...  
          `</texto>`  
          `</mensaje>`

Las listas de atributos de los elementos `<mensaje>` y `<texto>` se definirían como:

Prólogo { `<!ELEMENT mensaje (de, a, texto)>`  
          `<!ATTLIST mensaje prioridad (normal | urgente) normal>`  
          `<!ELEMENT texto(#PCDATA)>`  
          `<!ATTLIST texto idioma CDATA #REQUIRED>`

Referencia: <http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>)

### Partes de un documento XML (21)

#### Definiciones de Listas de Atributos (3)

Las definiciones de los atributos:

- empiezan con **<!ATTLIST**,
- a continuación del espacio en blanco viene el identificador del elemento al que se aplica el atributo
- después viene el nombre del atributo, su tipo y su valor por defecto.

Referencia: <http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>)

### Partes de un documento XML (22)

#### Definiciones de Listas de Atributos (4)

##### Ejemplo:

Prólogo { `<!ELEMENT mensaje (de, a, texto)>`  
`<!ATTLIST mensaje prioridad (normal | urgente) normal>`  
`<!ELEMENT texto(#PCDATA)>`  
`<!ATTLIST texto idioma CDATA #REQUIRED>`

En este ejemplo, el atributo "prioridad" pertenece al elemento <mensaje> y puede tener el valor "normal" o "urgente", siendo "normal" el valor por defecto si no especificamos en el cuerpo el valor del atributo. El atributo "idioma", pertenece al elemento texto, y puede contener datos de tipo carácter (CDATA). La palabra **#REQUIRED** significa que no tiene valor por defecto y es obligatorio especificar en el cuerpo el valor de este atributo.

Referencia: <http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>)

### Partes de un documento XML (23)

#### Definiciones de Listas de Atributos (5)

A menudo interesa que se pueda omitir un atributo sin que se adopte automáticamente un valor por defecto. Para ésto se usa la condición **#IMPLIED**.

Ejemplo: En una supuesta DTD que defina la etiqueta <IMG> de HTML:

```
<!ATTLIST IMG URL CDATA #REQUIRED  
ALT CDATA #IMPLIED>
```

el atributo "URL" es obligatorio, mientras que el atributo "ALT" es opcional (y, si se omite, no toma ningún valor por defecto).

Referencia: <http://www.desarrolloweb.com/articulos/modelos-contenido-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/tipos-atributos-xml.html>)

### Partes de un documento XML (24)

#### Tipos de Atributos (1)

Los tipos de atributos son:

- CDATA y NMTOKEN
- enumerados y notaciones
- ID e IDREF

Referencia: <http://www.desarrolloweb.com/articulos/tipos-atributos-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/tipos-atributos-xml.html>)

### Partes de un documento XML (25)

#### Tipos de Atributos (2)

#### Atributos CDATA y NMTOKEN

Los atributos CDATA (character data) son los más sencillos y pueden contener casi cualquier cosa.

Los atributos NMTOKEN (name token) son parecidos, pero sólo aceptan los caracteres válidos para nombrar cosas (letras, números, puntos, guiones, subrayados y dos puntos).

#### Ejemplos:

```
<!ATTLIST mensaje fecha CDATA #REQUIRED>  
<mensaje fecha="15 de Julio de 1999">
```

- prólogo -  
- cuerpo -

```
<!ATTLIST mensaje fecha NMTOKEN #REQUIRED>  
<mensaje fecha="15-7-1999">
```

- prólogo -  
- cuerpo -

Referencia: <http://www.desarrolloweb.com/articulos/tipos-atributos-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/tipos-atributos-xml.html>)

### Partes de un documento XML (26)

#### Tipos de Atributos (3)

#### Atributos enumerados y notaciones (a)

Los atributos enumerados sólo pueden tener un valor de entre un número reducido de opciones.

Ejemplo:

```
<!ATTLIST mensaje prioridad (normal | urgente) normal>
```

El tipo **NOTATION** permite al autor declarar que su valor se ajusta a una notación declarada.

Ejemplo:

```
<!ATTLIST mensaje fecha NOTATION (ISO-DATE | EUROPEAN-DATE) #REQUIRED>
```

Referencia: <http://www.desarrolloweb.com/articulos/tipos-atributos-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/tipos-atributos-xml.html>)

### Partes de un documento XML (27)

#### Tipos de Atributos (4)

#### Atributos enumerados y notaciones (b)

Para declarar las notaciones se utiliza **<!NOTATION** con una definición externa de la notación. La definición externa puede ser:

- pública,
- un identificador del sistema para la documentación de la notación,
- una especificación formal, o
- un asistente de la aplicación que contenga objetos representados en la notación.

#### Ejemplos:

```
<!NOTATION HTML SYSTEM "http://www.w3.org/Markup">
```

```
<!NOTATION HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

Referencia: <http://www.desarrolloweb.com/articulos/tipos-atributos-xml.html>



## XML (<http://www.desarrolloweb.com/articulos/tipos-atributos-xml.html>)

### Partes de un documento XML (28)

#### Tipos de Atributos (5)

#### Atributos ID e IDREF

El tipo ID permite que un atributo determinado tenga un nombre único que podrá ser referenciado por un atributo de otro elemento que sea de tipo IDREF.

Ejemplo: Implementar un sencillo sistema de hipervínculos en un documento:

```
<!ELEMENT enlace EMPTY>  
<!ATTLIST enlace destino IDREF #REQUIRED>  
<!ELEMENT capitulo (parrafo)*>  
<!ATTLIST capitulo referencia ID #IMPLIED>
```

En este caso, una etiqueta <enlace destino="seccion-3"> haría referencia a un <capitulo referencia="seccion-3">, de forma que el procesador XML lo podría convertir en un hipervínculo, u otra cosa.

Referencia: <http://www.desarrolloweb.com/articulos/tipos-atributos-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>)

### Partes de un documento XML (29)

#### Definiciones de Entidades (1)

XML puede hacer referencia a objetos tales como ficheros, páginas web, imágenes, y, en general, cualquier cosa, que no deben ser analizados sintácticamente según las reglas de XML. Para ello usa entidades, que se declaran en la DTD mediante el uso de **<!ENTITY**

Ejemplos: Una entidad puede no ser más que una abreviatura que se utiliza como una forma corta de algunos textos. Cuando se usa una referencia a esta entidad, el analizador sintáctico reemplaza la referencia por su contenido. En otras ocasiones es una referencia a un objeto externo o local.

Las entidades pueden ser:

- Internas o Externas
- Analizadas o No analizadas
- Generales o Parámetro

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>)

### Partes de un documento XML (30)

#### Definiciones de Entidades (2)

#### Entidades generales internas

Son las más sencillas. Son, básicamente, abreviaturas definidas en la sección DTD del documento XML. ***Siempre son entidades analizadas***, es decir, una vez reemplazada la referencia a la entidad por su contenido, pasa a ser parte del documento XML y, como tal, es analizada por el procesador XML.

#### Ejemplo:

Prólogo { `<!DOCTYPE texto[  
<!ENTITY alf "Alien Life Form">  
>`

Cuerpo `<texto><titulo>Un día en la vida de un &alf;</titulo></texto>`

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>)

### Partes de un documento XML (31)

#### Definiciones de Entidades (3)

#### Entidades generales externas analizadas

Las entidades externas obtienen su contenido en cualquier otro sitio del sistema, ya sea otro archivo del disco duro, una página web o un objeto de una base de datos.

Se hace referencia al contenido de una entidad de esta clase mediante la palabra **SYSTEM** seguida de un URI (Universal Resource Identifier).

#### Ejemplo:

```
<!ENTITY intro SYSTEM "http://www.miservidor.com/intro.xml">
```

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>)

### Partes de un documento XML (32)

#### Definiciones de Entidades (4)

#### Entidades no analizadas

Este tipo de entidades siempre son generales y externas. Por ejemplo, si el contenido de la entidad es un archivo MPG, una imagen GIF, o un fichero ejecutable EXE, es evidente que el procesador XML no debería intentar interpretarlo como si fuera texto XML.

#### Ejemplo:

```
<!ENTITY logo SYSTEM "http://www.miservidor.com/logo.gif">
```

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>)

### Partes de un documento XML (33)

#### Definiciones de Entidades (5)

#### Entidades parámetro internas y externas (a)

Se denominan entidades parámetro a aquellas que sólo pueden usarse en la DTD, y no en el documento XML.

Se pueden utilizar para agrupar ciertos elementos del DTD que se repitan mucho.

Las entidades parámetro se diferencian de las generales en que, para hacer referencia a ellas, se usa el símbolo % en lugar de &, tanto para definir las o declararlas como para usarlas.

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>)

### Partes de un documento XML (34)

#### Definiciones de Entidades (6)

#### Entidades parámetro internas y externas (b)

##### Ejemplos:

```
<!DOCTYPE texto[
<!ENTITY % elemento-alf "<!ELEMENT ALF (#PCDATA)>">
...
%elemento-alf;
]>
```

También puede ser externa:

```
<!DOCTYPE texto[
<!ENTITY % elemento-alf SYSTEM "alf.ent">
...
%elemento-alf;
]>
```

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/estructuras-xml.html> )

### Partes de un documento XML (35)

#### Entidades predefinidas

En XML 1.0, se definen cinco entidades para representar caracteres especiales y que no se interpreten como marcado por el procesador XML. Es la manera de poder usar, por ejemplo, el carácter < sin que se interprete como el comienzo de una etiqueta XML. Estas entidades son:

Entidad	Carácter
&amp;	&
&lt;	<
&gt;	>
&apos;	'
&quot;	"

Referencia: <http://www.desarrolloweb.com/articulos/estructuras-xml.html>



## XML (<http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>)

### Partes de un documento XML (36)

#### Ejemplo de DTD (a)

Un ejemplo de DTD que puede servir para resumir todo lo visto hasta ahora podría ser una DTD que nos defina un lenguaje de marcado para una base de datos de personas con direcciones e-mail.

El fichero LISTIN.DTD podría ser algo así:

```
<?xml encoding="UTF-8"?>
<!ELEMENT listin (persona)+>
<!ELEMENT persona (nombre, email*, relacion?)>
<!ATTLIST persona id ID #REQUIRED>
<!ATTLIST persona sexo (hombre | mujer) #IMPLIED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT relacion EMPTY>
<!ATTLIST relacion amigo-de IDREFS #IMPLIED
enemigo-de IDREFS #IMPLIED>
```

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>)

### Partes de un documento XML (37)

#### Ejemplo de DTD (b)

Basándonos en esta DTD, podríamos escribir nuestro primer listín en XML de la siguiente manera:

```
<?xml version="1.0"?>
<!DOCTYPE listin SYSTEM "LISTIN.DTD">
<listin>
  <persona sexo="hombre" id="ricky">
    <nombre>Ricky Martin</nombre>
    <email>ricky@puerto-rico.com</email>
    <relacion amigo-de="laetitia">
  </persona>
  <persona sexo="mujer" id="laetitia">
    <nombre>Laetitia Casta</nombre>
    <email>castal@micasa.com</email>
  </persona>
</listin>
```

Referencia: <http://www.desarrolloweb.com/articulos/declaraciones-entidades-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/xml-schemas.html>)

### Partes de un documento XML (38)

### Esquemas XML (XML Schemas) (1)

Un esquema XML es algo similar a una DTD, es decir, define:

- qué elementos puede contener un documento XML,
- cómo están organizados, y
- qué atributos y de qué tipo pueden tener sus elementos.

La ventaja de los esquemas con respecto a las DTDs son:

- usan sintaxis de XML, al contrario que las DTDs.
- permiten especificar los tipos de datos.
- son extensibles.

Por ejemplo, un esquema permite definir el tipo del contenido de un elemento o de un atributo, y especificar si debe ser un número entero, una cadena de texto, una fecha, etc. Las DTDs no nos permiten hacer estas cosas.

Referencia: <http://www.desarrolloweb.com/articulos/xml-schemas.html>

## XML (<http://www.desarrolloweb.com/articulos/xml-schemas.html>)

### Partes de un documento XML (39)

### Esquemas XML (XML Schemas) (2)

Ejemplo (a): Documento XML y su esquema correspondiente

```
<documento xmlns="x-schema:personaSchema.xml">  
  <persona id="fulano">  
    <nombre>Fulano Menganez</nombre>  
  </persona>  
</documento>
```

En este documento se hace referencia a un espacio de nombres (namespace) llamado "x-schema:personaSchema.xml", es decir, le estamos diciendo al analizador sintáctico XML que valide el documento contra el schema "personaSchema.xml".

Referencia: <http://www.desarrolloweb.com/articulos/xml-schemas.html>

## XML (<http://www.desarrolloweb.com/articulos/xml-schemas.html>)

### Partes de un documento XML (40)

### Esquemas XML (XML Schemas) (3)

Ejemplo (b): Documento XML y su esquema correspondiente

El schema sería algo parecido a esto:

```
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
<AttributeType name='id' dt:type='string' required='yes'/>
<ElementType name='nombre' content='textOnly'/>
<ElementType name='persona' content='mixed'>
<attribute type='id'/>
<element type='nombre'/>
</ElementType>
<ElementType name='documento' content='eltOnly'>
<element type='persona'/>
</ElementType>
</Schema>
```

Referencia: <http://www.desarrolloweb.com/articulos/xml-schemas.html>

## XML (<http://www.desarrolloweb.com/articulos/xml-schemas.html>)

### Partes de un documento XML (41)

### Esquemas XML (XML Schemas) (4)

Ejemplo (c): Documento XML y su esquema correspondiente

El primer elemento del schema define dos espacios de nombre:

- el primero **xml-data** le dice al analizador que ésto es un schema y no otro documento XML cualquiera
- el segundo **datatypes** nos permite definir el tipo de elementos y atributos utilizando el prefijo **dt**.

Referencia: <http://www.desarrolloweb.com/articulos/xml-schemas.html>

## XML (<http://www.desarrolloweb.com/articulos/xml-schemas.html>)

### Partes de un documento XML (42)

### Esquemas XML (XML Schemas) (5)

Los esquemas XML pueden incluir las siguientes definiciones:

**ElementType.-** Define el tipo y contenido de un elemento, incluyendo los sub-elementos que pueda contener.

**AttributeType.-** Asigna un tipo y condiciones a un atributo.

**attribute.-** Declara que un atributo previamente definido por AttributeType puede aparecer como atributo de un elemento determinado.

**element.-** Declara que un elemento previamente definido por ElementType puede aparecer como contenido de otro elemento.

Referencia: <http://www.desarrolloweb.com/articulos/xml-schemas.html>

## XML (<http://www.desarrolloweb.com/articulos/xml-schemas.html>)

### Partes de un documento XML (43)

### Esquemas XML (XML Schemas) (6)

Para definir un schema es necesario empezar por los elementos más profundamente anidados dentro de la estructura jerárquica de elementos del documento XML, es decir, tenemos que trabajar "de dentro hacia fuera".

Dicho de otra manera, las declaraciones de tipo **ElementType** y **AttributeType** deben preceder a las declaraciones de contenido **element** y **attribute** correspondientes.

Más información: Referencia de schemas XML de Microsoft (<http://msdn.microsoft.com/xml/>)

Referencia: <http://www.desarrolloweb.com/articulos/xml-schemas.html>



**XML** (<http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html> )

## Partes de un documento XML (44)

### Etiquetas básicas (1)

XHTML tiene una serie de etiquetas, llamadas etiquetas básicas, tales como:

- párrafos,
- saltos de línea,
- títulos,
- citas,
- separadores horizontales, y
- comentarios.

Referencia: <http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html>

**XML** (<http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html> )

## Partes de un documento XML (45)

### Etiquetas básicas (2)

**Párrafos**.- Los párrafos estructuran el contenido. Contienen una o más frases relacionadas entre sí, igual que en el mundo real. Si queremos crear un párrafo, metemos el texto entre las etiquetas **<p>** y **</p>** . Ejemplo:

```
<p>
Hola, me llamo Luke Skywalker y soy piloto
de una X- Wing en el Rogue Squadron. También
soy un Jedi del Lado Luminoso de la Fuerza.
Mis maestros han sido Yoda y Obi -Wan Kenobi.
</p>
```

Si pruebas este ejemplo en el navegador, notarás que pasa por alto los saltos de línea. Podrías haber puesto todo el párrafo en la misma línea y hubieras obtenido el mismo resultado.

Referencia: <http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html>

**XML** (<http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html> )

## Partes de un documento XML (46)

### Etiquetas básicas (3)

**Saltos de línea.**- Hay veces que necesitamos forzar un salto de línea dentro de un párrafo. Para ello usamos la etiqueta **<br />**. Ejemplo:

```
<p>
Dark Chest of Wonders <br />
Seen through the eyes <br />
Of the one with pure heart <br />
Once , so long ago.
</p>
```

Aunque estéticamente podamos obtener el mismo resultado mediante párrafos (con **<p>** ) que con saltos de línea ( **<br />** ) de forma indiscriminada, debemos recordar que un documento XHTML utiliza un lenguaje semántico, es decir que lo importante para que esté bien estructurado es el significado de las etiquetas que utilizamos y no el efecto estético que generan.

Referencia: <http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html>

**XML** (<http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html> )

## Partes de un documento XML (47)

### Etiquetas básicas (4)

Títulos (headings) (1).- La información se puede agrupar y ordenar según el tamaño de los títulos. Para conseguir ésto, se tienen las etiquetas `<hx>` y `</hx>`, donde x es un número del 1 al 6:

- `<h1>` corresponde al título más importante (mayor tamaño) y sólo debería haber uno por página
- le siguen `<h2>`, `<h3>`, y así sucesivamente hasta `<h6>`

Los elementos de encabezado deben guardar un orden lógico y no se debe saltar ninguno de los niveles.

Referencia: <http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html>

## XML (<http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html> )

### Partes de un documento XML (48)

#### Etiquetas básicas (5)

Títulos (headings) (2).- Ejemplo: Consideremos la sección de enlaces de una página web en la que el título principal es **Mis links favoritos** y luego aparecen los links clasificados por secciones, cada una de ellas con un subtítulo diferente: Blogs, videojuegos, ... Incluso aparecen subcategorías dentro de una misma sección, como por ejemplo Aventuras y Arcades

```
<h1>Mis links favoritos</h1>
<h2>Blogs</h2>
<!-- bla bla bla -->
<h2>Videojuegos</h2>
<h3>Aventuras</h3>
<!-- bla bla bla -->
<h3>Arcades</h3>
<!-- bla bla bla -->
```

Referencia: <http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html>

**XML** (<http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html> )

## Partes de un documento XML (49)

### Etiquetas básicas (6)

**Citas** (1).- Hay tres etiquetas diferentes para escribir citas:

- `<blockquote>`,
- `<q>`, y
- `<cite>`

Con `<blockquote>` y `<q>` escribimos la cita en sí, mientras que con `<cite>` marcamos su origen (persona, libro, canción, etc.).

La diferencia entre `<blockquote>` y `<q>` es que `<blockquote>` se emplea para escribir citas largas (esta etiqueta contiene párrafos) en cambio `<q>` funciona a la inversa porque está hecha para escribir citas cortas y, entonces, va dentro de párrafos. Técnicamente hablando, `<blockquote>` es un elemento block, y `<q>` es inline, los elementos inline no pueden ir “suelos” en un documento xhtml

Referencia: <http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html>

**XML** (<http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html> )

## Partes de un documento XML (50)

### Etiquetas básicas (7)

#### Citas (2).- Ejemplo:

```
<p>Aquí os dejo un fragmento de la canción  
<cite>Die for Rock ?n? Roll </cite>, de Dover:</p>
```

```
<blockquote>  
<p>  
Everybody danced (while)<br />  
I was lying on the floor <br />  
I was ready to die <br />  
for Rock ?n? Roll <br />  
</p>  
</blockquote>
```

```
<p>Mi parte preferida es cuando dice lo de  
<q>I was ready to die [...] </q>.</p>
```

Referencia: <http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html>

**XML** (<http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html> )

## Partes de un documento XML (51)

### Etiquetas básicas (8)

**Separadores horizontales.**- Para los separadores horizontales se emplea la etiqueta `<hr />` (horizontal rules). Actualmente casi no se utilizan debido a que el empleo de estilos CSS permite crear bordes delimitadores muy interesantes.

Ejemplo:

```
<h2>Los videojuegos</h2>
<p>Bla bla bla?</p>
<hr />
<h2>Música</h2>
<p>Bla bla bla?</p>
```

Referencia: <http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html>



**XML** (<http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html> )

## Partes de un documento XML (52)

### Etiquetas básicas (9).-

**Comentarios**.- Los comentarios se utilizan para indicar partes del código de una página a modo de nota, pero son invisibles para el navegador. Sirven como guía para el desarrollador o como ayuda para usuarios que utilizan navegadores especiales, como por ejemplo los no videntes.

Para escribir un comentario en el código fuente, lo hacemos entre `<!--` y `-->`. Siempre se tiene que hacer en una sola línea.

Se pueden introducir comentarios en cualquier lugar de la instancia o del prólogo, pero nunca dentro de las declaraciones, etiquetas, u otros comentarios.

Ejemplo:

```
<!-- Esto es un comentario -->
```

Referencia: <http://www.desarrolloweb.com/articulos/etiquetas-basicas-xhtml.html>

## XML (<http://www.desarrolloweb.com/articulos/estructuras-xml.html> )

### Partes de un documento XML (53)

#### Secciones CDATA (1)

Existe otra construcción XML que permite especificar datos utilizando cualquier carácter, especial o no, sin que éste se interprete como marcado XML. Es la construcción llamada **CDATA** (Character Data).

CDATA permite leer fácilmente documentos XML sin tener que descifrar los códigos de entidades, especialmente cuando hay muchas en el texto.

Una sección CDATA empieza por **<![CDATA[** y termina por **]]>**

Dentro de una sección CDATA se puede poner cualquier cosa con la seguridad de que no será interpretada como algo que no es. No obstante *existe una excepción*, y es *la cadena **]]>*** con la que termina el bloque CDATA. Esta cadena *no puede utilizarse dentro de una sección CDATA*.

Referencia: <http://www.desarrolloweb.com/articulos/estructuras-xml.html>

## XML (<http://www.desarrolloweb.com/articulos/estructuras-xml.html> )

### Partes de un documento XML (54)

#### Secciones CDATA (2)

Ejemplo.- Se pretende que la única entidad XML sea <ejemplo> y que todo lo demás sea texto.

a) con entidades predefinidas:

```
<ejemplo>
&lt;HTML>
&lt;HEAD>&lt;TITLE>Rock & Roll&lt;/TITLE>&lt;/HEAD>
</ejemplo>
```

b) con CDATA

```
<ejemplo>
<![CDATA[
<HTML>
<HEAD><TITLE>Rock & Roll</TITLE></HEAD>
]]>
</ejemplo>
```

Referencia: <http://www.desarrolloweb.com/articulos/estructuras-xml.html>

**XML** (<http://www.desarrolloweb.com/articulos/extended-style-language.html> )

## **XSL (Extended Style Language) (1)**

XSL es un lenguaje que nos permite definir una presentación o formato para un documento XML.

Un mismo documento XML puede tener varias hojas de estilo XSL que lo muestren en diferentes formatos (HTML, PDF, RTF, VRML, PostScript, sonido, etc.)

La aplicación de una hoja de estilo XSL a un documento XML puede ocurrir tanto en el origen (por ejemplo, un servlet que convierta de XML a HTML para que sea mostrado a un navegador conectado a un servidor web), o en el mismo navegador (como es el caso del MS IE5, y Netscape 5).

**Básicamente, XSL es un lenguaje que define una transformación entre un documento XML de entrada, y otro documento XML de salida.**

Referencia: <http://www.desarrolloweb.com/articulos/extended-style-language.html>

**XML** (<http://www.desarrolloweb.com/articulos/extended-style-language.html> )

## **XSL (Extended Style Language) (2)**

Una hoja de estilo XSL es una serie de reglas que determinan cómo va a ocurrir la transformación. Cada regla se compone de un patrón (pattern) y una acción o plantilla (template)

Cada regla afecta a uno o varios elementos del documento XML. El efecto de las reglas es recursivo, para que un elemento situado dentro de otro elemento pueda ser también transformado.

Una hoja de estilo tiene una **regla raíz** que, además de ser procesada, llama a las reglas adecuadas para los elementos hijos.

Referencia: <http://www.desarrolloweb.com/articulos/extended-style-language.html>

**XML** (<http://www.desarrolloweb.com/articulos/extended-style-language.html> )

## **XSL (Extended Style Language) (3)**

Ejemplo (1).- Consideremos el siguiente documento XML:

```
<libro>
<titulo>Un título cualquiera</titulo>
<capitulos>
<capitulo>
<titulo>Capítulo 1</titulo>
<parrafo>....</parrafo>
<parrafo>....</parrafo>
</capitulo>
<capitulo>
<titulo>Capítulo 2</titulo>
...
</capitulo>
</capitulos>
</libro>
```

Referencia: <http://www.desarrolloweb.com/articulos/extended-style-language.html>

**XML** (<http://www.desarrolloweb.com/articulos/extended-style-language.html> )

## **XSL (Extended Style Language) (4)**

Ejemplo (2).- Queremos convertir el anterior documento XML a HTML bien-formado de manera que aparezca como:

```
<HTML>
<HEAD>
<TITLE>Un título cualquiera</TITLE>
</HEAD>
<BODY>
<H1>Un título cualquiera</H1>
<HR>
<H2>Capítulo 1</H2>
<P>...</P>
<P>...</P>
<HR>
<H2>Capítulo 2</H2>
<P>...</P>
</BODY>
</HTML>
```

Referencia: <http://www.desarrolloweb.com/articulos/extended-style-language.html>

**XML** (<http://www.desarrolloweb.com/articulos/extended-style-language.html> )

## XSL (Extended Style Language) (5)

Ejemplo (3).- La hoja de estilo XSL necesaria sería algo parecido a lo siguiente:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="libro">
    <HTML>
    <HEAD>
    <TITLE><xsl:process select="titulo"/></TITLE>
    </HEAD>
    <BODY>
    <H1><xsl:process select="titulo"/></H1>
    <xsl:process select="capitulos"/>
    </BODY>
    </HTML>
  </xsl:template>
  <xsl:template match="capitulos">
    <xsl:process select="capitulo">
    </xsl:template>
    <xsl:template match="capitulo">
    <HR/>
    <H2><xsl:process select="titulo"/></H2>
    <xsl:process select="parrafo"/>
    </xsl:template>
    <xsl:template match="parrafo">
    <P><xsl:process-children/></P>
    </xsl:template>
  </xsl:stylesheet>
```



Referencia: <http://www.desarrolloweb.com/articulos/extended-style-language.html>



**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (1)

### Sistemas de información Heterogéneos (1)

Un ejemplo clásico y muy importante es el uso de XML en la integración de sistemas de información heterogéneos.

En la actualidad el mercado está inundado de aplicaciones específicas y/o verticales que junto a la existencia de aplicaciones generales y/o horizontales, lleva a que muchas veces se deban integrar aplicaciones desarrolladas sobre plataformas, modelos de datos y lenguajes distintos.

Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (2)

### Sistemas de información Heterogéneos (2)

Habitualmente encontramos tres opciones cuando se plantea este tipo de problemas:

1. mantener las aplicaciones que funcionan correctamente e integrar con XML,
2. cambiar todos los sistemas para conseguir una integración “de fábrica”,
3. no integrar manteniendo las aplicaciones independientes,

y es lógico que la decisión que se tome deba sustentarse en criterios:

- tecnológicos, y
- en relación Coste - Beneficio.

Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (3)

### Sistemas de información Heterogéneos (3)

La tercera opción, aplicaciones no integradas, es muy problemática debido a las ineficiencias que se generan en los procesos de la empresa, por lo que esa opción se debe descartar aunque se encuentra en la realidad más veces de lo que sería aconsejable.

La segunda opción, cambiar todos los sistemas, tiene un impacto muy importante en cuanto a costes y en cuanto a cambios en las empresas, por lo que muchas veces también es desestimada.

Frente a estas dos opciones, el uso de XML permite desarrollar una solución integradora (MiddleWare) para que puedan comunicarse entre sí sistemas que ya están probados y funcionando correctamente, otorgando las ventajas de:

- lograr la mejor integración,
- con un coste contenido, y
- con resultados en corto plazo.

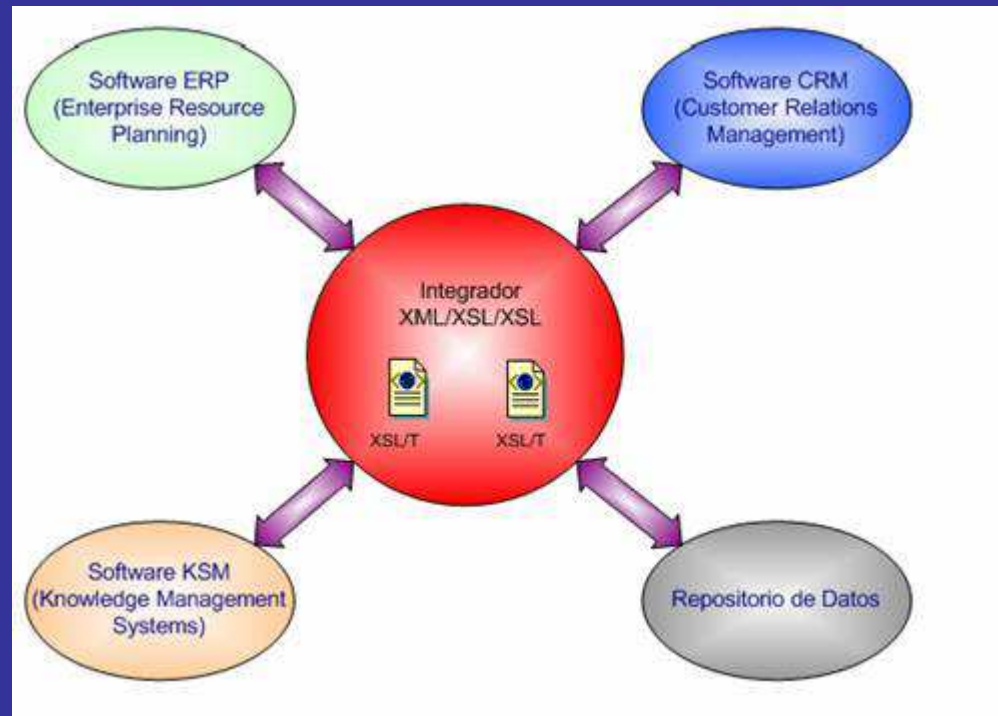
Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (4)

### Sistemas de información Heterogéneos (4)

En este sentido, la arquitectura que se presentaría en la integración de los usando XML sistemas podría ser:



Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (5)

### Gestión de catálogos en Internet (1)

XML también ofrece importantes oportunidades para la gestión de catálogos electrónicos a través de Internet.

Frente a otros lenguajes, XML permite que la gestión del contenido se limite sólo a su carga en base de datos y no hay que hacer artesanalmente cada página del catálogo.

Los catálogos se deben gestionar usando XML como medio de transporte de los datos de artículos, familias, categorías, descripciones, etc., y los formatos para su visualización estarán dados por XSL y su lenguaje XPath, que permite dinámicamente armar los contenidos de un catálogo.

De esta manera, encontramos distintos escenarios:

- un origen de datos y varias presentaciones posibles
- varios orígenes de datos y un modelo de presentación

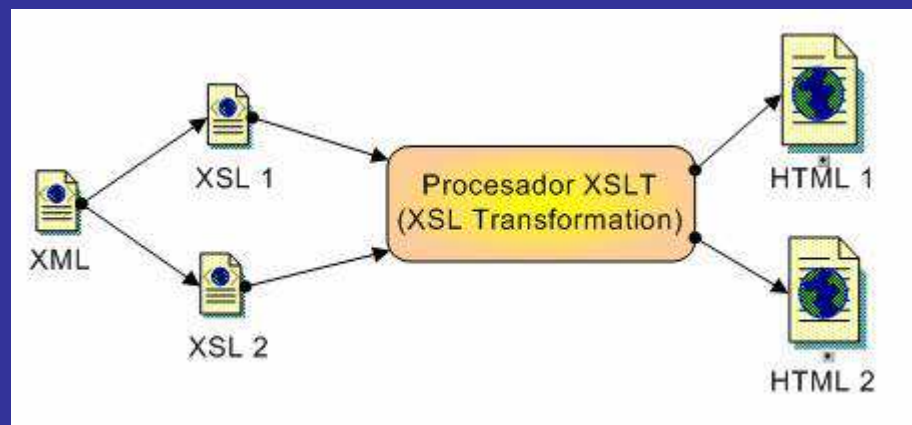
Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (6)

### Gestión de catálogos en Internet (2)

Escenario.- Un origen de datos y varias presentaciones posibles (distintas plantillas de presentación en función de determinados parámetros):



Este concepto permite dar formato visual distinto a los datos vertidos por un XML con determinada estructura, de modo que se puedan mostrar los datos bajo distintas plantillas. La potencia de este concepto permite gestionar múltiples tiendas electrónicas desde una única fuente de datos .

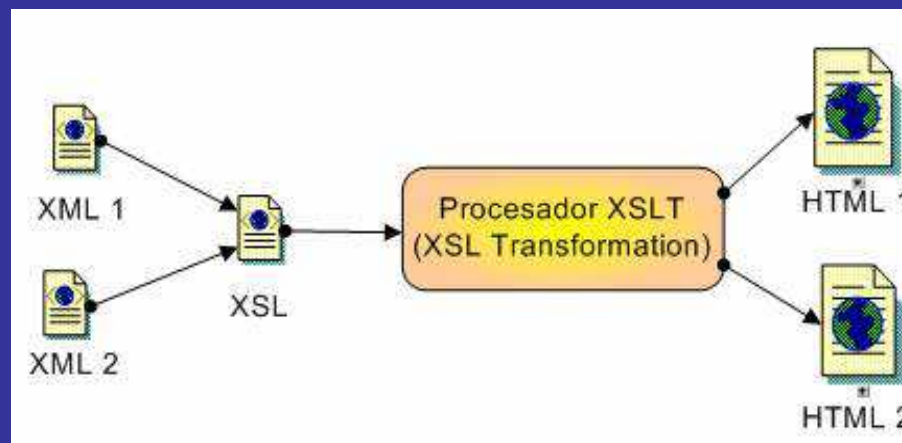
Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (7)

### Gestión de catálogos en Internet (3)

Escenario. - Varios orígenes de datos y un solo modelo de presentación (concepto de catálogo):



En este caso las estructuras de datos recibidas en XML se combinan con la plantilla en XSL dando como resultado una visualización similar para todos los datos que se reciban en dicha estructura. El catálogo electrónico es el caso por excelencia donde todos los productos con sus descripciones y características siempre se muestran con el mismo formato

Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (8)

### Dispositivos móviles

En la actualidad la movilidad del personal de una empresa es en muchos casos vital para su buen funcionamiento. La principal complicación en estos escenarios consiste normalmente en dotar al usuario del dispositivo móvil (empleado) de **información inmediata, oportuna y actualizada** proveniente del centro de datos.

Además, el usuario debe tener la **posibilidad de modificar** dicha información y actualizarla en el centro de datos sin tener que trasladarse físicamente, conectarse a la red y actualizar.

La tecnología móvil nos permite utilizar PDAs, Portátiles, teléfonos móviles, etc. que se pueden comunicar con un servidor intercambiando información en XML, WML y Servicios Web, y así optimizar la dinámica de la empresa contando con información fiable y actualizada en todo momento.

Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>



**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (9)

### Servicios Web (Web Services) (1)

Quizás la tecnología que más uso hace, y hará, de XML es la relacionada con los Servicios Web. Esta nueva forma de transmisión de datos permite la comunicación bidireccional, con lo cual se pueden establecer comunicaciones entre aplicaciones utilizando protocolos estándares basados en XML como SOAP (Simple Object Access Protocol), y especificaciones (aunque no estándares aún) como UDDI (Universal Description, Discovery and Integration) y WSDL (Web Service Definition Language).

Estas tecnologías encapsulan XML en paquetes de transmisión o mensajes (SOAP), permiten la ubicación en internet de los Servicios Web existentes cual si se tratase de unas Páginas Amarillas de Web Services (UDDI), y dan la posibilidad de que la aplicación que hace uso del Servicio Web comprenda las interfaces de comunicación de este último (WSDL).

Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (10)

### Servicios Web (Web Services) (2)

Este conjunto de definiciones permite que las aplicaciones distribuidas se basen en una tecnología abierta basada en estándares a diferencia de protocolos propietarios como DCOM o CORBA.

El resultado es la posibilidad de que aplicaciones de escritorio o Web se comuniquen con otras aplicaciones remotas para obtener datos, o gestionarlos, como si se tratase de una aplicación local, sin importar la plataforma en que se encuentra cada una en tanto se respeten los estándares citados.

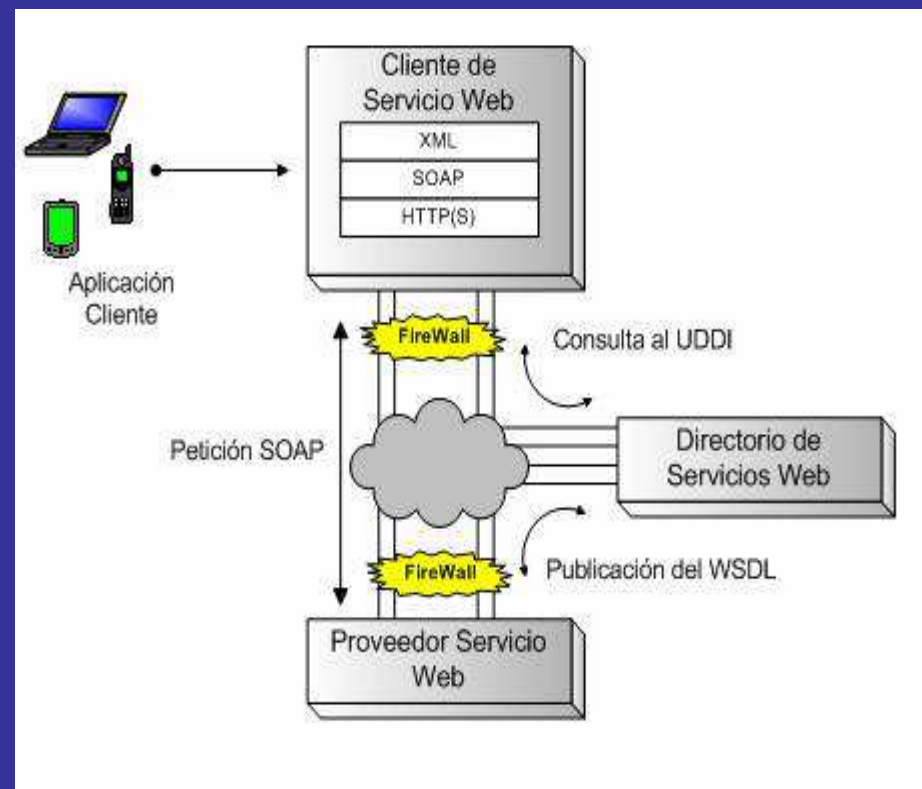
Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (11)

### Servicios Web (Web Services) (3)

Así, si nuestra aplicación requiriese de la cotización actual de una determinada empresa, se puede buscar mediante UDDI un Servicio Web que brinde dicha información, utilizar las interfaces programáticas del mismo con WSDL, y finalmente comunicar nuestra aplicación con el Web Service mediante SOAP, sin necesidad de tener que preocuparse por cortafuegos (firewalls) que pudieran interrumpir la comunicación por tratarse de un estándar basado en texto plano y que se comunica por un protocolo montado sobre HTTP.



Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (12)

### Moda o Solución (1)

Quizás el hecho de escuchar tanto acerca de XML, XSL, SOAP, Servicios Web, nos puede llevar a la conclusión errónea de que no es más que una simple moda producto de la escasez de resultados tecnológicos de los últimos tiempos.

La verdad es que XML es una tecnología que promete quedarse entre nosotros por mucho tiempo, por lo que puede asegurarse que no se trata de una moda sino de **una solución**.

Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (13)

### Moda o Solución (2)

Las principales empresas de software a nivel mundial han apostado todas sus fichas a la integración de sistemas y dispositivos. El objetivo a corto – medio plazo es que las aplicaciones sirvan datos que puedan ser visualizados indistintamente en páginas Web, Teléfonos móviles, PDAs, Portátiles, Televisión, Electrodomésticos, etc. Y la única forma posible de integración hasta el momento es la transmisión de datos por XML y la comunicación por SOAP (Servicios Web).

Un estudio realizado por Giga Group muestra que ya durante el año 2002 se usó XML en un 45% de las aplicaciones denominadas críticas, lo cual da clara idea del peso de esta tecnología en el desarrollo de aplicaciones actual.

Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>

**XML** (Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm> )

## XML en los negocios. Ejemplos (14)

### Conclusión

XML es una herramienta que se debe tener en cuenta a la hora de establecer soluciones tecnológicas. Los escenarios presentados en este artículo no son más que algunos de la infinidad de posibilidades que presenta este lenguaje.

XML tampoco es la solución ideal en todos los casos debido a que tiene su cuello de botella en la performance de ejecución y que los costes de ancho de banda no son los ideales para el uso de esta tecnología, aunque día a día se hacen más accesibles.

Asimismo las empresas líderes de tecnología como Sun, Microsoft, IBM, etc. soportan XML en todas las líneas de productos software y tienen sus propios marcos de trabajo (frameworks) para el desarrollo de Servicios Web XML.

La tendencia a futuro está claramente marcada.

Referencia: Marcelo Balbuena, <http://www.gestiopolis.com/canales2/gerencia/1/xml2.htm>