UNIDAD 1 RECONOCIMIENTO DE LAS CARACTERÍSTICAS DE LENGUAJES DE MARCAS

INTRODUCCIÓN LENGUAJE DE MARCAS.

Lenguaje de marcas es un tipo de lenguaje que combina texto con información extra acerca del texto. Esa información extra se entremezcla con el texto primario. El lenguaje de marcas más conocido en la actualidad es el HTML, que se utiliza en las páginas web.

Las marcas también están formadas de texto, pero que es interpretado cuando se muestra el documento, y suelen llamarse también etiquetas.

CARACTERÍSTICAS BÁSICAS DEL LENGUAJE, LENGUAJE DE MARCADO

Uno de los primeros problemas que se plantearon en informática era como traspasar o enviar información entre equipos. Para solventar esta situación se decidió codificar la información para su envió, ya que la mayoría de información que se manejaba era texto.

La codificación se aplica a un conjunto de caracteres de forma que cada carácter del conjunto tenga una asociación numérica. A estos conjuntos de caracteres se les denomina tablas, una de las tablas o conjuntos más conocidos y utilizados es el código ASCII, que relaciona cada carácter del conjunto con una combinación numérica en código binario, este código contempla caracteres alfabéticos, numéricos, de puntuación, etc.

Una vez solucionado el paso de información se planteo la posibilidad de enviar junto a la información el formato correspondiente a la misma, como atributos del texto, color, tamaño, alineación, etc. Una de las soluciones propuestas fue la de crear una serie de marcas que insertadas en la información a enviar establecieran dicho formato. A estas marcas se les llama Etiquetas y al conjunto de las mismas que determinan el formato se le conoce como un lenguaje de marcado.

El siguiente ejemplo pretende mostrar una demostración de un lenguaje de marcado. Suponga que se desea enviar la siguiente información, destacando algunas de sus palabras:

Hola **Mundo** ¡Buenos días!

Su codificación utilizando etiquetas podría ser la siguiente:

```
<texto>Hola <negrita>Mundo</negrita>¡Buenos días!</texto>
```

Como puede observar al incluir las etiquetas se puede establecer con bastante claridad que es lo que se desea resaltar.

Tanto HTML como XHTML son lenguajes de marcado como su nombre indica: HTML significa Hypertext Markup Language (Lenguaje de marcas e hipertexto), mientras que XHTML significa Extended HyperText Markup Language (Lenguaje de marcas e hipertexto extendido).

Lógicamente cada uno de los lenguajes de marcado tienen sus propias etiquetas, aparte de estos dos lenguajes citados anteriormente, existen muchos otros lenguajes como XML, MathML, DocBook, etc.

Normalmente los lenguajes de etiquetas suelen tener etiquetas de inicio y cierre para los diferentes formatos que se quieren aplicar. De esta forma se indica cuando tiene que empezar y finalizar el formato.

En el caso de HTML y XHTML el formato de sus etiquetas para expresar el principio y final de un formato es el siguiente:

Etiqueta de apertura o inicio del formato:

```
<Etiqueta>
```

La etiqueta anterior está compuesta por el símbolo “<” seguidamente y sin espacios en blanco el nombre de la etiqueta y a continuación el símbolo “>”.

Etiqueta de cierre o final del formato:

```
</Etiqueta>
```

La etiqueta de cierre está compuesta por el símbolo “<” seguidamente y sin espacios en blanco, el símbolo “/”, el nombre de la etiqueta y por último el símbolo “>”.

A continuación tiene una serie de ejemplos que se podrían aplicar en un documento XHTML para dar formato al texto:

```
<p>Esta etiqueta es para indicar que este texto es un párrafo</p>

<p>un párrafo con una <strong>palabra</strong> resaltada</p>

<p>visítanos en <em>www.aprendoencasa.com</em>, gracias</p>
```

IDENTIFICACIÓN DE ÁMBITOS DE APLICACIÓN

Los lenguajes de marcas tienen una amplia aplicación en multitud de ámbitos. Algunos de los más importantes son:

- **Marcado de documentos de tipo general**. Por ejemplo, Docbook es un lenguaje de marcas que permite etiquetar documentos para todo tipo de material y programas informáticos.
- **Tecnologías de Internet**. Sin duda es uno de los ámbitos de aplicación donde más se usan los lenguajes de marcas. De hecho, la evolución de la World Wide Web está directamente ligada a lenguajes de marcas como SGML (*Standard Generalized Markup Language*), HTML, XML, etc.
- **Lenguajes demarcas especializados**. Se aplícan a ámbitos específicos como las matemáticas (OpenMath), modelado de realidades virtuales (VRML) o los videojuegos (BulletML)...

CLASIFICACIÓN

TIPOS DE MARCADO

- El marcado procesal, consiste en códigos que contienen la información sobre cómo una aplicación específica debe procesar el documento.
- El marcado de presentación, consiste en códigos que describen cómo el documento se debe presentar o mostrar, sea en el monitor o en una página impresa.

- El marcado descriptivo, consiste en códigos que describen la estructura lógica y semántica de un documento, normalmente en una forma que pueda ser interpretada por diversas aplicaciones de software.

EL MARCADO PROCESAL

La mayoría de los sistemas electrónicos de publicación, como los procesadores de textos y los editores electrónicos, usan el marcado procesal.

El marcado procesal está enfocado hacia la presentación del texto. Este tipo de marcado se refiere a los caracteres especiales de control que se insertan en los archivos electrónicos de texto, antes de su presentación y la subsiguiente interpretación por los dispositivos de salida. Normalmente las marcas de formato se mezclan con el texto del documento.

Los códigos procesales se aplican a una manera sola de presentar la información, como por ejemplo, una página impresa, y no tienen la capacidad de definir la apariencia a otros medios de comunicación, tales como CD-ROM e Internet.

EL MARCADO DE PRESENTACIÓN

La marcación de presentación describe las características gráficas, de diseño y de control de la página, tanto en la pantalla del monitor como en una página impresa.

Una de las formas de codificación de presentación más extensamente usada es el HTML (Hyper Text Mark-up Language – Lenguaje de marcado de hipertexto).
El marcado HTML está entre paréntesis angulares < > y especifica títulos, párrafos, texto en negrita, listas, tablas, etc. La forma cómo se despliega cada uno de estos elementos depende del navegador usado para ver el documento.

Por lo que los códigos de presentación indican la manera de presentar la información pero son los medios son los que deciden finalmente.

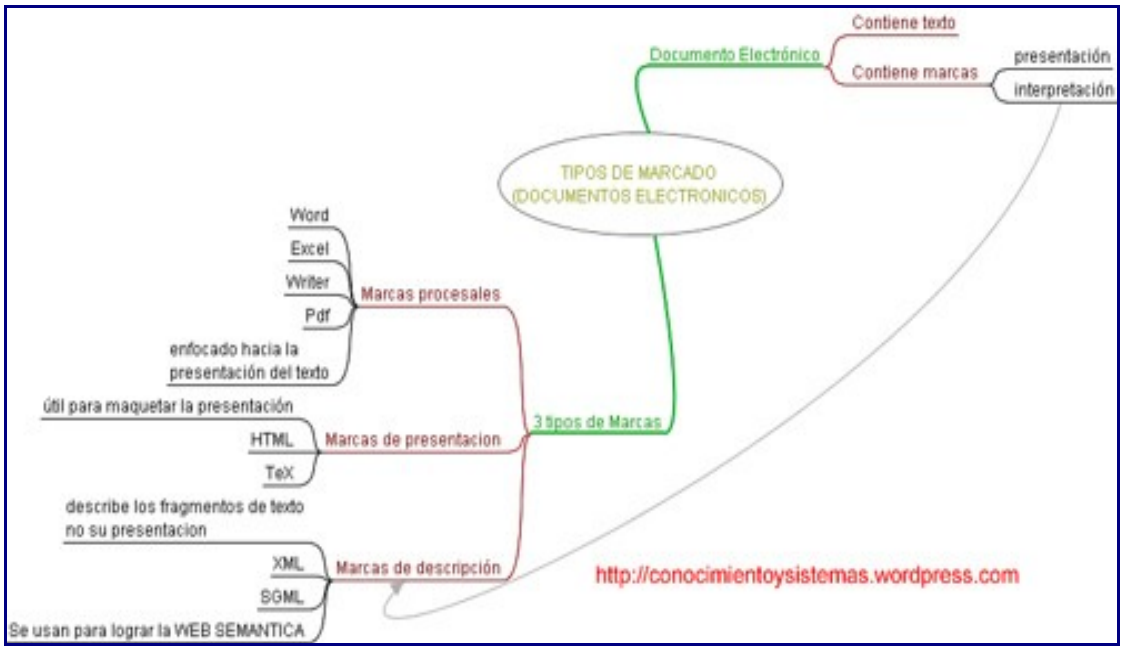
MARCADOR DESCRIPTIVO

En lugar de contener códigos que describen el diseño o presentación del documento, el marcado descriptivo contiene los códigos que definen una estructura lógica, Normalmente jerárquica.

Un ejemplo de este tipo de marcado es el XML (lenguaje de Marcado Extensible) que permite definir las propias estructuras del documento y lenguajes de marcado.

Los lenguajes de marcado descriptivo son la herramienta fundamental en el diseño de la web semántica, aquella que no solo permite acceder a la información, sino que además define su significado, de forma que sea más fácil su procesamiento automático y se pueda reutilizar para distintas aplicaciones.

MAPA MENTAL DE LOS TIPOS DE MARCADO DE UN DOCUMENTO ELECTRONICO



XML: ESTRUCTURA Y SINTAXIS

El XML (eXtensible Markup Language) es un meta-lenguaje que permite crear etiquetas adaptadas a las necesidades (de ahí lo de "extensible"). El estándar define cómo pueden ser esas etiquetas y qué se puede hacer con ellas. Es además especialmente estricto en cuanto a lo que está permitido y lo que no, todo documento debe cumplir dos condiciones: ser válido y estar bien formado.

El XML fue desarrollado por el World Wide Web Consortium, mediante un comité creado y dirigido por Jon Bosak. El objetivo principal era simplificar el SGML para adaptarlo a un campo muy preciso: documentos en internet.

Una de las principales ventajas de este tipo de codificación es que puede ser interpretada directamente, dado que son archivos de texto plano. Esto es una ventaja evidente respecto al los sistemas de archivos binarios, que requieren siempre de un programa intermediario para trabajar con ellos. Un documento escrito con lenguajes de marcado puede ser editado por un usuario con un sencilloeditor de textos, sin perjuicio de que se puedan utilizar programas más sofisticados que faciliten el trabajo.

Al tratarse solamente de texto, los documentos son independientes de la plataforma, sistema operativo o programa con el que fueron creados. Esta fue una de las premisas de los creadores de GML en lo años 70 , para no añadir restricciones innecesarias al intercambio de información. Es una de las razones fundamentales de la gran aceptación que han tenido en el pasado y del excelente futuro que se les augura.

Los fundamentos del XML son muy sencillos y en principio,para ir empezando, las únicas herramientas que nos harán falta son:

- Un **editor** para poder escribir los documentos XML, por ejemplo, el notepad en windows.
- Un **procesador** o **parser** XML, por ejemplo, usaremos los parsers que incorporan el Internet Explorer 5 (o superior) o el Mozilla (cualquier versión)

ESTRUCTURA

Aquí podemos ver un ejemplo muy sencillo:

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

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

<ficha>

<nombre> Angel </nombre>

<apellido> Barbero </apellido>

<direccion> c/Ulises, 36 </direccion>

</ficha>

Lo primero que tenemos que observar es la primera línea. Con ella deben empezar todos los documentos XML, ya que es la que indica que lo que la sigue es XML. Aunque es opcional, es recomendable incluirla. Puede tener varios atributos, algunos obligatorios y otros no:

- version: Indica la versión de XML usada en el documento. Es obligatorio ponerlo, a no ser que sea un documento externo a otro que ya lo incluía.
- encoding: La forma en que se ha codificado el documento. Se puede poner cualquiera, y depende del parser el entender o no la codificación. Por defecto es UTF-8, aunque podrían ponerse otras, como UTF-16, US-ASCII, ISO-8859-1, etc. No es obligatorio salvo que sea un documento externo a otro principal.
- standalone: Indica si el documento va acompañado de un DTD ("no"), o no lo necesita ("yes"); en principio no hay porqué ponerlo, porque luego se indica el DTD si se necesita.

La "declaración de tipo de documento" define qué tipo de documento estamos creando para ser procesado correctamente. Es decir, definimos que declaración de tipo de documento (DTD) valida y define los datos que contiene nuestro documento XML.

En ella se define el tipo de documento, y dónde encontrar la información sobre su Definición de Tipo de Documento, mediante:

- Un identificador público (PUBLIC): que hace referencia a dicha DTD.
- Identificador universal de recursos (URI): precedido de la palabra SYSTEM.

Ejemplos:

<!DOCTYPE MESAJE SYSTEM "mesaje.dtd">

<!DOCTYPE HTML PUBLIC "-/ /W3C/ /DTD HTML 3.2 Final/ /EN">

<!DOCTYPE LABEL SYSTEM "http://azuaje.ulpgc.es/dtds/label.dtd">

SINTAXIS

cabecera del documento se tiene que poner el texto

<?xml version="1.0"?>

En el resto del documento se deben escribir etiquetas como las de HTML, las etiquetas que nosotros queramos, por eso el lenguaje se llama XML, lenguaje de etiquetas extendido. Las etiquetas se escriben anidas, unas dentro de otras.

<ETIQ1>...<ETIQ2>...</ETIQ2>...</ETIQ1>

Cualquier etiqueta puede tener atributos. Le podemos poner los atributos que queramos.

<ETIQ atributo1="valor1" atributo2="valor2"...>

Los comentarios de XML se escriben igual que los de HTML.

<!-- Comentario -->

Y esto es todo lo que es el lenguaje XML en si, aunque tenemos que tener en cuenta que el XML tiene muchos otros lenguajes y tecnologías trabajando alrededor de él. Sin embargo, no cabe duda que la sintaxis XML es realmente reducida y sencilla.

Para definir qué etiquetas y atributos debemos utilizar al escribir en XML tenemos que fijarnos en la manera de guardar la información de una forma estructurada y ordenada. Por ejemplo, si deseamos guardar la información relacionada con una película en un documento XML podríamos utilizar un esquema con las siguientes etiquetas.

<?xml version="1.0"?>
<PELICULA nombre="El Padrino" año=1985>
<PERSONAL>
</DIRECTOR nombre="Georgie Lucar">
</INTERPRETE nombre="Marlon Brando" interpreta-a="Don Corleone">
</INTERPRETE nombre="Al Pacino" interpreta-a="Michael Corleone">
</PERSONAL>
</ARGUMENTO descripción="Pelicula de mafias sicilianas en Estados Unidos">
</PELICULA>

Como podéis ver, nos hemos inventado las etiquetas para poner este ejemplo y las hemos anidado de manera que la etiqueta más grande es la PELICULA y dentro de ella tenemos el PRESONAL y el ARGUMENTO. A su vez, dentro de PERSONAL tenemos tanto al DIRECTOR como a los actores (INTERPRETE).

ETIQUETAS

Ejemplo.

Imagina que diriges una clínica veterinaria y que desea utilizar XML para almacenar los datos de los distintos animales que tratas. Los archivos de datos XML contendrán los datos de cada animal. Cada fragmento de datos está rodeado por una etiqueta y cada una de esas etiquetas describe el significado de ese fragmento de datos. La combinación de etiqueta y datos se denomina **nodo**.

La ilustración presenta un archivo de datos XML de ejemplo de un gato cuyo nombre es Izzy. Las etiquetas son las combinaciones de corchetes angulares y texto:

```
<GATO>, <NOMBRE>, <EDAD>
```

y así sucesivamente.

En la práctica, las etiquetas constan de dos partes, una **etiqueta de apertura** y una **etiqueta de cierre**, de esta manera:

```
<RAZA> … </RAZA>
```

La barra diagonal (/) es lo que convierte una etiqueta en etiqueta de cierre. Los datos van rodeados por las etiquetas de apertura y cierre, de esta forma:

```
<RAZA>Siamés</RAZA>.
```

En XML, las etiquetas se diseñan para describir claramente los datos que contienen. Si alguien le pregunta qué significan todas esas etiquetas, puede contestar que significan lo que usted necesita que signifiquen. Esa característica es parte de lo que hace que el lenguaje XML sea "extensible". En este caso, usted sabe lo que "sí", "no" e "Izzl138bod" significa.

El hecho de que las etiquetas describan la estructura y el significado de los datos sirve para que cualquier programa o sistema informático que admita el uso de XML pueda comprender los datos y utilizarlos. Por ejemplo, leyendo los datos, podría cargar el nombre del gato y el nombre del propietario en un informe de vacunación y realizar una solicitud de pago al mismo tiempo.

Estos son sólo un par de ejemplos de cómo puede poner a trabajar el lenguaje XML.. Se pueden utilizar los datos en informes, páginas Web y bases de datos,.

DOCUMENTOS XML BIEN FORMADOS

Un documento XML tiene dos estructuras, una lógica y otra física. Físicamente, el documento está compuesto por unidades llamadas entidades. Una entidad puede hacer referencia a otra entidad, causando que esta se incluya en el documento. Cada documento comienza con una entidad documento, también llamada raíz. Lógicamente, el documento está compuesto de declaraciones, elementos, comentarios, referencias a caracteres e instrucciones de procesamiento, todos los cuales están indicados por una marca explícita. Las estructuras lógica y física deben encajar de manera adecuada:

Los documentos XML se dividen en dos grupos, documentos bien formados y documentos válidos.

- Bien formados: Son todos los que cumplen las especificaciones del lenguaje respecto a las reglas sintácticas sin estar sujetos a unos elementos fijados en un DTD. De hecho

los documentos XML deben tener una estructura jerárquica muy estricta y los documentos bien formados deben cumplirla.

- Válidos: Además de estar bien formados, siguen una estructura y una semántica determinada por un DTD: sus elementos y sobre todo la estructura jerárquica que define el DTD, además de los atributos, deben ajustarse a lo que el DTD dicte. Un documento XML se dice que está bien formado si encaja con las especificaciones XML de producción, lo que implica:

Estructura jerárquica de elementos

Los documentos XML deben seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que que delimitan sus elementos. Una etiqueta debe estar correctamente "incluida" en otra. Asímismo, los elementos con contenido, deben estar correctamente "cerrados". A continuació se muestra un ejemplo incorrecto y posteriormente otro ejemplo escrito correctamente.

HTML permite esto .

En XML la estructura es jerárquica .

Etiquetas vacías

HTML permite elementos sin contenido. XML también, pero la etiqueta debe ser de la siguiente forma <elemento sin contenido/>. A continuación se muestra un ejemplo incorrecto y posteriormente otro correcto.

Esto es HTML
 en el que casi todo está permitido

En XML, es
 más restrictivo.

Un solo elemento raiz

Los documentos XML sólo permiten un elemento raiz, del que todos los demás sean parte. Es decir, la jerarquía de elemento de un documento XML bien formado sólo puede tener un elemento inicial.

Valores de atributos

Los valores de atributos en XML siempre deben estar encerradas entre comillas simples (') o doble ("). En la siguiente ejemplo, la primera línea sería incorrecta en XML, no así la segunda:

<a HREF=http://www.dis.ulpgc.es/>

<a HREF="http://www.dis.ulpgc.es/">

Tipos de letras, espacios en blanco

El XML es sensible al tipo de letra que se utiliza, es decir, trata las mayúsculas y minúsculas como caracteres diferentes. Por lo tanto, los elemento definidos como "FICHA", "Ficha", "ficha" y "fiCha" serían elementos diferentes.

Esite un conjunto de caracteres denominados "espacios en blanco" que los procesadores XML tratan de forma diferente en el marcado XML. Estos caracteres son los ":espacios", tabuladores, retornos de carro y los saltos de línea.

La especificación XML 1.0 permite el uso de esos "espacios en blanco" para hacer más legible el código, y en general son ignorados por los procesadores XML.

En estos casos, sin embargo, los "espacios en blanco" resultan muy significativos, por ejemplo, para separar las palabras de un texto, o separa líneas de párrafos diferentes.

Nombrando cosas

Al utilizar XML, es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen algunas características en común.

No se pueden crear nombres que empiecen con la cadena "xml", "xML", "XML" o cualquier otra variante. 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.

Marcado y datos

Las construcciones con etiquetas, referencias de entidad y declaraciones se denominan "marcas".Éstas son las partes del documento que el procesador XML espera entender. El resto del documento que se encuentra entre las marcas son los datos que resultan entendibles por las personas.

Es sencillo reconocer las marcas en un documento XML. Son aquellas porciones que empiezan con "<" y acaban con ">", o bien, en el caso de las referencias de entidad, empiezan por "&" y acaban con ";".

ELEMENTOS

Los elementos XML pueden tener contenido (más elementos, caracteres, o ambos a la vez), o bien ser elementos vacíos.

Un elemento con contenido es, por ejemplo:

<nombre>Fernando Damián</nombre>

<aviso tipo="emergencia" gravedad="mortal">Que no cunda el pánico</aviso>

Siempre empieza con una <etiqueta> que puede contener atributos o no, y termina con una </etiqueta> que debe tener el mismo nombre. Al contrario que HTML, en XML siempre se debe "cerrar" un elemento.

Hay que tener en cuenta que el símbolo "<" siempre se interpreta como inicio de una etiqueta XML. Si no es el caso, el documento no estará bien formado.

Un elemento vacío, es el que no tiene contenido. 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. La sintaxis de HTML permite etiquetas vacías tipo <hr> o <img src="..."> En HTML reformulado para que sea un documento XML bien formado, se debería usar <hr/> o

DEFINICIÓN DE TIPO DE DOCUMENTOS (DTD)

La DTD es una definición, en un documento XML, que especifica restricciones en la estructura y sintaxis del mismo. La DTD se puede incluir dentro del archivo del documento, pero normalmente se almacena en un fichero separado.

Ejemplo DTD interno

```
<?xml version = "1.0"?>
<?xml-stylesheet href="libro.css" type= "text/css"?>
<!DOCTYPE agendas [
<!ELEMENT agendas(nombre,dirección,teléfono,codigo_p)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT direccion (#PCDATA)>
<!ELEMENT telefono (#PCDATA)>
]>
<agendas>
<nombre>paco</nombre>
<direccion>calle 1</direccion>
<telefono>666666666</telefono>
<codigo_p>36666</codigo_p>
</agendas>
```

Ejemplo DTD externo

Agendas.XML:

```
<?xml version = "1.0"?>
<?xml-stylesheet href="libro.css" type= "text/css"?>
<!DOCTYPE agendas SYSTEM “agenda.dtd” > // Llamada a la DTD
<agendas>
<nombre>paco</nombre>
<direccion>calle 1</direccion>
<telefono>666666666</telefono>
<codigo_p>36666</codigo_p>
</agendas>
```

Agenda.dtd:

```
<!ELEMENT agendas(nombre,dirección,teléfono,codigo_p)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT direccion (#PCDATA)>
<!ELEMENT telefono (#PCDATA)>
<!ELEMENT codigo_p (#PCDATA)>
```

En las DTD podemos hacer 3 tipos de declaraciones:

1-**Declaraciones de tipo de elemento**: Estas declaraciones establecen que elementos pueden formar parte del documento y cuales pueden formar parte de su interior.
Sintaxis: Los subelementos que puede contener cada elemento van siempre encerrados entre paréntesis y precedidos por la etiqueta <!ELEMENT>, dentro de las etiquetas cada elemento podrá llevar uno de los siguientes símbolos:

Signo	**Significado**
,	Secuencia de elementos
¿	0 o 1 ocurrencia
*	0 o más ocurrencias
+	1 o más ocurrencias
EMPTY	Que el elemento está vacío
ANY	Cualquier contenido es válido
#PCDATA	Que el contenido puede ser una cadena de texto

Ejemplos:

```
<!ELEMENTS nombre (#PCDATA)>
<!ELEMENT nombre EMPTY>
<!ELEMENT Cliente (nombre,apellidos,nif?,teléfono*,dirección+)>
```

También se puede hacer una redefinición de los subelementos del siguiente modo:

```
<!ELEMENTS nombre (#PCDATA)>
<!ELEMENTS apellidos(ape1,ape2?)>
<!ELEMENTS ape1 (#PCDATA)>
<!ELEMENTS ape2 (#PCDATA)>
```

2- **Declaraciones de listas de atributos**: Para definirlos se usan las declaraciones de lista de atributos, cuya sintaxis es la siguiente.
Sintaxis: Todas las declaraciones de atributos empiezan por <!ATTLIST>. Cada atributo está formado por 3 partes, que son, el nombre del atributo, el tipo y el valor por defecto.

Tipo de atributo:

Valor	Significado
CDATA	El atributo será una cadena de caracteres.
ID	El atributo sirve para identificar el elemento dentro del documento.
IDREF/S	Este atributo se empleará para referenciar otros elementos.
ENTITY/S	Contiene nombres de entidades.
NMTOKEN/S	Una única cadena de texto
<<enumerados>>	El conjunto de valores que puede tomar el atributo (|)

Valores por defecto:

Valor	Significado

#REQUIRED	Es obligatorio darle un valor al atributo
#IMPLIED	Es opcional darle el valor al atributo
<<valor>>	Valores -> si no se le da asumirá un valor por defecto
#FIXED <<valor>>	Con esto obligamos a que el atributo tome necesariamente el valor dado en <<valor>>

Ejemplos:

<!ELEMENT mensaje (#PCDATA)>
<!ATTLIST mensaje prioridad(normal | urgente) “normal”>

<!ELEMENT texto (#PCDATA)>
<!ATTLIST texto idioma CDATA #REQUIRED>

<!ELEMENT pago EMPTY>
<!ATTLIST pago metodo #IMPLIED>
Válido<pago método= “tarjeta” />
Válido<pago>

<!ELEMENT pago EMPTY>
<!ATTLIST pago metodo CDATA (#REQUIRED)>
Válido<pago metodo= “tarjeta” />

No Válido<pago>

<!ATTLIST semáforo color(rojo|amarillo|verde) “rojo”>

<!ATTLIST semáforo color (R|A|V) (#FIXED) ”R”>

3- Declaración de entidades: XML hace referencia a objetos que no deben ser analizados sintácticamente según las reglas de XML, mediante el uso de entidades. Se declaran en la DTD, mediante el uso de <!ENTITY>.
Una entidad puede no ser más que una abreviatura que se utiliza como una forma corta de algunos textos. Al usar una referencia a esta entidad, el analizador reemplazará la referencia con su contenido, por ejemplo:

<!DOCTYPE texto [
<!ENTITY alf “Alien Life Form” >
]>
-- - - - - - - - -
<texto><titulo>Un dia en la vida de un &alf</titulo></texto>

LOS ESPACIOS DE NOMBRES (NAMESPACES)

XML es un estándar diseñado para permitir que se comparta información con facilidad. ¿Qué pasaría si uniésemos información en XML procedente de dos fuentes diferentes para enviarla a una tercera? ¿Podríamos en ese caso tener algún conflicto por coincidencia del nombre de las etiquetas? Imaginemos el siguiente caso: un proveedor de Internet guarda todos sus datos en XML. La sección comercial almacena la dirección de la vivienda del cliente en un campo llamado <direccion>. Por otro lado, el servicio de asistencia al cliente guarda en <direccion> la dirección de correo electrónico del cliente, y finalmente el centro de control de red guarda en <direccion> la dirección IP del ordenador del cliente. Si unimos en un sólo fichero lo procedente de las tres divisiones de la empresa, nos podemos encontrar con:

```
<cliente>
...
```

```
<direccion>Calle Real</direccion>
...
<direccion>ventas@cliente.com
Esta dirección de correo electrónico está
siendo protegida de \"spam bots\",
necesitas habilitar Javascript para poder verlo.
</direccion>
...
lt;direccion>192.168.168.192</direccion>
...
</cliente>
```

Evidentemente en este caso tendríamos un problema, ya que no podríamos distinguir el significado de <direccion> en cada uno de los casos. Para ello, en 1999 el W3C definió una extensión de XML llamada espacios de nombres (namespaces) que permite resolver conflictos y ambigüedades de este tipo.

USO DE LOS ESPACIOS DE NOMBRES

Los espacios de nombres son un prefijo que ponemos a las **etiquetas de XML** para indicar a qué contexto se refiere la etiqueta en cuestión. En el ejemplo anterior podríamos definir:

<red:direccion>: para uso por el centro de control de red.
<aten:direccion>: para uso por el servicio de atención al cliente.
<comer:direccion>: para uso por el departamento comercial.

De esta forma, nuestro elemento queda compuesto así:

```
<cliente>
...
<comer:direccion>Calle Real</comer:direccion>
....
<aten:direccion>ventas@cliente.com
Esta dirección de correo electrónico
está siendo protegida de \"spam bots\",
necesitas habilitar Javascript para poder verlo.
</aten:direccion>
....
<red:direccion>192.168.168.192</red:direccion>
...
</cliente>
```

Para usar un espacio de nombres en un documento, debemos declararlo previamente. Esta declaración puede tener lugar en el elemento raíz del documento de la siguiente forma:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<carteraclientes
xmlns:comer="http://www.empresa.com/comercial"
xmlns:aten="http://www.empresa.com/atencion"
xmlns:red="http://www.empresa.com/red">
<cliente>
...
<comer:direccion>Calle Real</comer:direccion>
...
<aten:direccion>ventas@cliente.com
Esta dirección de correo electrónico
está siendo protegida de \"spam bots\",
necesitas habilitar Javascript para poder verlo.
</aten:direccion>
```

```
...
<red:direccion>192.168.168.192</red:direccion>
...
</cliente>
</carteraclientes>
```

La definición consta de unos atributos xmlns (XML namespace), donde proporcionamos el prefijo que usaremos para el espacio de nombres y una URI (Uniform Resource Identifier) que será un identificador único del espacio de nombres.