

UNIDAD 3

TECNOLOGÍAS PARA LA WEB

Elementos de XML

Definición del Tipo de Documento (DTD)

Esquema de documentos XML (XML Schema)

Modelo de Objetos del Documento (DOM)

Familia de dialectos XML

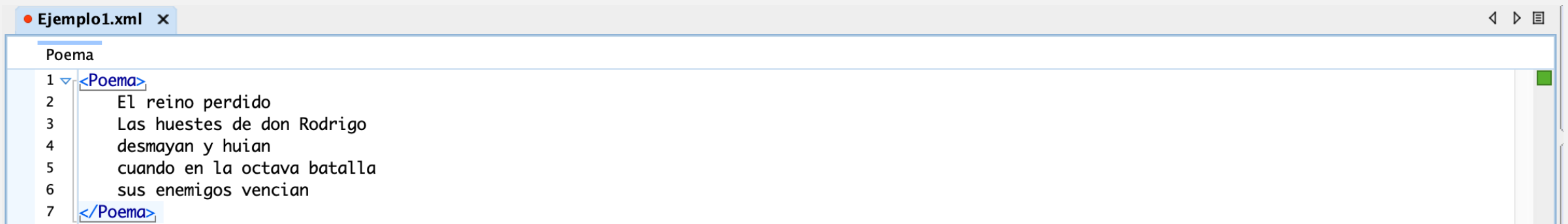
ELEMENTOS

XML usa el término elemento para hacer referencia a cada uno de los componentes estructurales o secciones de un documento XML. En un documento XML.

Un elemento queda delimitado por una marca o etiqueta de inicio y por una marca o etiqueta de fin. Estas dos marcas comparten el mismo nombre. Para diferenciar qué marca determina el inicio de un elemento y qué marca determina su fin, se tiene que escribir el carácter / a continuación del carácter < en la etiqueta que marca el final de la sección o elemento. El elemento comprende las marcas que determinan su inicio y su fin, así como el contenido que esté escrito entre estas dos marcas. Un elemento puede contener datos de tipo carácter, y las marcas de inicio y de fin de otros elementos.

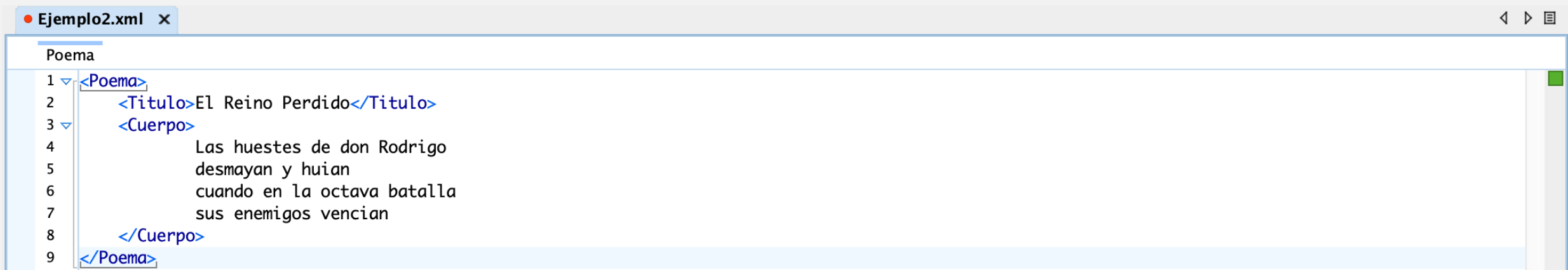
Las distintas piezas de información en las que podemos dividir un documento reciben, en XML, el nombre de elementos, son los ladrillos, las piezas básicas del rompecabezas. Una definición más formal podría describir los elementos como "la unidad lógica básica con capacidad para representar la estructura lógica y la semántica de un documento XML, en definitiva, su contenido".

Ejemplo.



```
<?xml version="1.0" encoding="UTF-8" />
<Poema>
  El reino perdido
  Las huestes de don Rodrigo
  desmayan y huian
  cuando en la octava batalla
  sus enemigos vencian
</Poema>
```

Éste es un documento XML muy sencillo, está reducido al mínimo y tiene un único elemento, "Poema", gracias al cual podríamos saber que el documento contiene un poema, y podríamos diferenciarle de otros documentos como cartas, pedidos, recetas, etc. Lo cierto es que la información que añade este elemento no es demasiada y seguramente no queramos conformarnos con eso. El documento de partida se puede refinar para añadirle otros elementos (en **negrita**).



```
<?xml version="1.0" encoding="UTF-8" ?>
<Poema>
  <Titulo>El Reino Perdido</Titulo>
  <Cuerpo>
    Las huestes de don Rodrigo
    desmayan y huian
    cuando en la octava batalla
    sus enemigos vencian
  </Cuerpo>
</Poema>
```

En este segundo ejemplo existen elementos que identifican el título y el cuerpo del poema, con su ayuda, seríamos capaces, por ejemplo, de listar todos los títulos de los poemas de los que disponemos, o buscar un título determinado y mostrar el cuerpo del poema. Las piezas de texto que forman el título y el cuerpo están perfectamente identificadas y delimitadas. Este proceso de refinamiento y adición de elementos podría continuarse mientras fuera necesario.

UN ELEMENTO DE TRES PARTES

- Se puede observar en los ejemplos que los elementos están formados por tres componentes, una etiqueta de inicio. por ejemplo "<Poema>", una etiqueta de finalización. por ejemplo "</Poema>". y un contenido, situado entre ambas etiquetas y que en este caso es un romance. Hay que notar que, al contrario de lo que sucede en HTML, todos los elementos de XML. están formados por estas tres partes, las etiquetas no son el elemento en sí, son una parte de él que lo delimita y que lo marca. El hecho de olvidar una etiqueta de finalización o de inicio implica un error de buena formación. Una buena costumbre para no olvidar una etiqueta de fin es escribir la de inicio y la de fin al mismo tiempo.

DOCUMENTOS BIEN FORMADOS

Con el fin de que las aplicaciones informáticas sean capaces de interpretar las marcas que identifican a cada elemento y deducir la estructura jerárquica del documento XML para su posterior procesamiento, es necesario que las marcas estén anidadas correctamente.

Para que las marcas estén anidadas de la forma adecuada, se deben cumplir las siguientes restricciones:

1. La marca de fin de un elemento A no puede escribirse hasta que no se hayan introducido todas las etiquetas de cierre de los elementos cuya marca de inicio esté situada después de la marca de apertura del elemento A".
2. No se puede introducir una marca de cierre de un elemento si no se ha introducido antes su marca de inicio.
3. Para todos los elementos no vacíos del documento se debe incluir una marca de inicio y una de fin.

REGLAS DE FORMACIÓN

Los elementos al igual que todos los componentes de un documento XML deben seguir ciertas reglas de buena formación, aplicables tanto a las etiquetas de inicio y fin como al contenido que admiten. En este apartado se recogen las condiciones que deben cumplir los elementos para considerarse bien formados Y aunque recoge algunas ideas ya vistas también se añaden otras nuevas.

Un procesador XML conforme con la especificación XML, comprobará que el documento cumple con estas restricciones de buena formación. Cualquier restricción no cumplida será detectada y se tratará como un error fatal; el procesador informará a la aplicación y dejará de trabajar de una manera normal. Fundamentalmente, el objetivo de estas restricciones es asegurar que los documentos XML puedan ser interpretados por los procesadores XML sin ambigüedad, de manera que todos los elementos (y atributos) quedan perfectamente definidos.

En los siguientes puntos se mostrarán algunas maneras de comprobar mediante programas si un documento en concreto está bien formado.

NOMBRES

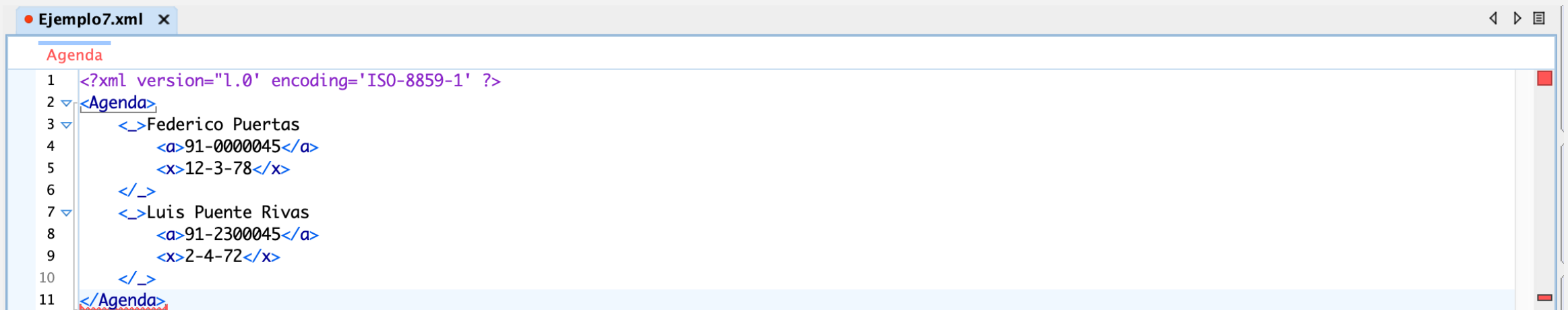
Una de las primeras tareas que se plantea al utilizar los elementos es la necesidad de asignarles un nombre. Los nombres deben cumplir ciertas restricciones para ser aceptados por el procesador XML.

Las reglas de formación de los nombres (reglas 4 y 5 de la Recomendación) definen la forma general que debe tener un nombre y básicamente, estas reglas de producción dicen que:

- Un nombre (Name) se compone como mínimo de una letra, (desde la "a" a la "z" o desde la "A" a la "Z") de un guion bajo o de un carácter de dos puntos.
- El carácter inicial (letra, guion bajo o carácter de dos puntos) puede estar seguido de otros caracteres, definidos dentro del grupo "NameChar" (regla 4), es decir, una o más letras, dígitos, puntos, guiones, guiones bajos, caracteres de dos puntos, carácter "CombiningChar" y caracteres "Extender" en cualquier combinación

El resto de los caracteres, destacando los espacios en blanco y los siguientes caracteres ",", "!" o ";", no pueden formar parte de un nombre, por ejemplo, "<nombredeelemento con espacios>" sería interpretado como tres nombres. Fíjese además que un nombre no puede comenzar ni por un dígito, ni por un punto, ni por un guion. Para ilustrar estas reglas se muestra un ejemplo.

En el documento XML los nombres son mínimos, es decir, están formados por un único carácter.



```
1 <?xml version="1.0" encoding='ISO-8859-1' ?>
2 <Agenda>
3   <>Federico Puertas
4     <a>91-0000045</a>
5     <x>12-3-78</x>
6   </>
7   <>Luis Puente Rivas
8     <a>91-2300045</a>
9     <x>2-4-72</x>
10  </>
11 </Agenda>
```


ETIQUETAS DE INICIO, DE FIN Y DE ELEMENTO VACÍO

Al comenzar el capítulo se describían los elementos como un conjunto de tres piezas, una etiqueta de inicio, un contenido y una etiqueta de fin. En esta sección del capítulo revisaremos estos conceptos para formalizar algunos detalles, Y veremos como existen también elementos sin contenido, que pueden reducirse a una única etiqueta de inicio y fin.

- Etiqueta de inicio: las etiquetas de inicio comienzan con un carácter de menor que, "<", y a continuación, sin ninguna separación, un identificador genérico (un nombre XML). A lo anterior, le pueden seguir espacios en blanco completamente opcionales. Las etiquetas finalizan con un carácter de mayor que, ">". Todas las etiquetas de inicio deben tener su correspondiente etiqueta de fin con el mismo nombre XML.
- Etiqueta de fin: las etiquetas de fin comienzan con la cadena "< /", seguida, sin separación, de un identificador genérico (un nombre XML) seguido a su vez de un carácter de mayor que, ">". Entre el nombre del elemento y el carácter ">" puede, opcionalmente, haber espacios en blanco. Para que una etiqueta de fin tenga sentido debe haber una etiqueta de inicio anterior, con el mismo identificador genérico (incluidas las mayúsculas y las minúsculas).

Los elementos sin contenido reciben el nombre de elementos vacíos, estos elementos pueden aparecer expresados de dos maneras, mediante dos etiquetas, una de inicio y otra de fin, seguidas, o mediante una única etiqueta especial, la etiqueta de elemento vacío.

- Etiqueta de elemento vacío: está formada por el carácter "<", seguido de un identificador genérico (un nombre XML). Para finalizar espacios en blanco opcionales, y la cadena ">".

Dos etiquetas de inicio y de fin seguidas, sin contenido en medio, pueden indicar que se trata de un elemento vacío por naturaleza, o que un elemento, por naturaleza con contenido, está vacío (temporalmente o no).

Para mostrar un ejemplo de elemento vacío podemos añadirle al romance anterior un elemento llamado Datos.

```
Ejemplo8.xml x
Poema
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <Poema>
3   <Titulo>El Reino Perdido</Titulo>
4   <Datos/>
5   <Cuerpo>
6     Las huestes de don Rodrigo
7     desmayan y huían
8     cuando en la octava batalla
9     sus enemigos vencían
10  </Cuerpo>
11 </Poema>
```

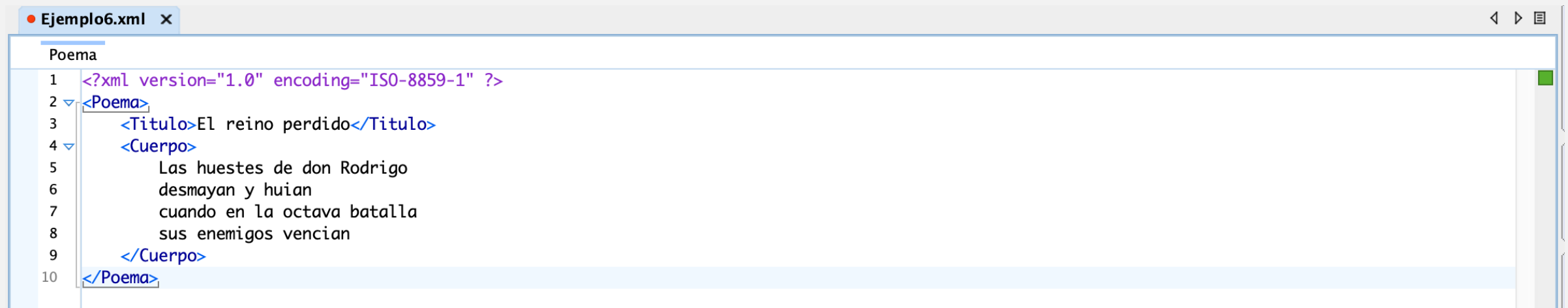
Tal y como está ahora este nuevo elemento no aporta información, entonces, ¿cuál es el sentido de los elementos vacíos? Los elementos vacíos normalmente se complementan con atributos. Los atributos se explicarán más adelante en el capítulo.

El elemento Datos también podría haberse escrito de la siguiente manera:

```
14 <Datos></Datos>
```

EL ELEMENTO RAÍZ

Para entender lo que es un elemento raíz volvamos al ejemplo de partida.

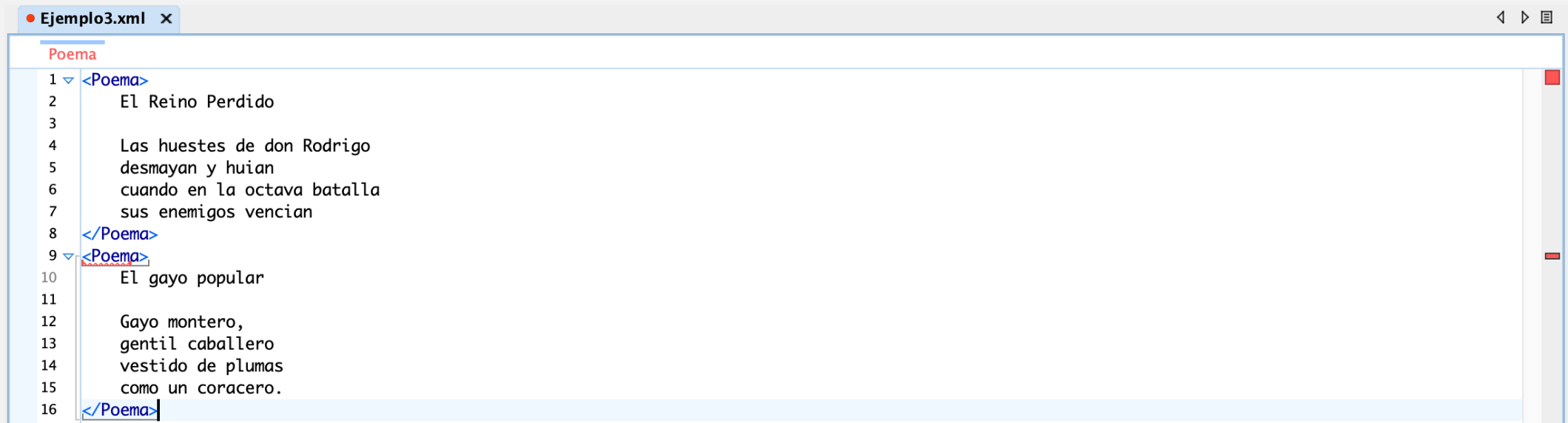


```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <Poema>
3   <Titulo>El reino perdido</Titulo>
4   <Cuerpo>
5     Las huestes de don Rodrigo
6     desmayan y huian
7     cuando en la octava batalla
8     sus enemigos vencian
9   </Cuerpo>
10 </Poema>
```

Podemos observar en el ejemplo que "Poema" es un elemento especial, ya que, agrupa en su interior al resto de los elementos y datos carácter y, en definitiva, recoge todo el contenido del ejemplar documento, separándolo de otras partes (si las hubiera). Por todo ello, "Poema" recibe el apelativo de "raíz", con otras palabras. Poema es el elemento raíz del documento. El elemento raíz también recibe el nombre de elemento documento porque recoge todo el contenido del documento bajo un mismo elemento. mismo elemento.

Todos los documentos XML tienen obligatoriamente un elemento raíz (uno y sólo uno), un documento XML sin el elemento raíz o con dos o más elementos raíz no está bien formado.

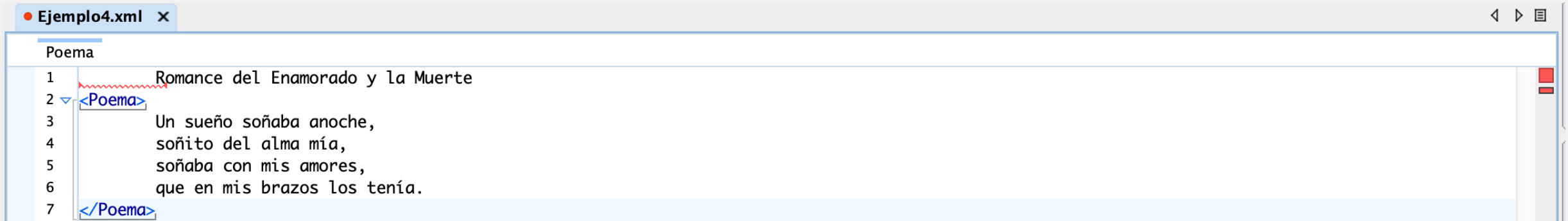
Los siguientes dos ejemplos muestran algunos errores de formación, cada uno de estos listados se almacena en un archivo.



```
1 <Poema>
2   El Reino Perdido
3
4   Las huestes de don Rodrigo
5   desmayan y huían
6   cuando en la octava batalla
7   sus enemigos vencían
8 </Poema>
9 <Poema>
10  El gajo popular
11
12  Gajo montero,
13  gentil caballero
14  vestido de plumas
15  como un coracero.
16 </Poema>
```

Este documento no está bien formado, existen dos presuntos elementos raíz. Para solucionarlo habría que crear dos documentos XML o bien englobar los dos elementos "Poema" bajo otro elemento común que se denominará, por ejemplo, "Poemas".

¿Qué le ocurre al siguiente intento de documento XML?



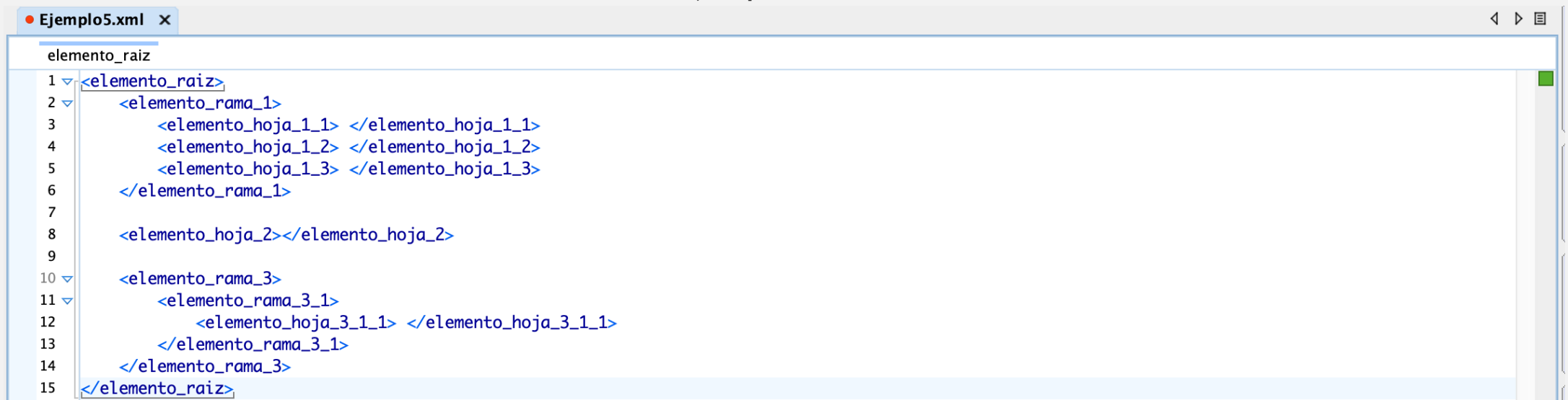
```
1 Romance del Enamorado y la Muerte
2 <Poema>
3   Un sueño soñaba anoche,
4   soñito del alma mía,
5   soñaba con mis amores,
6   que en mis brazos los tenía.
7 </Poema>
```

Alguien podría crear el documento anterior y pretender utilizarlo como XML, pero ¿existe en este texto algún elemento que englobe a todo lo demás? La pregunta es bien sencilla de responder, no, la cadena "Romance del Enamorado y la Muerte" no está contenida dentro de ningún elemento.

En los documentos HTML el elemento raíz es `<HTML>`. Sin embargo, los documentos XML no pueden utilizar ningún elemento con etiqueta `<XML>` (incluidas todas sus posibles variantes cambiando las mayúsculas por minúsculas), ya que, la cadena XML está reservada. Es decir, ni el elemento raíz, ni ningún otro puede llamarse XML. A este respecto hay que añadir que, si en general los nombres de los elementos deben ser representativos de lo que se puede encontrar en su interior, esta afirmación cobra una especial importancia con este elemento ya que su nombre debería resumir la esencia del contenido del documento.

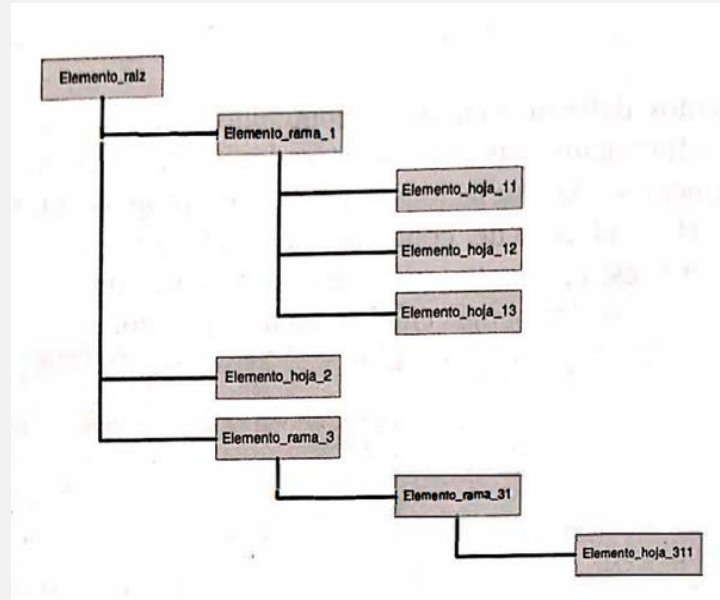
JERARQUÍA DE LOS ELEMENTOS

- Como se decía al introducir el concepto de elementos, éstos pueden anidarse unos con otros apropiadamente (sin solaparse) y entremezclarse con datos carácter, para representar la estructura lógica del documento y la semántica de sus diferentes contenidos. Esta combinación de elementos forma una jerarquía en árbol que parte de un único elemento: la raíz. Observemos este ejemplo:

A screenshot of an XML editor window titled 'Ejemplo5.xml'. The editor displays an XML document with a root element 'elemento_raiz'. The document structure is as follows: Line 1: <elemento_raiz> Line 2: <elemento_rama_1> Line 3: <elemento_hoja_1_1> </elemento_hoja_1_1> Line 4: <elemento_hoja_1_2> </elemento_hoja_1_2> Line 5: <elemento_hoja_1_3> </elemento_hoja_1_3> Line 6: </elemento_rama_1> Line 7: Line 8: <elemento_hoja_2></elemento_hoja_2> Line 9: Line 10: <elemento_rama_3> Line 11: <elemento_rama_3_1> Line 12: <elemento_hoja_3_1_1> </elemento_hoja_3_1_1> Line 13: </elemento_rama_3_1> Line 14: </elemento_rama_3> Line 15: </elemento_raiz> The editor has a line number margin on the left and a tree view on the right showing the hierarchy: elemento_raiz (expanded) -> elemento_rama_1 (expanded) -> elemento_hoja_1_1, elemento_hoja_1_2, elemento_hoja_1_3 -> elemento_hoja_2 -> elemento_rama_3 (expanded) -> elemento_rama_3_1 (expanded) -> elemento_hoja_3_1_1. The root element 'elemento_raiz' is highlighted in blue at the bottom of the list.

```
elemento_raiz
1 <elemento_raiz>
2   <elemento_rama_1>
3     <elemento_hoja_1_1> </elemento_hoja_1_1>
4     <elemento_hoja_1_2> </elemento_hoja_1_2>
5     <elemento_hoja_1_3> </elemento_hoja_1_3>
6   </elemento_rama_1>
7
8   <elemento_hoja_2></elemento_hoja_2>
9
10  <elemento_rama_3>
11    <elemento_rama_3_1>
12      <elemento_hoja_3_1_1> </elemento_hoja_3_1_1>
13    </elemento_rama_3_1>
14  </elemento_rama_3>
15 </elemento_raiz>
```

Podemos comprobar como existe una estructura subyacente en árbol, cuya representación gráfica podría ser la que se muestra:



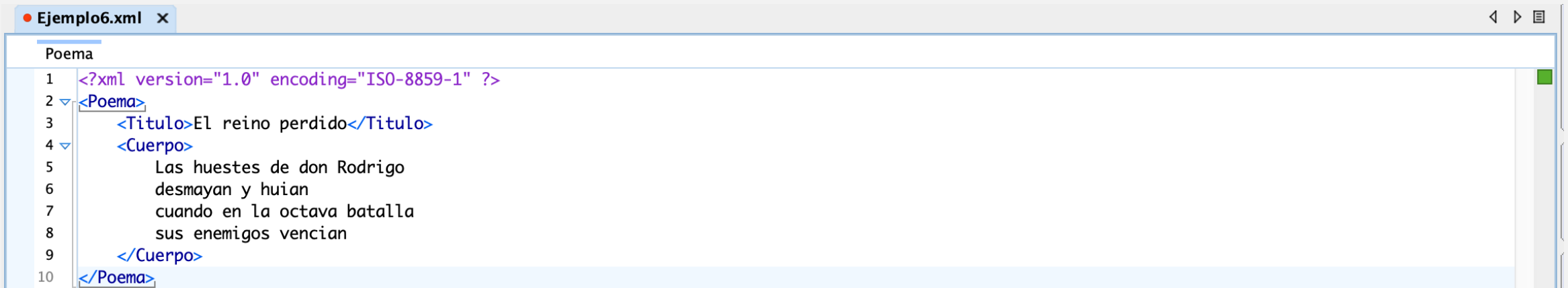
Los elementos terminales en la jerarquía, aquellos que no contienen ningún otro elemento dentro de ellos, reciben el nombre de hojas y los intermedios son las ramas. Ahora es fácil entender el porqué del nombre "elemento raíz".

Para describir las relaciones entre elementos se puede hablar de elementos que son padres ("parent" en inglés), hermanos (siblings) e hijos (children) de otros. Un elemento es padre de todos los elementos que incluye entre sus etiquetas de inicio y fin. Los elementos que comparten al mismo padre se denominan hermanos. Así, en el ejemplo anterior, el elemento raíz es el padre de todos los demás elementos que son hermanos entre sí. Los elementos "elemento_hoja_1_1", "elemento_hoja_1_2" y "elemento_hoja_1_3" son los hijos de "elemento_rama_1". Es fácil.

La disposición de los elementos en un documento XML se denomina estructura lógica. A la hora de realizar un documento XML o a la hora de comprenderlo puede ser útil plasmar su estructura lógica en forma de árbol.

LA DECLARACIÓN XML

Ya sabemos que los elementos únicamente se pueden emplear dentro del ejemplar del documento para complementar el texto original, (no tienen sentido en ningún otro lugar), y aunque es perfectamente posible trabajar solamente con el ejemplar, es recomendable incluir también una declaración XML, que defina nuestro documento como XML, que concrete el número de versión (lo que posiblemente evitará problemas en el futuro) y que nos permita cambiar la codificación por defecto para incluir cómodamente acentos en los textos.



```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <Poema>
3   <Titulo>El reino perdido</Titulo>
4   <Cuerpo>
5     Las huestes de don Rodrigo
6     desmayan y huían
7     cuando en la octava batalla
8     sus enemigos vencían
9   </Cuerpo>
10 </Poema>
```


CARACTERES PROHIBIDOS DENTRO DEL CONTENIDO DE LOS ELEMENTOS

Dentro del contenido de los elementos los siguientes caracteres y cadenas tienen funciones especiales y si aparecen deben escaparse para que no se interpreten erróneamente:

- <: este carácter podría confundirse con el comienzo de una etiqueta, para usarse en otro contexto debe escaparse.
- &: este carácter debe escaparse si no se utiliza como comienzo de una referencia a entidad (todavía no se ha hablado de ellas).
-]]>: esta cadena no debe aparecer dentro del contenido de un elemento. Esto es así para asegurar cierta compatibilidad hacia atrás con SGML.

Ya se comentó que la mejor manera para escapar caracteres es utilizar referencias a entidades predefinidas.

RESTRICCIONES EN CUANTO AL ANIDAMIENTO DE LOS ELEMENTOS

Este apartado resume o reúne algunos conceptos que ya deberían haber quedado claros.

- Debe existir un elemento raíz, y sólo uno. Todos los documentos XML tienen como mínimo un elemento, la raíz, aunque lo normal es que existan más elementos, incluidos dentro de él.
- Todos los elementos tienen una etiqueta de inicio y finalización, aunque los elementos vacíos podrán tener una única etiqueta, de inicio y finalización. Las letras mayúsculas/minúsculas deben coincidir en las etiquetas de inicio y fin del mismo elemento, en otro caso se consideran pertenecientes a elementos distintos.
- Los elementos deben anidarse correctamente. Es decir, si la etiqueta de inicio de un elemento está dentro de otro elemento, la etiqueta de finalización del primer elemento debe estar dentro de la etiqueta de finalización del segundo elemento.
- Siempre que estas reglas de anidamiento se cumplan, y a falta de una declaración que especifique el contenido del elemento (se estudiarán más adelante), los elementos admiten cualquier combinación de elementos y datos carácter.

DEFINICIÓN DEL TIPO DE DOCUMENTO

DTD y Documentos

Las definiciones de tipo de documento (DTD, Document Type Definition) son los medios con que podemos restringir la estructura de una clase de documentos XML. Las DTD especifican los componentes que están disponibles para un determinado tipo de documento (por ejemplo, un documento catálogo tiene artículos, precios, descripciones y otros) y la forma en que esos componentes se pueden mezclar para producir una instancia válida (por ejemplo, cada artículo sólo tiene una descripción y un precio; pueden existir muchos artículos en un catálogo).

Una analogía natural para la relación entre DTD y documento es la relación entre clase y objeto. Una clase define los tipos y nombres de cierto grupo de objetos. Los objetos cuentan con todo tipo de información, pero todos cumplen con las reglas establecidas por la clase. La especificación XML 1.0 proporciona la sintaxis para la creación de DTD y las reglas que rigen el comportamiento de los analizadores sintácticos XML que validan los documentos XML.

DTD: DEFINICIÓN Y FUNCIONALIDAD

El conjunto de marcas debe escribirse siguiendo unas reglas. Algunas de estas reglas son aplicables a todos los documentos XML, e indican la forma en la que deben incluirse las marcas. El cumplimiento de estas reglas hace que un documento XML esté bien formado.

Otras reglas solo son aplicables a los documentos de un tipo específico: por ejemplo, un capítulo de un manual técnico, un recibo de compra o un registro bibliográfico. Si el documento XML cumple las reglas indicadas para su tipo de documento, el documento será un documento válido. Las reglas específicas a cada tipo de documento se definen en una DTD (Document Type Definition) o definición de tipo de documento.

LA DTD CONTIENE INFORMACIÓN RELATIVA A:

- Los elementos que se pueden utilizar en un tipo de documento específico.
- Los elementos que son opcionales y los que son obligatorios.
- Los elementos que se pueden repetir en más de una ocasión y los que pueden aparecer solo una vez.
- El orden en el que se deben escribir los elementos y como pueden anidarse. Es decir, que elementos deben pueden contener a otros elementos.

En la DTD también se indica

- Los atributos válidos para cada elemento.
- El tipo de dato que puede recoger cada atributo.
- El carácter obligatorio u opcional de los atributos.
- Las entidades que se pueden referenciar en el documento, cómo se les debe hacer referencia y cómo deben ser resueltas. En el caso de las entidades, la DTD indicará:
 - Los recursos externos de tipo XML, también llamados procesables, que se pueden incluir en el documento.
 - Los recursos externos no XML (también llamados no procesables) que pueden aparecer en el documento: gráficos, multimedia, documentos HTML, etc. Para cada recurso externo no procesable que se declare en la DTD se tendrá que añadir una notación que indique a los programas como se deben tratar estos recursos.
 - Las cadenas de texto que se pueden usar como “comodín” o “abreviatura” en vez de palabras o frases de uso frecuente, difíciles de escribir o susceptibles de cambiar a lo largo del periodo de creación del documento.

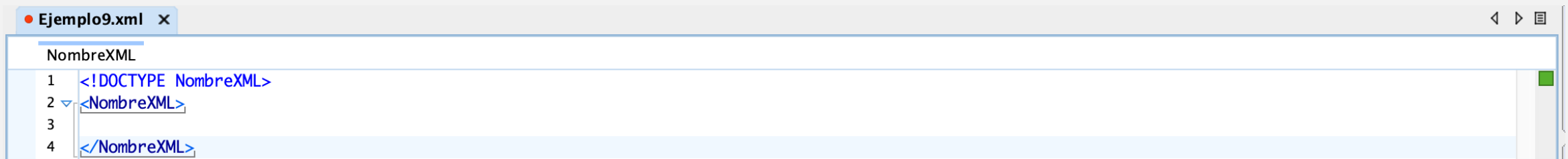
LA DECLARACIÓN DEL TIPO DEL DOCUMENTO

Todas las declaraciones de tipo comienzan con lo que es la declaración propiamente dicha, es decir, el texto concreto que especifica el nombre del tipo. ¡Para ello se emplea la cadena “<!DOCTYPE” seguida del nombre del tipo. El siguiente ejemplo muestra una declaración de tipo, y aunque está incompleta, podemos ver su estructura.

```
1 <!DOCTYPE NombreXML>
```

¡El nombre que se le asigne al tipo debe ser un nombre XML y debe estar separado de la cadena “<!DOCTYPE” por al menos un espacio en blanco, aunque pueden ser más.

Un tipo de documento no tendría sentido si no se pudiera asociar con sus documentos XML, este enlace se realiza en cada documento a través del elemento raíz, que debe tener el mismo nombre que el tipo, como muestran las siguientes líneas.



```
Ejemplo9.xml x
NombreXML
1 <!DOCTYPE NombreXML>
2 <NombreXML>
3
4 </NombreXML>
```

LA DEFINICIÓN DEL TIPO DEL DOCUMENTO

Toda declaración de tipo propiamente dicha para un documento se complementa con una definición del tipo (abreviada como DTD) que asocia al tipo las cualidades que posee. La DTD define los tipos de elementos, atributos, entidades y notaciones que se podrán utilizar en el documento, así como ciertas restricciones estructurales y de contenido, valores por defecto, etc. Para formalizar todo ello XML provee ciertas estructuras especiales, las llamadas declaraciones de marcado y que pertenecen a alguno de los siguientes tipos:

- Declaraciones de tipos de elementos.
- Declaraciones de listas de atributos para los tipos de elementos.
- Declaraciones de entidades.
- Declaraciones de notación.

El estudio de cada tipo de declaración se realizará, de momento y para ir tomando contacto con la DTD (la definición del tipo del documento), al documento anterior añadiremos algunas "declaraciones de tipos de elementos" que forman la "definición del tipo" y que dan sentido a la "declaración del tipo". No se preocupe por entender las declaraciones, únicamente comprenda la idea de fondo que quiere mostrar el ejemplo, y que se refiere a la capacidad de definir un tipo para un documento XML y a cómo el documento se debe ajustar a esa definición (si se desea que sea de tipo válido).


```
Ejemplo10.xml x
Casas_Rurales
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <!DOCTYPE Casas_Rurales[
3      <!ELEMENT Casas_Rurales (Casa)*>
4      <!ELEMENT Casa (Dirección, Descripción, Estado, Tamano)>
5      <!ELEMENT Dirección (#PCDATA)>
6      <!ELEMENT Descripción (#PCDATA)>
7      <!ELEMENT Estado (#PCDATA)>
8      <!ELEMENT Tamano (#PCDATA)>
9      ]>
10 <Casas_Rurales>
11
12 </Casas_Rurales>
```

La definición del tipo anterior especifica que el tipo de documento Casas_Rurales está formado por elementos de tipo Casa. Los elementos de tipo Casa contienen a su vez elementos de tipo Dirección, Descripción, Estado y Tamaño, en este orden y sin faltar ninguno. El contenido de estos elementos está formado exclusivamente por datos carácter (#PCDATA). El lector podría, a partir del ejemplo, pensar que todas las definiciones del tipo de un documento se parecen en mayor o menor medida a la anterior, la verdad es que admiten un gran número de posibilidades.

Para acabar de completar el documento XML del ejemplo, incluiremos el contenido del ejemplar del documento. Se puede comprobar cómo el ejemplar cumple con las restricciones de su tipo, su estructura y contenido son los especificados.

```
Ejemplo11.xml x
Casas_Rurales
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE Casas_Rurales[
3     <!ELEMENT Casas_Rurales (Casa)*>
4     <!ELEMENT Casa (Dirección, Descripción, Estado, Tamano)>
5     <!ELEMENT Dirección (#PCDATA)>
6     <!ELEMENT Descripción (#PCDATA)>
7     <!ELEMENT Estado (#PCDATA)>
8     <!ELEMENT Tamano (#PCDATA)>
9 ]>
10 <Casas_Rurales>
11 <Casa>
12     <Dirección>Calle las Palmas 23. La Sagra. Toledo</Dirección>
13     <Descripción>Se trata de una casa del siglo XVII</Descripción>
14     <Estado>El estado de conservación es magnifico</Estado>
15     <Tamano>La casa tiene 259 metros cuadrados</Tamano>
16 </Casa>
17 <Casa>
18     <Dirección>Calle Or 5, Fregenal de la sierra. Badajoz</Dirección>
19     <Descripción>De arquitectura espectacular la casa</Descripción>
20     <Estado>Recientemente restaurada. Su estado actual es bueno</Estado>
21     <Tamano>La superficie habitable es de 180 metros cuadrados</Tamano>
22 </Casa>
23 </Casas_Rurales>
```

Volvemos a recordar en este punto que las palabras clave "XML" como "ELEMENT" y "DOCTYPE" deben escribirse tal y como se muestran en los manuales, de otro modo generarán errores fatales, de buena formación.

Tras el nombre del tipo de documento, la declaración de tipo de documento puede contener:

- La definición del tipo de documento o DTD correspondiente a ese tipo de documento, o
- Una referencia a un recurso externo que contiene la DTD de ese tipo de documento.
- Una referencia a un recurso externo con la DTD, y la definición de parte de la DTD aplicable únicamente a ese documento.

En el primer caso hablaremos de DTD internas. En el segundo, de DTD externas. Y en el tercero, de DTD mixtas. Estas posibilidades se explican con mayor detalle en los siguientes apartados.

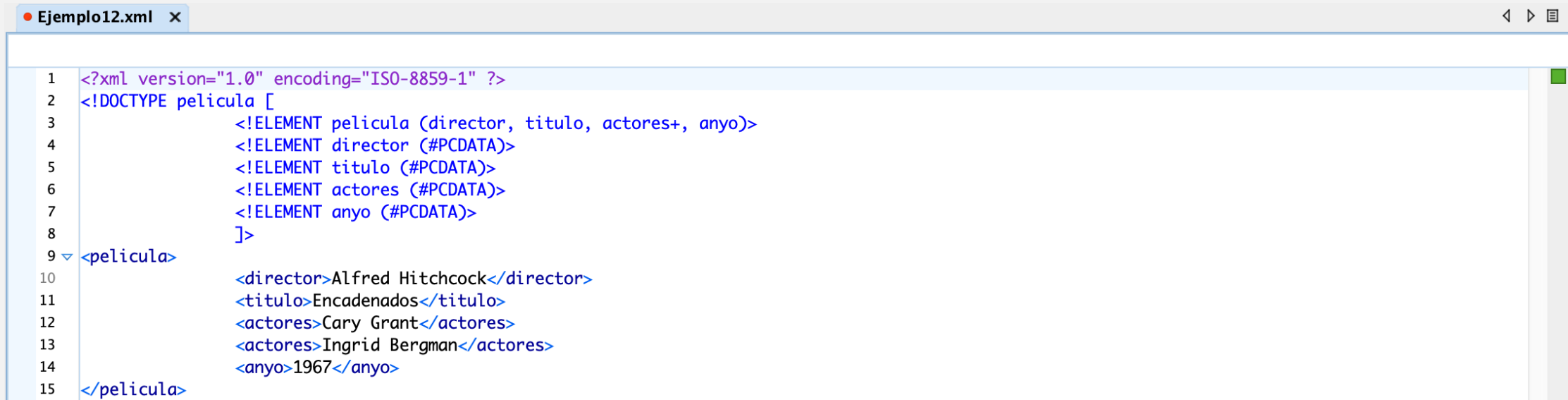
LAS DTD INTERNAS

La totalidad de la DTD se puede incluir en la declaración del tipo de documento incluida en el documento XML. En este caso diremos que la DTD es interna. Si se opta por incluir la DTD en la declaración de tipo de documento.

La DTD formada por una o más declaraciones- se escribe entre corchetes a continuación del nombre del tipo de documento, y antes del carácter > que marca el final de la declaración de tipo de documento.

En este momento no vamos a tratar la sintaxis que se debe utilizar para escribir las declaraciones de la DTD.

El siguiente documento contiene una DTD interna en la que se declaran los cinco elementos que se pueden utilizar en el documento XML.



The screenshot shows a code editor window titled 'Ejemplo12.xml'. The code defines an internal DTD for a document type named 'pelicula'. The DTD declarations are as follows:

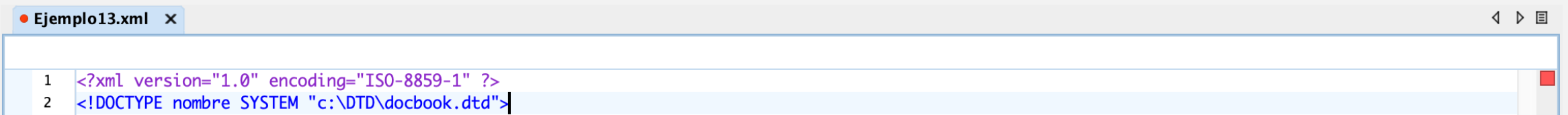
```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE pelicula [
3     <!ELEMENT pelicula (director, titulo, actores+, anyo)>
4     <!ELEMENT director (#PCDATA)>
5     <!ELEMENT titulo (#PCDATA)>
6     <!ELEMENT actores (#PCDATA)>
7     <!ELEMENT anyo (#PCDATA)>
8 ]>
9 <pelicula>
10     <director>Alfred Hitchcock</director>
11     <titulo>Encadenados</titulo>
12     <actores>Cary Grant</actores>
13     <actores>Ingrid Bergman</actores>
14     <anyo>1967</anyo>
15 </pelicula>
```

DTD EXTERNAS

La declaración de tipo de documento puede asociar un documento XML con una DTD que se ha escrito en un archivo diferente. En este caso diremos que la DTD es una DTD externa. Las DTD externas se pueden incluir de dos formas distintas en la declaración de tipo de documento, dependiendo de si la DTD es una DTD privada (disponible en un archivo local o remoto), o una DTD "pública".

DTD PRIVADAS

El término DTD privada se utiliza cuando la DTD se encuentra en un archivo de un directorio físico local, de una red corporativa o en un servidor remoto accesible mediante el protocolo HTTP. Para indicar que un tipo de documento está asociado a una DTD externa, se utiliza la siguiente sintaxis:

A screenshot of an XML editor window titled 'Ejemplo13.xml'. The editor shows two lines of XML code. Line 1 is the XML declaration: `<?xml version="1.0" encoding="ISO-8859-1" ?>`. Line 2 is the DTD declaration: `<!DOCTYPE nombre SYSTEM "c:\DTD\docbook.dtd">`. The text 'nombre' is highlighted in blue, and the text 'SYSTEM' is highlighted in purple. The file path 'c:\DTD\docbook.dtd' is enclosed in double quotes.

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE nombre SYSTEM "c:\DTD\docbook.dtd">
```

En la declaración de tipo de documento con una DTD externa privada, tras el nombre del tipo de documento, se escribe la palabra reservada **SYSTEM**, y a continuación se escribe entre comillas dobles la ruta absoluta o relativa del archivo que contiene las declaraciones de la DTD.

Por convención, el archivo que contiene la DTD suele tener la extensión **dtd** y como nombre el nombre del tipo de documento (que a su vez coincidirá con el del elemento documento o raíz). Sin embargo, esto no es obligatorio y el archivo con la DTD podría tener un nombre que no coincidiese con el del tipo de documento, o una extensión distinta a **DTD**.

Los siguientes casos nos muestran algunos ejemplos de declaraciones de tipos de documentos los cuales apuntan a una DTD situada en un archivo físico aparte.

- En este caso, la DTD está en el subdirectorio C:\XML\DTD\pelicula.dtd.

```
Ejemplo14.xml x
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE pelicula SYSTEM "c:\xml\DTD\pelicula.dtd">
3 <pelicula>
4     <director>Alfred Hitchcock</director>
5     <titulo>Encaderrados</titulo>
6     <reparto>
7         <actor>Cary Grant</actor>
8         <actor>Ingrid Bergman</actor>
9     </reparto>
10    <anyo>1966</anyo>
11 </pelicula>
```

En este segundo caso, la DTD se encuentra en un directorio local al que se hace referencia con una ruta relativa:

```
Ejemplo15.xml x
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE pelicula SYSTEM "..\DTD\pelicula.dtd">
3 <pelicula>
4     <director>Alfred Hitchcock</director>
5     <titulo>Encaderrados</titulo>
6     <reparto>
7         <actor>Cary Grant</actor>
8         <actor>Ingrid Bergman</actor>
9     </reparto>
10    <anyo>1966</anyo>
11 </pelicula>
```

En el caso que vemos a continuación la DTD se encuentra situada en un URL remoto:

Ejemplo16.xml x

1234567891011

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE pelicula SYSTEM "http://www.server.com/DTD/pelicula.dtd">
<pelicula>
  <director>Alfred Hitchcock</director>
  <titulo>Encaderrados</titulo>
  <reparto>
    <actor>Cary Grant</actor>
    <actor>Ingrid Bergman</actor>
  </reparto>
  <anyo>1966</anyo>
</pelicula>
```

DTD PÚBLICAS

El término DTD pública se utiliza para hacer referencia a las DTD que están ampliamente difundidas y que son utilizadas por distintas comunidades de usuarios.

Podemos decir que se trata de DTD de "dominio público", como por ejemplo las DTD publicadas por instituciones o asociaciones con capacidad de definir estándares y normas para el intercambio de datos. Así, podemos considerar como DTD públicas las DTD DocBook, MARTIF (Machine Readable Terminology Interchange Format), MTX (Machine Translation Exchange), etc.

En la declaración de un tipo de documento se puede indicar que el tipo de documento se debe validar contra una DTD pública. Para ello se escribirá, tras el nombre del tipo de documento, la palabra reservada PUBLIC. A continuación de la palabra reservada PUBLIC se escribirá, entre comillas dobles, un identificador para la DTD pública. Las DTD públicas tienen un identificador único que tiene la siguiente forma:

1	Tipo identificador//organización//DTD descripción//Idioma
---	---

Esta cadena está formada por cuatro partes separadas por dos barras invertidas. Cada parte se debe interpretar de la forma indicada en la tabla:

Sub cadena de texto	Descripción
Tipo de identificador	Indica si el identificador público de la DTD ha sido registrado o no internacionalmente, de acuerdo con lo indicado en la norma ISO 9070. Si el identificador está registrado, se escribirá el signo +. Si no está registrado, el signo -. En el caso de que la DTD haya sido registrada por la ISO (International Standard Organization) como norma internacional, se escribirá la cadena ISO 8879: 1986
Organización	Nombre de la organización o persona propietaria de la DTD. Si el dueño de la DTD es la ISO no se indicará nada en este apartado
DTD	Palabra reservada para indicar que el recurso público es una DTD
Descripción	Texto descriptivo para la DTD
Lenguaje	Idioma en el que se ha escrito la DTD. El idioma se indica utilizando las convenciones indicadas en la norma ISO 639

En la declaración de una DTD pública se utilizan los caracteres / / para separar el tipo de identificador del propietario, éste de la palabra reservada DTD y el texto descriptivo del identificador de idioma.

La siguiente declaración de tipo de documento se interpretaría de la siguiente forma:

1

<!DOCTYPE docbook PUBLIC "-//oasis/DTD articles to publish//EN">

El tipo de documento docbook se define en una DTD pública no registrada diseñada por una organización llamada oasis. La DTD se ha escrito en inglés y se utiliza para codificar artículos.

La utilización de DTD públicas nos conduce a la siguiente pregunta ¿cómo conoce una aplicación las características de una DTD pública? ¿en qué archivo o sede remota se encuentran las declaraciones que marcan las normas que deben seguir los documentos de este tipo?

Las aplicaciones que soportan SGML y XML incorporan un archivo de texto referido comúnmente como catálogo, en el que se indica, para cada identificador de un recurso público, la ruta física en la que se encuentra un archivo con todas sus declaraciones.

Dependiendo de la sofisticación del programa, el archivo físico correspondiente puede encontrarse en un directorio local o bien apuntar a un servidor web remoto.

Cuando la aplicación intente procesar un documento XML que incluya una declaración de tipo de documento que apunte a una DTD pública, el programa recuperará el archivo con la DTD correspondiente al tipo de documento a partir de la información indicada en el catálogo.

DTD MIXTAS: SUBCONJUNTO INTERNO Y EXTERNO

Las declaraciones que forman la DTD utilizada por un documento pueden escribirse dentro de la declaración de tipo de documento o en un archivo aparte al que se hace referencia desde esta declaración. XML nos ofrece una tercera posibilidad, mediante la cual es posible distribuir las declaraciones de la DTD en la declaración de tipo de documento y en un archivo externo. De esta forma la DTD se encuentra en dos archivos físicos diferentes.

Se utiliza el termino subconjunto interno de la DTD para referirnos a las declaraciones que se incluyen dentro de la declaración de tipo de documento.

El termino subconjunto externo de la DTD se refiere a las declaraciones que se incluyen en un archivo independiente en el que solo se incluyen declaraciones de la DTD.

El siguiente documento XML contiene una declaración de tipo de tipo documento con una DTD mixta. El subconjunto externo de la DTD se ha definido en el archivo pelicula.dtd. El subconjunto interno consiste en las declaraciones que se han indicado entre corchetes en la declaración de tipo de documento.

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE pelicula SYSTEM "http://www.server.com/DTD/pelicula.dtd"[
3 <!ENTITY cartel SYSTEM "c:\img\encadenados.gif">
4 <!ENTITY ibergman SYSTEM "c:\img\ibergman.gif">
5 ]>
6 <pelicula>
7     <director>Alfred Hitchcock</director>
8     <titulo>Encadenados</titulo>
9     <reparto>
10         <actor>Cary Grant</actor>
11         <actor>Ingrid Bergman</actor>
12     </reparto>
13     <anyo>1996</anyo>
14 </pelicula>
```

LA UTILIZACIÓN DE DTD MIXTAS PERMITE:

- La reutilización de las declaraciones genéricas de la DTD en múltiples documentos.
- Un mantenimiento más fácil de la DTD, al poder realizar cambios que se aplicarán automáticamente en todos los documentos que la utilicen.
- Aislar las características propias de cada documento (es decir, las que se aplican a un documento en concreto, pero no a todos los documentos de la misma clase). De esta forma no se ampliarán las declaraciones de la DTD con declaraciones innecesarias en la mayoría de los casos.
- Normalmente, en el subconjunto interno de la DTD se declaran las entidades que se utilizarán únicamente en documentos particulares. Las entidades, de los documentos XML que se utilizan con distintos fines, entre ellos la inclusión de archivos binarios (imágenes, gráficos, etc.) en los documentos XML. Las entidades se deben declarar en la DTD.
- Si un gráfico aparece únicamente en un documento, no es necesario declarar su entidad correspondiente en el subconjunto externo de la DTD, ya que es una entidad que no se va a utilizar en todos los documentos. Sin embargo, si se quiere incluir en todos los documentos de un mismo tipo un logotipo o imagen que los caracterice.

DECLARACIONES DE LA DTD

La DTD está formada por una o más declaraciones. Las declaraciones se escriben utilizando una sintaxis especial. Una DTD puede contener declaraciones de los diferentes tipos que aparecen a continuación:

- Declaraciones de elementos.
- Declaraciones de listas de atributos.
- Declaraciones de entidades.
- Declaraciones de notaciones.
-

¡Cada declaración se escribe entre los caracteres reservados `< ! Y >`. ¡Tras los caracteres de inicio `< !` se escribe el tipo de la declaración, utilizando una de las siguientes palabras reservadas:

- **ELEMENT**: Para declarar elementos.
- **ATTLIST**: Para declarar los atributos válidos para un elemento.
- **ENTITY**: Para declarar una entidad y
- **NOTATION**: Para declarar una notación.

DECLARACIONES DE TIPOS DE ELEMENTOS.

Los documentos, o instancias, están integrados principalmente por elementos; por ello, la primera característica de las DTD que debemos explorar son las declaraciones de tipo de elemento. Las “Declaraciones de tipos de elementos”, como dice la propia expresión permiten definir tipos de elementos. Una declaración de elemento especifica el tipo de contenido que debe tener un elemento determinado.

A través de ellas podemos comunicar a un procesador de XML, que valide el documento, qué elementos están permitidos y cual es su contenido. Con otras palabras, en lo que se respecta a los elementos, para que un documento XML sea de tipo valido debe cumplir que:

- Todos los elementos estén reconocidos mediante “declaraciones de tipos”, o lo que es lo mismo, todos los elementos deben pertenecer a un tipo declarado.
- El contenido de cada elemento se ajusta a lo declarado en su tipo.

Así, las declaraciones de tipos de elementos permitirán la detección de ciertos errores: inclusión de elementos no declarados, contenidos no validos en un elemento, elementos obligatorios olvidados, elementos rápidos, etc.

Un elemento puede contener datos de tipo carácter (o texto propiamente dicho), otros elementos o ambos a la vez. En este último caso, diremos que el elemento tiene contenido mixto.

Cuando se declara un elemento en una DTD se debe indicar:

- El nombre del elemento
- El contenido que puede tener. Se utiliza el termino “declaración de contenido” para hacer referencia a esta parte de la declaración

¡La declaración de un elemento se encierra entre las marcas `<!Y >`. ¡Tras los caracteres de inicio `<!` Se escribe la palabra reservada **ELEMENT** (para indicar que se está declarando un elemento), seguida del nombre que se quiera dar al elemento. Detrás del nombre del elemento se escribirá, entre paréntesis, su declaración de contenido.

NOMBRES DE ELEMENTOS

En un nombre de elemento se pueden utilizar los caracteres A-Z, a-z, 0-9, y los signos de puntuación guion (-), punto(.), dos puntos (:) y subrayado (_). Los nombres de elementos deben comenzar con una letra (mayúscula o minúscula), dos puntos (:) o el carácter subrayado (_). No pueden comenzar con un número, un punto(.) o un guion (-).

Estas restricciones también se aplican a los nombres de atributos.

DECLARACIONES DE CONTENIDO

La declaración de contenido puede incluir:

- La palabra reservada `#PCDATA`, que indica que el elemento puede contener datos de tipo carácter, es decir, no marcas. En documentos XML basados en DTD, el contenido `PCDATA` se refiere tanto a texto como a números o fechas.
- El nombre de otros elementos que se puede incluir entre las etiquetas de inicio y de fin del elemento que se declara.
- La palabra reservada `#PCDATA` y el nombre de otros elementos. En este caso se habla de un modelo de contenido mixto.
- La palabra reservada `EMPTY`, para indicar que se trata de un elemento vacío o
- La palabra reservada `ANY`, para indicar que no se aplica ninguna restricción sobre el contenido del elemento.

En el caso de que un elemento contenga a otros elementos, estos elementos también tendrán que declararse en la DTD, aunque sus declaraciones no tienen que preceder obligatoriamente a la declaración del elemento desde el que se les hace referencia.

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE datos_libro [
3     <!ELEMENT bib-ref (autor, titulo, pie-editorial)>
4     <!ELEMENT autor (#PCDATA)>
5     <!ELEMENT titulo (#PCDATA)>
6     <!ELEMENT pie-editorial (lugar-edicion, editor, fecha-edicion)>
7     <!ELEMENT lugar-edicion (#PCDATA)>
8     <!ELEMENT editor (#PCDATA)>
9     <!ELEMENT fecha-edicion (#PCDATA)>
```

En este ejemplo se muestra una DTD simple formada por siete declaraciones de elementos. Como se puede apreciar, en las declaraciones de contenido de los elementos bib-ref y pie-editorial se incluyen referencias a otros elementos.

Todos los elementos referenciados se declaran en la DTD, aunque se pueden declarar posteriormente a la declaración de contenido en la que aparecen. Por ejemplo, las declaraciones de los elementos autor, titulo o pie-editorial se han escrito tras la declaración del elemento bib-ref, en cuya declaración de contenido se les está haciendo referencia. Lo mismo sucede con la declaración del elemento pie-editorial.

MODELO DE CONTENIDO PCDATA

En las declaraciones de contenido se puede incluir también la palabra reservada `#PCDATA`. Esta palabra corresponde a `parsed character data`, que podemos traducir como datos de tipo carácter analizados. Con esta declaración de contenido se indica que el elemento puede contener cualquier tipo de texto que no sean los caracteres reservados en XML para indicar la presencia de marcas. Dicho de otra forma, un elemento cuya declaración de contenido contenga únicamente la palabra `#PCDATA` podrá incluir cualquier carácter excepto `<`, `>`, `'`, `"` o `&`.

Si se necesita introducir estos caracteres dentro del contenido del elemento sin que sean interpretados como señal del inicio o fin de una marca o de una referencia a una entidad, se pueden utilizar dos alternativas:

- Emplear las entidades predefinidas `<`, `>`, `"`, `'`, ó `&` en lugar de `<`, `>`, `'`, `"` o `&` respectivamente.
- Utilizar una sección `CDATA`.

Los elementos con contenido `#PCDATA` pueden contener referencias a entidades procesables externas y a entidades de texto internas. Recordamos que las entidades procesables son datos XML que pueden contener a su vez otras marcas.

MODELO DE CONTENIDO CON OTROS ELEMENTOS.

Un elemento puede contener a otros elementos declarados en la DTD. Diremos que los elementos contenidos están anidados dentro del elemento que los contiene.

En la declaración de contenido de un elemento se puede indicar qué elementos se pueden incluir dentro del elemento que se declara. También se debe especificar:

- Si estos elementos deben incluirse obligatoriamente o no.
- Si los elementos contenidos pueden aparecer en más de una ocasión.
- El orden en que los elementos deben aparecer en el elemento contenedor.

Estas restricciones se indican escribiendo un carácter especial tras el nombre del elemento que se incluye en la declaración de contenido. Los caracteres especiales son:

- ? – El elemento puede aparecer 0 o 1 vez (opcional no repetible).
- * – El elemento puede aparecer 0 o más veces (opcional y repetible).
- + – El elemento debe aparecer 1 o más veces (obligatorio y repetible).

Si no se indica ninguno de los caracteres anteriores tras el nombre del elemento, se está indicando que el elemento debe aparecer una sola vez. Es decir, el elemento será obligatorio y no repetible.

En la declaración de contenido también se puede indicar el orden en que deben aparecer los elementos contenidos.

Si los elementos no se separan mediante el carácter | (equivalente al operador lógico O exclusivo), se podrá escribir uno de ellos.

A continuación, se explica el significado de algunas declaraciones de elementos de los siguientes ejemplos.

`<!ELEMENT autor (nombre, apellido+)>`

Esta declaración de elemento declara un elemento llamado autor. En la declaración de contenido (escrita entre paréntesis tras el nombre del elemento), se indica:

- Que el elemento autor debe contener un elemento nombre seguido de al menos un elemento apellido. El elemento apellido debe aparecer obligatoriamente, y puede aparecer más de una vez. Es decir, es repetible y obligaría, tal y como indica el carácter+ escrito tras el.
- El elemento nombre es obligatorio y debe aparecer una sola vez ya que no se ha escrito seguido por ninguno de los calificadores.
- Los elementos nombre y apellido deben aparecer en este orden, ya que se han separado mediante comas.

En este contexto, las siguientes estructuras serían válidas y se podrían utilizar en un documento XML:

Caso 1:

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <!DOCTYPE datos_autor[
3      <!ELEMENT datos_autor (nombre, apellido)>
4      <!ELEMENT nombre (#PCDATA)>
5      <!ELEMENT apellido (#PCDATA)>
6      ]>
7  <datos_autor>
8      <nombre>Camilo J.</nombre>
9      <apellido>Cela</apellido>
10 </datos_autor>
```

Caso 2:

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <!DOCTYPE datos_autor[
3      <!ELEMENT datos_autor (nombre, apellido)>
4      <!ELEMENT nombre (#PCDATA)>
5      <!ELEMENT apellido (#PCDATA)>
6      ]>
7  <datos_autor>
8      <nombre>Camilo J.</nombre>
9      <apellido>Cela</apellido>
10     <apellido>Trullock</apellido>
11 </datos_autor>
```

Sin embargo, los siguientes documentos no serían válidos (aunque sí estarían bien formados);

Caso 3:

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <!DOCTYPE datos_autor[
3      <!ELEMENT datos_autor (nombre, apellido)>
4      <!ELEMENT nombre (#PCDATA)>
5      <!ELEMENT apellido (#PCDATA)>
6      ]>
7  <datos_autor>
8      <apellido>Cela</apellido>
9      <nombre>Camilo J.</nombre>
10 </datos_autor>
```

En este caso, no se respeta el orden en el que deben escribirse los elementos, y se ha escrito el elemento apellido antes del elemento nombre.

Caso 4:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE datos_autor[
3     <!ELEMENT datos_autor (nombre, apellido+)>
4     <!ELEMENT nombre (#PCDATA)>
5     <!ELEMENT apellido (#PCDATA)>
6 ]>
7 <datos_autor>
8     <apellido>Cela</apellido>
9 </datos_autor>
```

En este caso, se ha omitido el elemento nombre, que tiene carácter obligatorio, por lo que el documento no sería válido.

Caso 5:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE datos_autor[
3     <!ELEMENT datos_autor (nombre, apellido+)>
4     <!ELEMENT nombre (#PCDATA)>
5     <!ELEMENT apellido (#PCDATA)>
6 ]>
7 <datos_autor>
8     <nombre>Camilo
9         <apellido>Cela</apellido>
10    </nombre>
11 </datos_autor>
```

En este último caso, se comete el error de incluir el elemento apellido dentro del elemento nombre, en lugar de incluirlo dentro del elemento autor, tal como se obliga hacer en la DTD.

<!ELEMENT catalogo (producto+)>

Esta declaración de elemento define un elemento con el nombre catalogo. El elemento catalogo debe incluir uno o más elementos producto, pero al menos uno. El elemento producto es obligatorio y repetible, tal y como se indica el carácter + escrito tras él.

El siguiente documento será un documento válido de acuerdo con la declaración anterior:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE catalogo[
3     <!ELEMENT catalogo (producto+)>
4     <!ELEMENT producto (#PCDATA)>
5 ]>
6 <catalogo>
7     <producto>Televisores color</producto>
8     <producto>Radiocassete doble pletina</producto>
9 </catalogo>
```

<!ELEMENT producto (¿especificaciones+, opciones?, precio+, ¿notas?)>

Esta declaración declara un elemento con nombre producto, que puede contener:

- Uno o más elementos especificaciones. Este elemento tiene carácter obligatorio y puede repetirse.
- Un elemento opciones, con carácter opcional y no repetible (es decir, el elemento opciones puede aparecer o no, pero en caso de que aparezca sólo podrá hacerlo una vez).
- Uno o más elementos precio (obligatoriamente al menos uno).
- Opcionalmente, un elemento notas que no podrá repetirse.

La declaración también indica que los elementos deben aparecer en el siguiente orden: especificaciones, opciones, precio y notas. Este orden se indica al separar los elementos mediante comas.

Caso 1:

Este caso muestra unas estructuras válidas según la DTD anterior:

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <!DOCTYPE producto[
3      <!ELEMENT producto (especificaciones+, opciones?, precio+, notas?)>
4      <!ELEMENT especificaciones (#PCDATA)>
5      <!ELEMENT opciones (#PCDATA)>
6      <!ELEMENT precio (#PCDATA)>
7      <!ELEMENT notas (#PCDATA)>
8  ]>
9  <producto>
10     <especificaciones>Lista de especificaciones</especificaciones>
11     <opciones>aplicable a los casos 4.56 y 4.57</opciones>
12     <precio>567000</precio>
13     <notas>Revisar antes de enviarlo al proveedor</notas>
14 </producto>
```

Caso 2:

La siguiente estructura también será válida:

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <!DOCTYPE producto[
3      <!ELEMENT producto (especificaciones+, opciones?, precio+, notas?)>
4      <!ELEMENT especificaciones (#PCDATA)>
5      <!ELEMENT opciones (#PCDATA)>
6      <!ELEMENT precio (#PCDATA)>
7      <!ELEMENT notas (#PCDATA)>
8  ]>
9  <producto>
10     <especificaciones>Gestionar versiones</especificaciones>
11     <especificaciones>Grabar documentos</especificaciones>
12     <opciones>aplicable a los casos 4.56 y 4.57</opciones>
13     <precio>567000</precio>
14 </producto>
```

Caso 3:

Los siguientes casos muestran una estructura no valida:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE producto[
3     <!ELEMENT producto (especificaciones+, opciones?, precio+, notas?)>
4     <!ELEMENT especificaciones (#PCDATA)>
5     <!ELEMENT opciones (#PCDATA)>
6     <!ELEMENT precio (#PCDATA)>
7     <!ELEMENT notas (#PCDATA)>
8 ]>
9 <producto>
10     <especificaciones>Lista de especificaciones</especificaciones>
11     <opciones>aplicable a los casos 4.56 y 4.57</opciones>
12     <notas>567000</notas>
13     <notas>Revisar antes de enviarlo al proveedor</notas>
14 </producto>
```

En este documento contiene dos errores:

- No se ha incluido el elemento precio dentro del elemento producto (recordamos que precio tiene carácter repetible y obligatorio, por lo que debe aparecer al menos una vez) y
- El elemento notas se ha repetido dos veces, cuando es un elemento opcional y no repetible, que por lo tanto sólo puede aparecer una vez.

Caso 4:

En este segundo caso se muestra una estructura no válida, ya que no se cumplen las restricciones impuestas en la declaración del elemento relativas al orden en el que se deben escribir los elementos contenidos:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE producto[
3     <!ELEMENT producto (especificaciones+, opciones?, precio+, notas?)>
4     <!ELEMENT especificaciones (#PCDATA)>
5     <!ELEMENT opciones (#PCDATA)>
6     <!ELEMENT precio (#PCDATA)>
7     <!ELEMENT notas (#PCDATA)>
8 ]>
9 <producto>
10     <especificaciones>Lista de especificaciones</especificaciones>
11     <precio>aplicable a los casos 4.56 y 4.57</precio>
12     <opciones>567000</opciones>
13     <notas>Revisar antes de enviarlo al proveedor</notas>
14 </producto>
```

El elemento precio se ha escrito después del elemento especificaciones. Esto sería correcto siempre que no se hubiese incluido el elemento opciones. Este elemento, aunque opcional, si aparece debe escribirse detrás del elemento especificaciones y antes del elemento precio.

<!ELEMENT autor (¿nombre, apellido, trabajo?, dirección, bio)>

Esta declaración corresponde al elemento autor. Este elemento puede contener a los siguientes elementos: nombre (obligatorio y no repetible), apellido (obligatorio y no repetible), trabajo (opcional y no repetible), dirección (obligatorio y no repetible) y bio (obligatorio y no repetible).

Los elementos deben aparecer en el mismo orden que se han escrito en la declaración, ya que están separados por comas.

Caso 1:

El siguiente constituye un ejemplo de una estructura válida según la declaración anterior. En el documento aparecen todos los documentos obligatorios. El elemento opcional jobtitle no se ha utilizado en el documento:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE autor[
3     <!ELEMENT autor (nombre, apellido, trabajo?, direccion, bio)>
4     <!ELEMENT nombre (#PCDATA)>
5     <!ELEMENT apellido (#PCDATA)>
6     <!ELEMENT trabajo (#PCDATA)>
7     <!ELEMENT direccion (#PCDATA)>
8     <!ELEMENT bio (#PCDATA)>
9 ]>
10 <autor>
11     <nombre>Juan</nombre>
12     <apellido>Garrido</apellido>
13     <direccion>c/ Castellana 105</direccion>
14     <bio>Nacido en 1973 en Huelva, estudia ... </bio>
15 </autor>
```

Caso 2:

La siguiente estructura no es válida

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE autor[
3     <!ELEMENT autor (nombre, apellido, trabajo?, direccion, bio)>
4     <!ELEMENT nombre (#PCDATA)>
5     <!ELEMENT apellido (#PCDATA)>
6     <!ELEMENT trabajo (#PCDATA)>
7     <!ELEMENT direccion (#PCDATA)>
8     <!ELEMENT bio (#PCDATA)>
9 ]>
10 <autor>
11     <apellido>Garrido</apellido>
12     <trabajo>Cirujano</trabajo>
13     <trabajo>Experto en transplantes</trabajo>
14     <direccion>c/ Castellana 105</direccion>
15     <bio>Nacido en 1973 en Huelva, estudia</bio>
16 </autor>
```

Se han cometido los siguientes errores:

- El elemento obligatorio nombre no se ha indicado.
- El elemento trabajo aparece dos veces, cuando es un elemento no repetible.

<!ELEMENT contenido (p | nota) +>

Se declara un elemento con contenido que puede contener elementos p, elementos nota, pero no los dos elementos a la vez. Tanto los elementos p como nota pueden escribirse en más de una ocasión, y siempre debe escribirse al menos uno de ellos. Esta última característica se indica mediante el signo + escrito tras el paréntesis que cierra la declaración de contenido.

Al escribir el calificador detrás del paréntesis, éste se aplicará a todos los elementos que estén escritos dentro del paréntesis.

Caso I:

La siguiente estructura es válida según la DTD anterior:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE contenido[
3     <!ELEMENT contenido (p | nota)+>
4     <!ELEMENT p (#PCDATA)>
5     <!ELEMENT nota (#PCDATA)>
6     ]>
7 <contenido>
8     <p>Si se produce este problema, siga los siguientes pasos:</p>
9     <nota>Debe comprobar antes que el sistema está correctamente instalado, tal y como se seftala en el apartado 1</nota>
10    <p>Seguidamente, seleccione la entrada Archivo->Abrir</p>
11 </contenido>
```

<!ELEMENT portada (¿titulo, subtítulo?, keyword*, autor+, resumen?)>

En esta declaración se declara el elemento portada. Este elemento puede contener en este orden, los siguientes elementos: titulo (obligatorio Y no repetible), subtítulo (opcional y no repetible), keyword (opcional y repetible) autor (obligatorio y repetible) y resumen (opcional no repetible).

Caso I:

El siguiente documento no sería válido para la DTD anterior:

```
1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <!DOCTYPE portada[
3      <!ELEMENT portada (titulo, subtítulo?, keyword*, autor+, resumen?)>
4          <!ELEMENT titulo (#PCDATA)>
5          <!ELEMENT subtítulo (#PCDATA)>
6          <!ELEMENT keyword (#PCDATA)>
7          <!ELEMENT autor (#PCDATA)>
8          <!ELEMENT resumen (#PCDATA)>
9      ]>
10 <portada>
11     <titulo>La familia de Pascual Duarte</titulo>
12     <titulo>Historia de..</titulo>
13     <resumen>Novela de situacion de plasma</resumen>
14 </portada>
```

En el documento no se cumplen las siguientes reglas:

- El elemento autor no aparece, cuando se trata de un elemento obligatorio.
- El elemento titulo se repite dos veces cuando sólo puede aparecer una única vez.

MODELO DE CONTENIDO MIXTO

En un modelo de contenido mixto, el elemento podrá contener datos de tipo carácter (PCDATA), y otros elementos. En una declaración de contenido mixto siempre se escribe en primer lugar la palabra reservada PCDATA, antes del nombre de cualquier elemento. Tras la palabra reservada PCDATA se escribirán, separados por el carácter | los nombres de los elementos que se pueden incluir dentro del elemento declarado. Tras el paréntesis que cierra la declaración de contenido se debe escribir el carácter * sin interponer ningún espacio en blanco.

Ejemplo:

```
<!ELEMENT especificaciones (#PCDATA | especificación) *>
```

Este es un ejemplo de una declaración de contenido mixto, que incluye datos de tipo carácter y un segundo elemento que deberá definirse en la DTD (elemento especificación).

MODELO DE CONTENIDO ANY

La declaración de contenido ANY indica que un elemento puede contener cualquier combinación de datos de tipo carácter y de elementos, sin que se aplique ninguna restricción. Incluso se pueden llegar a incluir, en un elemento ANY, referencias a elementos que no han sido declarados en la DTD, sin que esto afecte a la validez del documento.

Esta declaración de contenido no se suele utilizar, salvo en las fases de diseño y prueba de las DTD.

DECLARACIONES DE ATRIBUTOS

Los atributos se utilizan en XML para matizar el significado o alcance de los elementos.

Todos los atributos:

- Están vinculados al menos a un elemento.
- Tienen un tipo de dato asociado.
- Pueden tener un valor por defecto.
- Pueden ser obligatorios o no.
- Se deben declarar en la DTD mediante declaraciones de listas de atributos.

NOMBRES DE ATRIBUTOS

En el nombre de un atributo se pueden utilizar los caracteres A-Z, a-z, 0-9, y los signos de puntuación guion (-), punto (.), dos puntos (:) y subrayado (_). Los nombres de atributos deben comenzar con una letra (mayúscula o minúscula), dos puntos (:) o el carácter subrayado (_). No pueden comenzar con un número, un punto(.) o un guion (-). Es decir, se aplican las mismas restricciones que para los nombres de elementos.

TIPOS DE DATOS PARA ATRIBUTOS

Un atributo puede recoger los siguientes tipos de datos:

- **CDATA:** Cadena de caracteres. Se utiliza para atributos que recogen como valor cadenas de caracteres, números y fechas. Un atributo de este tipo puede incluir cualquier cadena de caracteres, salvo los caracteres reservados < y &. El valor de un atributo CDATA puede contener palabras separadas por espacios en blanco.
- **Enumeraciones:** En este caso el atributo puede tomar como valor un valor seleccionado de una lista de opciones cerrada. Los valores de la lista se deben indicar en el momento de crear la DTD.
- **ID única:** Se reserva para los atributos cuya función es identificar de manera única las distintas ocurrencias de un elemento. Los atributos de tipo ID se suelen utilizar con elementos que son destino de hipervínculos, o con elementos a los que se quiere aplicar una presentación única
- **IDREF:** Un atributo de tipo IDREF toma como valor el valor que se haya asignado a un atributo de tipo ID en ese mismo documento.
- **ENTITY:** Toma como valor el identificador o nombre de una entidad externa no procesable declarada en la DTD. Recordamos que las entidades externas no procesables se usan para incluir imágenes o cualquier dato no XML en un documento.
- **NMTOKEN:** Toma como valor una secuencia de caracteres que no puede incluir espacios en blanco. Un NMTOKEN, o name token, puede contener los caracteres A-Z, a-z, 0-9, guion (-), subrayado (_), punto (.) y dos puntos (:), y puede comenzar por cualquiera de estos caracteres.
- **NMTOKENS:** Toma como valor una secuencia de NMTOKEN separados entre sí por espacios en blanco.

CARÁCTER OPCIONAL U OBLIGATORIO DE LOS ATRIBUTOS

Los atributos pueden ser opcionales u obligatorios. Un atributo obligatorio deberá aparecer junto al elemento al que califica, siempre que este elemento aparezca en el documento. Un atributo opcional podrá aparecer o no. Con el fin de indicar en la declaración de una lista de atributos que un atributo es opcional u obligatorio se utilizan, respectivamente, las palabras reservadas `#IMPLIED` y `#REQUIRED`.

Un tipo especial de atributo es el de los atributos fijos. Un atributo fijo es aquel que, siempre que aparece en el documento, debe tomar el mismo valor. Para indicar en la DTD que un atributo es fijo, se utiliza la palabra reservada `#FIXED`.

DECLARAR ATRIBUTOS EN LA DTD

Los atributos se deben declarar en la DTD para poder utilizarlos. ¡La declaración de listas de atributos se escribe entre los caracteres `<!ATTLIST` y `>`. La palabra reservada `ATTLIST` indica que se está declarando uno o más atributos.

Tras la palabra reservada `ATTLIST` se escribe el nombre del elemento para el cual se definen los atributos.

Siguiendo con el nombre y separado de él por un espacio en blanco, se indica el nombre o identificador del atributo, seguido de su tipo de dato, del indicador de si es opcional u obligatorio, y de su valor por defecto en el caso de que lo tenga.

El tipo de dato será indicado con las palabras `ID`, `IDREF`, `CDATA`, `ENTITY`, `ENTITIES` o con los calores de una enumeración. Las enumeraciones se escriben entre paréntesis, separando cada valor del siguiente mediante el carácter `|`.

Si el atributo es opcional se escribirá, detrás del tipo de dato, la palabra reservada `#IMPLIED`. Si es obligatorio, se escribirá la palabra `#REQUIRED`. Si no se indica ninguna de ellas, se interpreta que el atributo es opcional. Por ejemplo, las siguientes declaraciones son equivalentes:

```
<!ATTLIST emp fecnac CDATA #IMPLIED>
```

```
<!ATTLIST emp fecnac CDATA>
```

Finalmente, si se indica un valor por defecto para el atributo, éste se escribirá entre comillas dobles antes del carácter > que marca el final de la declaración de la lista de atributos.

En los atributos asociados a un tipo de dato 'enumeración', siempre se debe indicar un valor por defecto. Este valor por defecto tiene que ser uno de los valores disponibles en la enumeración.

Ejemplo 1: Declaración de atributo

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE emp[
3     <!ATTLIST emp fecnac CDATA #IMPLIED>
4 ]>
5 <emp/>
```

Con esta declaración se declara un atributo para el elemento emp. El nombre del atributo es fecnac. Su tipo de dato es CDATA (datos de tipo carácter), y es un atributo opcional, ya que se utiliza la palabra reservada #IMPLIED.

Ejemplo 2: Declaración de una lista de atributos

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE emp[
3     <!ATTLIST seccion vinculo ID #REQUIRED ultimaAct CDATA #IMPLIED>
4 ]>
5 <emp/>
```

En este segundo caso se declararan dos atributos para el elemento de seccion. El primero tiene como nombre vinculo, es de tipo ID y es obligatorio. Es decir el element oseccion siempre deberá ir acopañado por este atributo. La obligatoriedad del atributo se indica mediante la palabra reserbada #REQUIRED.

El segundo atributo, ultimaAct, recoge como valor datos de tipo carácter y es opcional, tal y como indica la palabra reservada #IMPLIED.

Ejemplo 3: Declaracion de una lista de atributos.

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE emp[
3     <!ATTLIST empleado fecNacimiento CDATA #REQUIRED idPersonal CDATA #REQUIRED fecAlta CDATA #REQUIRED empresa (IBM|Lotus) "IBM">
4     ]>
5 <emp/>
```

En este caso se declaran cuatro atributos para el elemento empleado. Los tres primeros atributos fecNacimiento, idPersonal y fecAlta son obligatorios, como se indica con la palabra reservada #REQUIRED y recogen datos de tipo carácter.

El cuarto atributo, empresa, está asociado a una enumeración con dos valores (IBM y Lotus). El valor por defecto es IBM. Las opciones de la enumeración se separan mediante el carácter | .

ELEMENTOS VACÍOS

En los documentos XML se puede utilizar un tipo especial de elementos: los elementos vacíos.

Estos elementos se caracterizan porque carecen de contenido. En los documentos se puede incluir un elemento vacío de dos maneras:

- Escribiendo la etiqueta de inicio del elemento seguida de su etiqueta de fin, sin escribir nada entre ellas.
`<IMAGE></IMAGE>`
- Utilizando una sintaxis alternativa que emplea una única etiqueta: `<IMAGE />`

Los elementos vacíos, al igual que el resto de elementos, deben declararse en la DTD para poder ser utilizados.

Para indicar que un elemento es un elemento vacío se utiliza la palabra reservada `EMPTY` en el lugar correspondiente a la declaración de contenido.

Ejemplo 1: Declaración de un elemento vacío.

La siguiente declaración define un elemento vacío con nombre imagen.

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE imagen[
3     <!ELEMENT imagen EMPTY>
4 ]>
5 <imagen/>
```


Los elementos vacíos suelen ir acompañados de atributos que matizan su significado. Los atributos correspondientes a un elemento vacío se declararán en la DTD mediante declaraciones de listas de atributos.

En un documento se va a utilizar el elemento vacío mapa para incluir mapas.

El elemento irá acompañado de dos atributos:

- Zona -Recoge el nombre de la zona que muestra el mapa. Será opcional.
- nombreArchivo -Recoge la ruta de los archivos del archivo imagen con extension .gif correspondiente al mapa

En la DTD se tiene que declarar el elemento vacío map y sus dos atributos.

Se haría de la siguiente forma:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE mapa[
3     <!ELEMENT mapa EMPTY>
4     <!ATTLIST mapa zona CDATA #IMPLIED>
5     <!ATTLIST mapa nombreArchivo ENTITY #REQUIRED NDATA="gif">
6 ]>
```

COMENTARIOS

En la DTD se pueden añadir comentarios. El texto de los comentarios no será interpretado por los parsers encargados de procesar la DTD.

Es aconsejable incluir comentarios para explicar que función cumple cada uno de los elementos, atributos o entidades se declaran.

Los comentarios se escriben entre los caracteres especiales `<!--` y `-->`, de la misma forma en que se incluyen los documentos XML.

Entre las marcas de inicio y de fin de un comentario se pueden incluir cualquier carácter, excepto dos guiones seguidos `--`.

Ejemplo

La siguiente DTD es errónea, ya que se ha incluido dos guiones dentro del comentario correspondiente al elemento map.

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE mapa[
3     <!ELEMENT seccion (#PCDATA)>
4     <!-- declaración correspondiente al elemento section. Hay uno para cada apartado -->
5     <!ELEMENT mapa EMPTY>
6     <!--el elemento map se utiliza para insertar un mapa en --imagen gif -->
7 ]>
```

Los comentarios pueden abarcar más de una línea. La siguiente DTD, en la cual el comentario correspondiente al elemento mapa se extiende dos líneas, es correcta:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE mapa[
3     <!ELEMENT seccion (#PCDATA)>
4     <!--declaración correspondiente al elemento section. Hay uno para cada apartado-->
5     <!ELEMENT mapa EMPTY>
6     <!--el elemento map se utiliza para insertar un mapa en imagen gif
7     La imagen se ha declarado en una entidad externa. -->
8 ]>
9 <mapa/>
```

Al introducir comentarios en una DTD tenemos que recordar que no se pueden intercalar comentarios dentro de una declaración (cualquiera que sea el tipo de ésta, es decir, de elemento, de lista, de atributos, de entidades o de notación) De esta forma, la siguiente DTD sería errónea, ya que se ha intercalado un comentario dentro de la declaración del elemento map.

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE mapa[
3     <!ELEMENT map EMPTY <!--para insertar un mapa--> >
4 ]>
```

DECLARACIÓN DE ENTIDADES

Las entidades, que se han explicado en el capítulo anterior, se utilizan en los documentos XML con distintos propósitos:

- Representar caracteres especiales.
- Usar cadenas de texto fáciles de escribir o de recordar en lugar de nombres complejos, muy largos, o que se desconocen en el momento de escribir el documento.
- Incluir documentos XML dentro de otro documento XML, para crear documentos modulares.
- Incorporar al documento XML componentes no XML: imágenes, multimedia, HTML, etc.

En el primer caso nos referimos a las entidades de tipo carácter. En el segundo caso a las entidades de texto. En el tercero a las entidades externas procesables. Y en el último, a las entidades externas no procesables.

Todas las entidades a las que se hace referencia en un documento XML deben haber sido declaradas en la DTD asociada al documento. Si se hace una referencia a una entidad no declarada en la DTD, el documento ni siquiera estará bien formado.

Las declaraciones de entidades se escriben entre los caracteres `<!--` y `-->`, al igual que el resto de declaraciones de la DTD. Tras la marca de inicio (caracteres `<!--`) se escribirá la palabra reservada `ENTITY`.

DECLARAR ENTIDADES DE TEXTO

En este tipo de entidad, tras la palabra reservada ENTITY se escribirá, separado por un espacio, el identificador o nombre de la entidad. Este nombre es el que vamos a usar para hacer referencia a la entidad en el documento XML.

Tras el nombre de la entidad se escribe, entre comillas, el contenido de la entidad o la forma con la que ésta debe resolverse. El contenido de la entidad es el texto por lo que deben ser sustituidas las referencias a la entidad. Entre el nombre de la entidad y su contenido se deja un espacio en blanco.

Ejemplo

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE prod[
3   <!ENTITY nuevoprod "KTD-50-789890 A5">
4 ]>
```

En este ejemplo se declara una entidad con nombre nuevo prod y contenido KTD-50-789890 A5. El autor del documento deberá escribir la cadena de texto &nuevoprod; cuando tenga que indicar el nombre del producto. La aplicación XML se encargará de remplazar esta cadena por el nombre real del producto (KTD-50-789890 A5) en el momento de mostrar el documento.

DECLARAR ENTIDADES EXTERNAS PROCESABLES

Para declarar este tipo de entidades se escribe, después de la palabra reservada ENTITY, el identificador o nombre de la entidad.

Tras el nombre de la entidad se escribe la palabra reservada SYSTEM. Esta palabra indica al sistema que el contenido de la entidad es un recurso externo, que puede ser un archivo local o remoto.

Detrás de la palabra reservada SYSTEM se escribe, entre comillas dobles, la ruta del archivo o el URL en el que se encuentra el documento XML cuyo contenido se quiere mostrar en lugar de las referencias a la entidad.

Ejemplo.

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE licencia[
3     <!ENTITY licencia SYSTEM "c:\licencia.xml">
4 ]>
```

En este caso se declara una entidad con nombre licencia, que corresponde a un documento XML con la licencia de uso de un producto.

Como el texto de la entidad se encuentra situado en un archivo externo, se escribe la palabra reservada SYSTEM después del nombre de la entidad. Siguiendo a SYSTEM, entre comillas dobles, se escribe la ruta de directorios y el nombre del archivo XML que se debe incorporar al documento desde el que se hace la referencia.

DECLARAR ENTIDADES EXTERNAS NO PROCESABLES.

Las entidades externas no procesables apuntan a archivos no XML que se quieren incluir en el documento XML.

Su declaración es similar a la de las entidades externas procesables, con dos diferencias:

- En estas declaraciones es posible utilizar la palabra reservada PUBLIC en lugar de SYSTEM, para apuntar a un recurso disponible en el denominado catálogo.
- Las declaraciones de entidades externas no procesables deben ir acompañadas de una notación.

La sintaxis que tenemos que utilizar para ellos es la que podemos ver a continuación

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE nombre[
3     <!ENTITY nombreEntidad SYSTEM | PUBLIC "contenido" NDATA notacion>
4 ]>
```

Donde:

- nombreEntidad corresponde al nombre o identificador de la entidad. Es la cadena de caracteres que se utilizará al escribir el documento para hacerle referencia.
- Tras el nombre de la entidad se escribirá:
 - La palabra reservada SYSTEM si el contenido de la entidad se encuentra en un archivo externo local o remoto, o
 - La palabra reservada PUBLIC si el contenido de la entidad se encuentra en un archivo local o remoto que haya sido registrado en el catálogo de la aplicación XML.
- Si se ha utilizado la palabra reservada SYSTEM, tras ella se escribirá entre comillas dobles la ruta de directorios y nombre del archivo con el que se tienen que sustituir las referencias a la entidad.
- Si se ha usado la palabra reservada PUBLIC, tras ella se escribirá el identificador de un recurso público disponible en el catálogo de la aplicación XML. Este identificador se escribirá entre comillas dobles.

Tras el contenido de la entidad, se escribirá la palabra reservada NDATA, seguida de un espacio en blanco y del identificador de la notación correspondiente al tipo de recurso externo al que hace referencia la declaración de la entidad. NDATA tomará como valor el identificador de una notación declarada en la DTD. El valor que se indica después de NDATA no se debe escribir entrecomillado.

DECLARACIÓN DE NOTACIONES

Para cada tipo de entidad, externa no procesable se debe incluir una notación. Así, se tendrá que declarar una notación para las imágenes de tipo tif, gif, bmp, etc. Las notaciones cumplen distintos propósitos:

- Indicar la ruta de directorios y el nombre del programa encargado de procesar la entidad (por ejemplo un visor esopecial para archivos de imangen).
- Apuntar a un lugar en el que existe documentación sobre el formato.
- Otras funciones.

En este sentido la norma es abierta y cada aplicación XML puede utilizar las notaciones con uno u otro propósito.

Las notaciones se suele declarar al principio de la DTD, utilizando la siguiente sintaxis:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE nombre[
3     <!NOTATION nombreNotacion SYSTEM "contenido">
4 ]>
5 <nombre></nombre>
```

Donde nombreNotacion es el identificador para la notación. Este es el edentificador que se debe escribir tras la palabra reservada NDATA en las declaraciones de entidades externas no procesables.

Tras la palabra reservada SYSTEM se escribira entre comillas dobles el contenido de la notación. Como hemos indicado, este contenido puede tratar el recurso externo, una URL en el que se encuentra documentación sobre el formato, etc.

Ejemplo.

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE nombre[
3     <!NOTATION gif SYSTEM "C:\Program Files\GifViewer.exe">
4     <!ENTITY logo SYSTEM "c:\imgs\logo.gif" NDATA gif>
5 ]>
6 <nombre></nombre>
```

En primer lugar se declara una notación para los recursos externos que son imágenes en formato GIF. En la notación se indica un nombre para esta notación (gif), y se indica la ruta de directorios de un archivo con el que se tienen que visualizar los archivos de este tipo.

En la declaración de entidad siguiente, se declara una entidad con nombre logo, que se corresponde al archivo físico e:\imgs\ logo.gif. Este recurso externo está asociado a la notación gif (en la sección NDATA), de forma que si el programa XML tiene que procesar el recurso externo, lo hará llamando al programa indicado en la notación.

Tenemos que recordar que el tratamiento de notaciones y entidades externas no procesables es un aspecto que está estrechamente asociado a los programas que se utilicen para procesar los documentos XML. Por ejemplo, los editores XML incorporan visores para distintos tipos de archivos y las notaciones para los tipos de archivos externos más comunes suelen estar predefinidas en estos sistemas.

ENTIDADES PARÁMETRO.

En una DTD se puede incluir un tipo especial de entidades, llamadas entidades parámetro. Estas entidades se utilizan para reutilizar declaraciones dentro de la DTD.

Por ejemplo, si tenemos que utilizar una declaración de atributos en dos o más elementos, el mismo modelo de contenido, etc., podríamos crear una entidad parámetro reutilizable, y evitar así la necesidad de duplicar las declaraciones.

Para declarar una entidad parámetro, se debe utilizar la siguiente sintaxis:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE nombre[
3     <!ENTITY % nombre "contenido">
4     ]>
5 <nombre/>
```

Tras la palabra reservada ENTITY se escribirá un espacio en blanco, y a continuación el carácter %. Tras este carácter, se escribe otro espacio en blanco seguido del nombre o identificador de la entidad. Tras el nombre de la entidad, y seguido por otro espacio en blanco, se escribe entrecomillado el texto por el que se deben sustituir las referencias a la entidad.

Por otra parte, para hacer referencia a esta entidad parámetro en otras declaraciones de la DTD, bastará con escribir su identificador precedido por el carácter % y seguido por un punto y coma (;).

Ejemplo 1: Entidad parámetro para reutilizar modelo de contenido

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE autor[
3     <!ELEMENT autor (nombre, apellido+)>
4     <!ELEMENT lector (nombre, apellido+)>
5     <!ELEMENT nombre (#PCDATA)>
6     <!ELEMENT apellido (#PCDATA)>
7 ]>
8 <autor>
9     <nombre></nombre>
10    <apellido></apellido>
11 </autor>
12 <lector>
13     <nombre></nombre>
14     <apellido></apellido>
15 </lector>
```

En este caso, es posible reutilizar el modelo de contenido de los elementos autor y lector. Para ello, se declarará una entidad de tipo parámetro que tomará como valor la declaración de contenido que se quiere reutilizar:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE autor[
3     <!ENTITY % contPersona "(nombre, apellido+)">
4     <!ELEMENT autor %contPersona;>
5     <!ELEMENT lector %contPersona;>
6     <!ELEMENT nombre (#PCDATA)>
7     <!ELEMENT apellido (#PCDATA)>
8 ]>
```

Ejemplo 2: Entidad parámetro para reutilizar atributos.

Tras declarar esta entidad, se podrían sustituir las declaraciones de los atributos comunes para los elementos cliente y empleado por una referencia a la entidad parámetro atributosComunes:

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <!DOCTYPE cliente[
3     <!ELEMENT cliente EMPTY>
4     <!ATTLIST cliente %atributosComunes; >
5     <!ELEMENT empleado EMPTY>
6     <!ATTLIST empleado %atributosComunes; >
7     <!ATTLIST empleado nombre CDATA #IMPLIED>
8 ]>
```

DISEÑO DE DTD

El diseño de las DTD es una de las fases críticas en un proyecto XML. En primer lugar, se debe ver la posibilidad de utilizar una de las muchas DTD diseñadas por organismos o grupos de organizaciones dedicadas a la normalización a la reutilización de DTD existentes ofrece muchas ventajas, entre ellas:

- Reducción de costes en el desarrollo e implantación del sistema XML.
- Posibilidad de intercambiar nuestra información y establecer proyectos de colaboración con otras organizaciones que están utilizando DTD estandarizadas.

Sin embargo, en respuesta a problemas más específicos puede ser necesario diseñar una nueva DTD. En este caso se deben seguir los siguientes pasos:

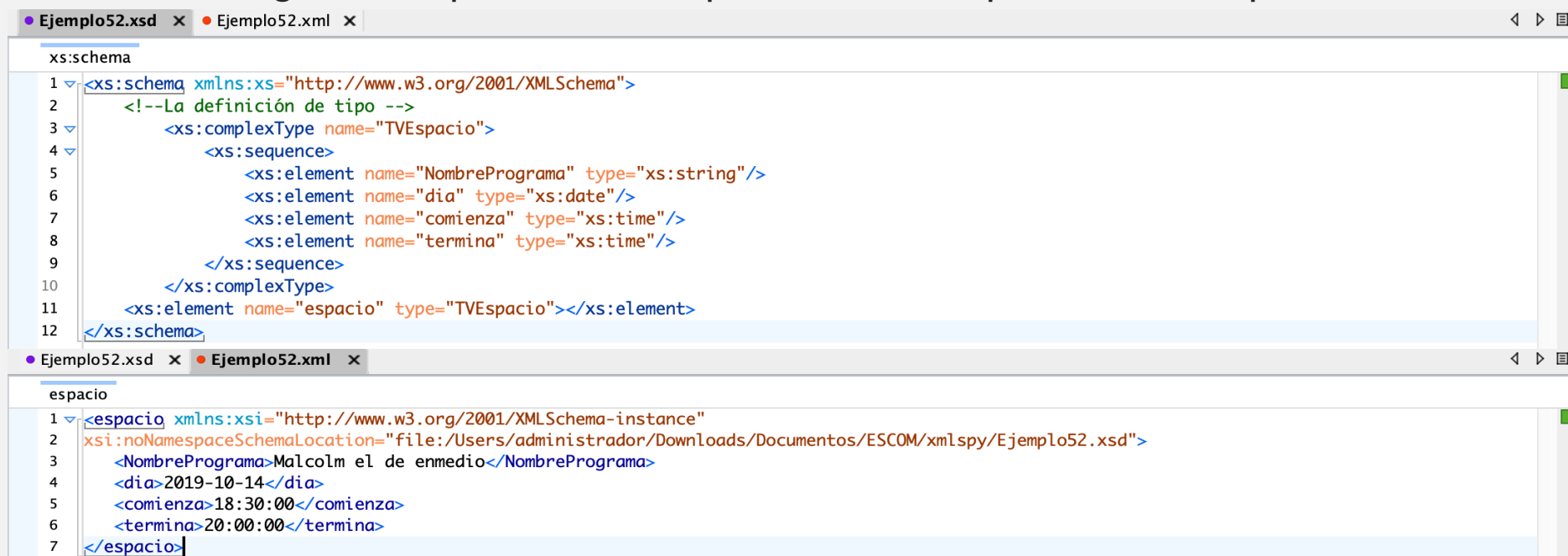
1. Identificar los distintos tipos de documentos que se van a generar o procesar. Para cada tipo de documento se tendrá que diseñar una DTD.
2. Identificar los elementos estructurales de cada tipo de documento. De esta forma se podrán extraer los elementos que conformarán la DTD.
3. Identificar la frecuencia con la que puede aparecer cada uno de los elementos en un documento.
4. Identificar el carácter opcional u obligatorio de cada elemento.
5. Estudiar si el contenido de cada elemento se puede "subdividir" en otros elementos significativos, y elegir el nivel de granularidad adecuado para el tipo de procesamiento que se tenga que realizar.
6. Identificar el orden en el que deben aparecer los elementos.
7. Identificar los atributos que pueden calificar a cada elemento.
8. Para cada atributo, decidir cuál va a ser su tipo de dato, si va a ser opcional u obligatorio, y si puede tener un valor por defecto.
9. Considerar cómo se van a representar en el documento los datos no XML; es decir, imágenes, mapas, planos, o cualquier otro elemento gráfico o multimedia. Generalmente, se utilizarán elementos vacíos para este tipo de objetos.
10. Finalmente, se debe decidir la estructura jerárquica de los elementos: cómo se anidan y cómo se pueden contener unos a otros.

Una vez diseñada la DTD, ésta debe probarse y deben realizar distintas pruebas para ver si la estructura final de la DTD es la adecuada para el tipo de procesamiento que se quiere hacer.

ESQUEMA XML: UN MÉTODO PRACTICO

El esquema XML es un mecanismo de definición de tipo que depende en gran medida de los conceptos de orientación a objetos y representación tradicional de información. El esquema XML se crea alrededor del concepto de tipos (como entero, fecha y booleano) y de instancias de estos tipos (por ejemplo, el elemento pi es una instancia del tipo float).

Para ilustrar esto, tome el siguiente esquema mínimo, que describe un tipo llamado TvEspacio.



```
Ejemplo52.xsd x Ejemplo52.xml x
xs:schema
1 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2   <!--La definición de tipo -->
3   <xs:complexType name="TVEspacio">
4     <xs:sequence>
5       <xs:element name="NombrePrograma" type="xs:string"/>
6       <xs:element name="dia" type="xs:date"/>
7       <xs:element name="comienza" type="xs:time"/>
8       <xs:element name="termina" type="xs:time"/>
9     </xs:sequence>
10  </xs:complexType>
11  <xs:element name="espacio" type="TVEspacio"/></xs:element>
12 </xs:schema>

Ejemplo52.xsd x Ejemplo52.xml x
espacio
1 <espacio xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:noNamespaceSchemaLocation="file:/Users/administrador/Downloads/Documentos/ESCOM/xmlspy/Ejemplo52.xsd">
3   <NombrePrograma>Malcolm el de enmedio</NombrePrograma>
4   <dia>2019-10-14</dia>
5   <comienza>18:30:00</comienza>
6   <termina>20:00:00</termina>
7 </espacio>
```

Este pequeño esquema ya nos dice al menos cuatro aspectos muy importantes del mecanismo del propio esquema XML:

- Los documentos de esquema XML estan escritos en XML. En lugar de usar otro tipo de notacion, con sus propias reglas, los esquemas XML son documentos XML bien formados; por tanto, mejorará en gran medida sus curvas de aprendizaje y legibilidad.
- Existen tipos básicos simples que se pueden usar de forma directa en los elementos (por ejemplo, string, date y time)
- Existe un mecanismo para construir tipos complejos, declarando dentro de ellos elementos de ciertos tipos. Esto es analogo a las construcciones de clases en los lenguajes orientados a objetos (como Java)
- Existe una diferencia clara entre la creacion de un tipo y la declaracion de un elemento de ese tipo.

TIPOS

En el esquema XML, los tipos estan divididos en dos categorias, simples y complejos. Los tipos simples solo pueden contener informacion de caracteres; no pueden tener atributos ni contenido de elementos. Los tipos complejos, por otra parte, pueden tener cualquier combinacion de contenido de elementos, informacion de caracteres y atributos.

El esquema XML proporciona un conjunto b;asico de tipos simples que se pueden usar de forma directa para declarar elementos que coinciden con nociones populares, como entero o fecha. En la siguiente tabla se presenta el conjunto completo de los tipos básicos y su significado.

Tipos integrados simples	Ejemplo	Tipos integrados simples	Ejemplo
String	“Confirma que esto es eléctrico”	name	EnviarA (es un tipo Name de XML 1.0)
boolean	True, false, 1, 0	Qname	Dirección (es un nombre calificado de espacio de nombre XML)
Float	INF, -1E4, -0, 0, 12.78E-2, 12, NaN, equivalente a un punto flotante de 64 bits de precision doble		
Decimal	-1.23, 0, 123.4, 1000.00	NCName	Dirección (es un nombre no canónico de espacio de nombre de XML, por ejemplo, puede ser un nombre calificado sin el prefijo y los dos puntos)
timeInstant	1999-05-31T13:20:00.000-05:00 (31 de mayo de 1999 a la 1:20pm, tempo estandar del este, que se encuentra cinco horas atrás de la coordenada de tiempo universal	Integer	-123456789, -1, 0, 1, 123456789
timeDuration	PIY2M3DT10H30M12.3S (1 año, 2 meses, 3 días, 10 horas, 30 minutos, 12.3 segundos)		
binary	100010	non-positive-integer	-123456789, -1, 0
uri-reference	http://www.exaple.com/ http://www.example.com/doc.html#ID5	negative-integer	-123456789, -1
ID	Es un tipo de atributo ID de XML 1.0	long	-1, 1234567543233
IDREF	Es un tipo de atributo ID de XML 1.0	int	-1, 12345675
ENTITY	Es un tipo de atributo ENTITY de XML 1.0	short	-1, 12678
NOTATION	Es un tipo de atributo NOTATION de XML 1.0	byte	-1, 126
Lenguaje	en-GB, en-US, sp y otros valores válidos para xml:lang, según la definición XML 1.0	non-negative-integer	0, 1, 126789
IDREFS	Es un tipo de atributo IDREFS de XML 1.0	unsigned-long	0, 12678967543233
ENTITIES	Es un tipo de atributo ENTITIES de XML 1.0	unsigned-int	0, 1267896754
NMTOKEN	US, es un tipo de atributo NMTOKEN de XML 1.0	unsigned-short	0, 12678
NMTOKENS	“US UK”, es un tipo de atributo NMTOKENS de XML 1.0	unsigned-byte	0, 126
		positive-integer	1, 126789
		date	1999-05-31---05 (quinto día de cada mes)
		time	13:20:00:000, 13:20:00.000-05:00

El primer uso, y el más popular, de los tipos integrados es la declaración directa de elementos y atributos que cumplen con ellos. El código siguiente es la declaración de un elemento de un tipo de programa que contiene varios elementos secundarios de tipos integrados.



```
xs:schema xs:element
1 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2   <!--La definición de tipo -->
3   <xs:complexType name="Programa">
4     <xs:sequence>
5       <xs:element name="nombre" type="xs:string"/>
6       <xs:element name="descripcion" type="xs:string"/>
7     </xs:sequence>
8     <xs:attribute name="id-programa" type="xs:ID"/>
9     <xs:attribute name="primera-transmision" type="xs:date"/>
10   </xs:complexType>
11   <xs:element name="programa" type="Programa"/></xs:element>
12 </xs:schema>
Ejemplo53.xsd x Ejemplo53.xml x

programa
1 <programa xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:noNamespaceSchemaLocation="file:/Users/administrador/Downloads/Documentos/ESCOM/xmlspy/Ejemplo53.xsd"
3   id-programa="nbc-18" primera-transmision="1990-01-01">
4   <nombre>Sinfield</nombre>
5   <descripcion>Programa acerca de nada</descripcion>
6 </programa>
```

Esta definición de tipo complejo muestra dos aspectos importantes adicionales sobre los tipos simples:

- Los atributos se pueden declarar de acuerdo con un tipo simple específico. Debido a que los valores de tipos simples no pueden incluir ningún contenido de elemento, esto no solo tiene sentido, sino que es una solución cabal para las limitaciones sintácticas de la representación de entidades puramente internas de las DTD.
- Los atributos también pueden tener tipos útiles definidos en las DTD, facilitando así la transición y operación entre las DTD y el esquema XML; además pueden tener la representación de conceptos que son muy útiles en la creación de DTD.

USO DE TIPOS COMPLEJOS DENTRO DE TIPOS COMPLEJOS.

Hasta ahora solo hemos revisado los tipos complejos que emplean elementos secundarios simples. Sin embargo, en la vida real no llegaríamos muy lejos con esto; por lo tanto, es momento de analizar la anidación de tipos complejos.

Una declaración de elemento, en su forma más simple, es la unión de un nombre de elemento y un tipo particular.



```

Ejemplo54.xsd x Ejemplo54.xml x
xs:schema
1 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2   <xs:complexType name="Programa">
3     <xs:sequence>
4       <xs:element name="nombre" type="xs:string"/>
5       <xs:element name="productor" type="xs:string"/>
6     </xs:sequence>
7     <xs:attribute name="id-programa" type="xs:ID"/>
8     <xs:attribute name="primera-transmision" type="xs:date"/>
9   </xs:complexType>
10  <xs:complexType name="TVEspacio">
11    <xs:sequence>
12      <xs:element name="programa" type="Programa"/>
13      <xs:element name="dia" type="xs:date"/>
14      <xs:element name="comienza" type="xs:time"/>
15      <xs:element name="termina" type="xs:time"/>
16    </xs:sequence>
17  </xs:complexType>
18  <xs:element name="espacio" type="TVEspacio"/></xs:element>
19 </xs:schema>
Ejemplo54.xsd x Ejemplo54.xml x
espacio
1 <espacio xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:noNamespaceSchemaLocation="file:/Users/administrador/Downloads/Documentos/ESCOM/xmlspy/Ejemplo54.xsd">
3   <programa id-programa="snl" primera-transmision="1974-01-03">
4     <nombre>Saturdat Night Live</nombre>
5     <productor>Francisco</productor>
6   </programa>
7   <dia>2001-10-24</dia>
8   <comienza>23:00:00</comienza>
9   <termina>23:58:00</termina>
10 </espacio>
```

EXPRESIONES NORMALES.

- El lenguaje de expresiones normales que se usa en los patrones es muy similar al de Perl; sin embargo, un usuario no requiere experiencia alguna en Perl para entenderlo. Un análisis completo de la teoría del lenguaje de expresiones de patrones está fuera del alcance de este capítulo pero los ejemplos siguientes, que se muestran en la tabla proporcionarían una idea de las posibilidades.

Expresion	Coincidencias
a*x	x, ax, aax, aaax
a?x	ax, x
(a b)+x	ax, bx, aax, abx, bax, bbx, aaax, aabx, abax, abbx, baax, babx, bbax, bbbx, aaaax
[abcde]x	ax, bx, cx, dx, ex
[a-e]x	ax, bx, cx, dx, ex
[-ae]x	-x, ax, ex
[ae-]x	ax, ex, -x
Capítulo \d	Capítulo 0, Capítulo 1, Capítulo 2
Capítulo\s\d	Capítulo seguido de un solo carácter de espacio en blanco (espacio en blanco, tabulador, linea nueva, etcetera), seguido por un solo dígito
Capítulo\s\w	Capítulo seguido de un solo carácter de espacio en blanco (espacio en blanco, tabulador, linea nueva, etcetera), seguido por una letra o digito de un carácter de palabra XML 1.0
[a-e-[bd]]x	ax, cx, ex
[^0-9]x	Cualquier carácter que no sea digito seguido por el carácter x
\Dx	Cualquier carácter digito seguido por el carácter x
.x	Cualquier carácter seguido por el carácter x
.*abc.*	l x2abc, abc l x2, z23456abchurra
ab{2}x	abbx
ab{2,4}x	abbx, abbbx, abbbbx
ab{2,}x	abbx, abbbx, abbbbx
(ab){2}x	ababx

FACETAS NOTABLES

- Entre las facetas notables estan minInclusive y maxInclusive (y sus contrapartes minExclusive y maxExclusive). Para revisar la forma en la que se usan, suponga que desea modelar un elemento de tipo que corresponda de forma exacta a la noción Java de short. Puede hacerlo con las facetas minInclusive y maxExclusive del tipo simple int, de la forma siguiente:

The screenshot displays four panels from the XMLSpy application, illustrating the definition and use of the `JavaShort` element.

Panel 1: Schema Definition
The top panel shows the `Ejemplo57.xsd` schema file. It defines an `xs:schema` with the following structure:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="JavaShort">
    <xs:simpleType>
      <xs:restriction base="xs:integer">
        <xs:minInclusive value="-32767"/>
        <xs:maxExclusive value="32769"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
</xs:schema>
```

Panel 2: Instance with Value -32768
The second panel shows the `Ejemplo57.xml` file with an instance of `JavaShort` containing the value `-32768`. The XML is:

```
<JavaShort xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:/Users/administrador/Downloads/Documentos/ESCOM/xmlspy/Ejemplo57.xsd"> -32768 </JavaShort>
```

Panel 3: Instance with Value -32767
The third panel shows the same XML file, but the value has been changed to `-32767`.

```
<JavaShort xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:/Users/administrador/Downloads/Documentos/ESCOM/xmlspy/Ejemplo57.xsd"> -32767 </JavaShort>
```

Panel 4: Instance with Value 32768
The fourth panel shows the XML file with the value `32768`.

```
<JavaShort xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:/Users/administrador/Downloads/Documentos/ESCOM/xmlspy/Ejemplo57.xsd"> 32768 </JavaShort>
```

Panel 5: Instance with Value 32769
The fifth panel shows the XML file with the value `32769`.

```
<JavaShort xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:/Users/administrador/Downloads/Documentos/ESCOM/xmlspy/Ejemplo57.xsd"> 32769 </JavaShort>
```

¿QUÉ ES EL DOM?

DOM (Document Object Model) es una API, una interfaz de programación para el desarrollo de aplicaciones, orientado a objetos y diseñado para trabajar con documentos HTML validos y documentos XML bien formados.

Así, en DOM un documento no es otra cosa que un conjunto de objetos que se relacionan entre si mediante una jerarquía de objetos en forma de árbol. DOM permite el acceso tanto a los objetos como a las relaciones entre ellos. Estos objetos a los que nos referimos se construyen a partir de elementos, atributos, entidades, instrucciones de proceso, etc. Y las relaciones vienen definidas en el propio documento a partir del que se elabora el árbol: elementos padres, elementos hijos, etc.

Dicen los libros que un paradigma es una forma de ver el mundo, una filosofía que interpreta la realidad. En el paradigma de la programación orientada a objetos, el mundo este compuesto de objetos, que se solían y amablemente se responden, intercambiando información. Bajo ese punto de vista, un objeto representa un comportamiento (métodos) y unas cualidades (atributos) propias de su tipo (su clase). Las clases de objetos posibles en un documento, así como su comportamiento y propiedades serán motivo de estudio.

Conocido esto, el lector ya puede intuir que nos permitirá hacer el DOM, algunas de sus utilidades son:

- Construcción y modificación de nuevos documentos: La construcción de documentos se puede realizar a partir de cero, o a partir de otros documentos, DOM permite cortar y pegar partes de un documento y pegarlas donde desee, incluso en otro documento
- Navegación y modificación de la jerarquía del documento: en DOM es posible extraer información de las relaciones entre los objetos, existen métodos para obtener listas con los nodos hijos de un dato, o conocer sus hermanos, su padre, etc. También es posible modificar esta estructura eliminando nodos del árbol o añadiendo hijos a un nodo, trasladando nodos, etc.
- Gestión de objetos: Los objetos presentan ciertas cualidades, como, por ejemplo, el valor de un atributo y que es posible modificar o consultar.

DOM crea una representación del documento en la memoria, por lo que consume ciertos recursos, de tiempo y de maquina. DOM y es útil las aplicaciones que deben realizar algunas de las funciones anteriores. DOM permite al programador abstraerse de ciertas particularidades de XML, que ahora no necesita conocer.

W3C. DOM DE NIVEL 1 Y DE NIVEL 2

DOM es una especificación de World Wide Consortium, W3C y esta reconocido como un estándar internacional. En el momento de la edición del libro DOM presenta dos niveles o versiones:

- DOM nivel 1 recomendación desde el 1 de octubre de 1998
- Dom nivel 2, que alcanzo el estatus de recomendación el 13 de noviembre de 2000. DOM nivel 2, presenta ciertas mejoras sobre DOM nivel 1, entre las que destacan el soporte a espacios de nombres, tratamientos de eventos, gestión de las vistas de un documento, y hasta catorce módulos (la mayoría nuevos) con diversas utilidades. Ciertas interfaces definiéndolos en DOM nivel 2.
- Actualmente, se esta trabajando en el nivel 3 de DOM que promete mas mejores.

Términos y nociones básicas sobre orientación a objetos

Quizás antes de comenzar a explicar más detalladamente algunos de las interfaces que están definidos en DOM sea interesante recordar o incluir ciertas definiciones básicas para aquellos lectores que lo necesiten:

- Los objetos son componentes de software auto contenidos, modelados a partir de "cosas", archivos, botones, ventanas, etc. Todos los objetos se definen en función de las características que poseen y del comportamiento que presentan. Así, un perro puede pensarse como un objeto que tiene como atributos un color de pelo, un nombre, etc., algunos de sus comportamientos incluyen ladrar, correr, dormir ...
- Clases. Un objeto pertenece siempre a una clase, que podría denominarse su descripción lógica. En el mundo real una clase sería análoga a una definición o a un conjunto de pautas que hacen que algo se clasifique en un grupo y no en otro. Los objetos como las realizaciones concretas que encajan en estas descripciones. De esta manera se puede definir la clase perro con los atributos nombre y color de pelo, etc. y los comportamientos ladrar, correr y dormir. Un perro en concreto es una realización de esta clase.
- Métodos: las clases y sus realizaciones, los objetos, presentan ciertos comportamientos característicos, son sus métodos. Los métodos pueden ser de varios tipos internos a la clase, públicos, abstractos, etc. Existen dos métodos que se llaman automáticamente cuando se crea un objeto de una clase, el constructor, que crea e inicializa el objeto y el destructor de la clase.
- Atributos: los objetos, al igual que ciertos comportamientos también poseen ciertas características propias, son como adjetivos o propiedades. Como buena práctica de programación se aconseja modificar o inicializar estos atributos mediante métodos de la clase.

- Herencia, la herencia permite que clases hijas o especializaciones de una clase, abstracta o no, reciban atributos y métodos por su relación con otra clase a la que se llamará "padre" y con la que comparte un grado afinidad alto. La herencia está soportada directamente por otro concepto "la jerarquía de clases".
- Una interfaz es una definición sin implementación, especifica la forma que deben tener los métodos de una clase y los atributos, pero no la implementación concreta, en definitiva, regula aquello que se espera que haga o que ofrezcan las clases que lo implementen, pero no la manera en la que esto se logra. Una interfaz es única pero sus implementaciones concretas son infinitas.
- En el capítulo se hablará de extender clases. Una clase que extend heredará de ella sus métodos y atributos, quedando estos a disposición del programador que ya no necesita implementarlos. Sobre estas características heredadas el programador define comportamientos adicionales o redefine métodos antiguos, añade atributos, en definitiva, crea diferencias que distinguirán a esta clase de la extendida y que crearán una especialización de esta. Así, se podría pensar que un hombre o una mujer son especializaciones de la clase persona
- Se dice que una clase implementa una determinada interfaz si externamente cumple con los requisitos de comportamiento que se definen en la interfaz y además incluye el código que da vida a esos comportamientos y hace que se pueda interaccionar con ellos en los programas. Distintas implementaciones ofrecen el mismo comportamiento aparente, el que garantiza la interfaz, pero la forma de lograr el resultado varía, por lo que algunas implementaciones pueden ser más eficientes que otras en determinados aspectos, necesidad de memoria, rapidez, eficiencia, etc. El usuario tiene la opción de elegir la implementación que más le convenga.
- Ancestro, un ancestro de un nodo es cualquier nodo anterior en la jerarquía en el camino de un nodo hacia la raíz
- Nombre cualificado: el nombre cualificado de un elemento o de un atributo se compone de un prefijo asociado a su espacio de nombres y, separado por dos puntos, el nombre del atributo o del elemento (nombre local).
- URL del espacio de nombres, los espacios de nombres, para asegurar su unicidad, se asocian con un URL.

IMPLEMENTACIONES

DOM define, a través de interfaces, los tipos de objetos que modelarán los documentos, sus métodos, sus atributos y las relaciones entre ellos, pero no provee ninguna implementación concreta, podríamos pensar que DOM define una carcasa genérica pero no el interior (el código, la implementación) que puede tomar multitud de formas. Será tarea del programador seleccionar una implementación en concreto.

DOM se ha diseñado para que pueda ser utilizada con una gran variedad de lenguajes de programación, o que facilita que haya implementaciones para un sinfín de lenguajes de programación como Java, Perl, JavaScript, C++, Python, etc. Esta bondad de su diseño se muestra especialmente en la elección de OMG JDL como una forma expresa las interfaces que no depende de ningún lenguaje, es decir, en un formato dependiente del lenguaje de programación que finalmente se utilice. No obstante, existen “lenguaje bindings” que son representaciones de estos interfaces genéricos en un lenguaje en concreto, en el capítulo se estudiara el “lenguaje binding” para Java

Algunas implementaciones para Java disponibles son las que a continuación se listan

- Xerces: xml.apache.org/dist/xerces-j
- SAXON: <http://users.iclway.co.uk/mhkay/saxon>
- XML para Java: <http://alphaworks.ibm.com/formula/xml>

DOM esta diseñado para que sea fácil cambiar de implementación sin tener que cambiar los programas. Esto es, si utilizan los métodos que define la implementación se podría cambiar el procesador XML sin necesidad de cambiar el programa.

Una determinada implementación de DOM puede ofrecer características adicionales, clases e interfaces que no están recogidas en la Recomendación de DOM y que posibiliten al programador, por ejemplo, almacenar en disco los documentos generados

DOM DE NIVEL 2

- Ya se ha comentado brevemente que es DOM de nivel 2, a continuación, aprenderemos algo mas sobre esta API. Para empezar, se listarán los módulos DOM de nivel 2 se compone de 14 módulos que responden a diversas necesidades y que son los siguientes
- Modulo Núcleo (Core Module)
- Modulo de XML (XML Module)
- Modulo de HTML (HTML Module).
- Modulo de Visitas (Views Module)
- Modulo de Hojas de Estilo (Style Sheets Module)
- Modulo de Hojas de Estilo en Cascada (CSS Module)
- Modulo de Hojas de Estilo en Cascada 2 (CSS2 Module)
- Modulo de eventos de interfaz de usuario (User Interface Events Module)
- Modulo de Eventos del Ratón (Mouse Events Module)
- Modulo de Eventos HTML (HTML Events Module).

El modelo de objetos expuestos por el DOM para páginas HTML no solo permitía trabajar con los distintos elementos de las páginas (hipervínculos, tablas, párrafos, etc.) También era posible trabajar con la ventana del navegador a través de un objeto especial llamado Window.

Para el lenguaje XML también disponemos de un DOM que nos permite acceder y manipular desde un programa los distintos componentes de un documento (elementos, atributos, comentarios, entidades etc.).

Es importante señalar que el DOM es independiente de cualquier lenguaje de programación. Es decir, no está vinculado ni depende de un lenguaje de programación específico, por lo que puede usarse con distintos lenguajes de programación: JavaScript, C, Visual Basic, etc.

El W3C se encarga de mantener la especificación del DOM para XML. Tomando como base esta especificación, los distintos fabricantes de aplicaciones informáticas implementan la funcionalidad que se describe en ella, representando los nombres de objetos, propiedades y métodos indicados en las especificaciones del W3C.

Actualmente, el W3C ha descrito tres niveles de conformidad con el DOM.

El primer nivel define interfaces para la navegación a través de los documentos con marcas y para manipular su estructura y contenido.

El segundo nivel de conformidad con el DOM establece interfaces para manipular hojas de estilo. Nivel define interfaces para la carga de memoria de documentos y su grabación.

En los siguientes apartados se describe el modelo de objetos DOM implementado por Microsoft a partir de la especificación del W3C. este modelo de objetos no servirá para procesar documentos XML desde programas escritos en visual Basic, VBA (Visual Basic for Applications), VBScript, JavaScript, etc.

Para poder utilizar esta implementación del DOM es necesario disponer del analizador MS XML, que corresponde al archivo msxml.dll. Este archivo se instala junto con el Microsoft internet Explorer 5. Sin embargo, es aconsejable visitar el sitio web de Microsoft para obtener la última versión disponible.

En este capítulo se describen las características generales del DOM Y las distintas interfaces que componen la implementación de Microsoft. Cada interfaz ofrece acceso a las propiedades y métodos de una clase (por ejemplo, las clases documento, nodo, lista de nodos, etc.). La documentación de Microsoft se ha tomado como base para la descripción de las interfaces, propiedades y métodos.

La descripción de métodos y propiedades acompaña en algunos casos de ejemplos escritas en el lenguaje VBA. Este lenguaje es una versión de visual Basic que se te lisa para desarrollar aplicaciones de Microsoft Office. La sintaxis VBA es idéntica a la de visual Basic.

El propósito de este capítulo es introducir el DOM y servir como referencia los distintos objetos, propiedades y métodos. Espero encontrar ejemplos detalladas de cómo utilizar el DOM para XML desde Visual Basic y JavaScript en el siguiente capítulo.

Características generales del DOM.

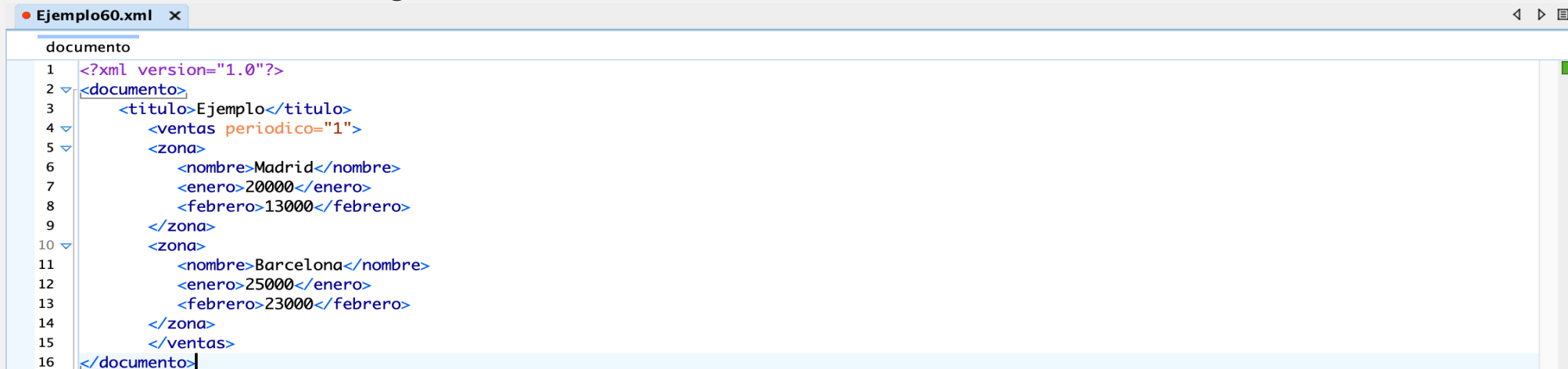
- El DOM representa un documento XML como un árbol formado por nosotros.

Estamos acostumbrados a representar documentos XML en forma de árboles para mostrar las relaciones de dependencia entre sus elementos.

Sin embargo, el árbol del DOM no representa la relación jerárquica entre los elementos de un tipo de documento determinado. Perdon representa la relación entre elementos, atributos, entidades, etc. Es una instancia de un tipo de documento específico.

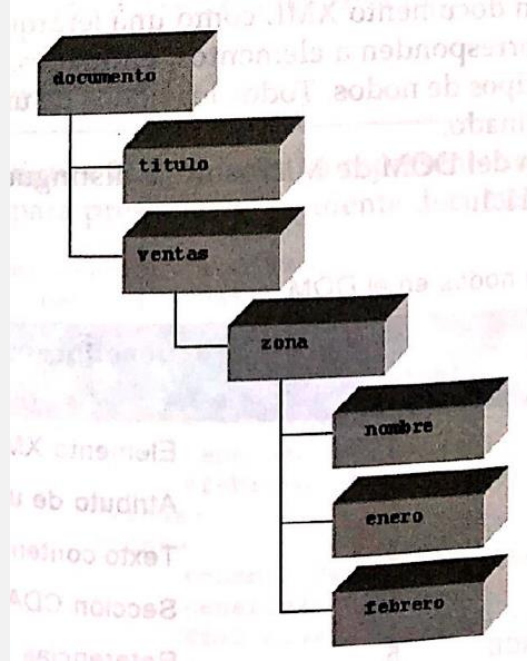
Ejemplo.

Tomamos como base el siguiente documento.



```
1 <?xml version="1.0"?>
2 <documento>
3   <titulo>Ejemplo</titulo>
4   <ventas periodo="1">
5     <zona>
6       <nombre>Madrid</nombre>
7       <enero>20000</enero>
8       <febrero>13000</febrero>
9     </zona>
10    <zona>
11      <nombre>Barcelona</nombre>
12      <enero>25000</enero>
13      <febrero>23000</febrero>
14    </zona>
15  </ventas>
16 </documento>
```

Normalmente utilizamos un árbol para representar la estructura de los elementos de este documento, tal y como se ha definido en su DTD O esquema, de forma similar a la que muestra la figura 11.1. Éste gráfico representa la relación entre los elementos declarados en la DTD Y qué elementos pueden ser hijos de otros.



El árbol utilizado por el DOM no se representan los elementos sino las ocurrencias o instancias de estos, junto con los comentarios, entidades, referencias a entidades, atributos, anotaciones y el contenido textual de los elementos. Se utiliza el término nodo para hacer referencia a todos estos componentes. De esta forma, un documento XML se presentará como un conjunto de nodos relacionados entre sí.

Los nodos se organizan jerárquicamente, de forma que unos dependen de otros y todos a su vez dependen de uno de raíz. La jerarquía de no me refleja las relaciones jerárquicas entre los elementos del documento. De esta manera, sin el documento XML el elemento autor es un elemento hijo del elemento del libro, en el árbol del DOM encontraremos esta misma relación, pero esta vez entre las ocurrencias de los elementos libros y autor.

Sin embargo, como en el DOM se trata también como nodos el contenido textual de los elementos, los comentarios, etc. Existen otras relaciones jerárquicas además de la relación entre elementos padres e hijos.

Por ejemplo

- Una sección `CDATA` incluso en el texto de un documento será un modo hijo del nodo correspondiente a ese elemento
- El contenido textual de un elemento será un nodo hijo del nodo correspondiente a esa instancia del elemento, etc.

De esta forma, el DOM ofrece una representación en forma de árbol complejo, a través de la cual podemos acceder a la estructura de un documento XML y a toda la información que este contiene.

TIPOS DE NODOS EN EL DOM

El DOM interpreta un documento XML como una jerarquía de nodos. Para diferenciar que no nos corresponden a elementos, entidades, comentarios, etc. El DOM distingue entre tipos de nodos. Todos los nodos de un documento XML serán de un tipo determinado.

En la implementación del DOM de Microsoft se distinguen los tipos de nodos que recoge la tabla 11.1.

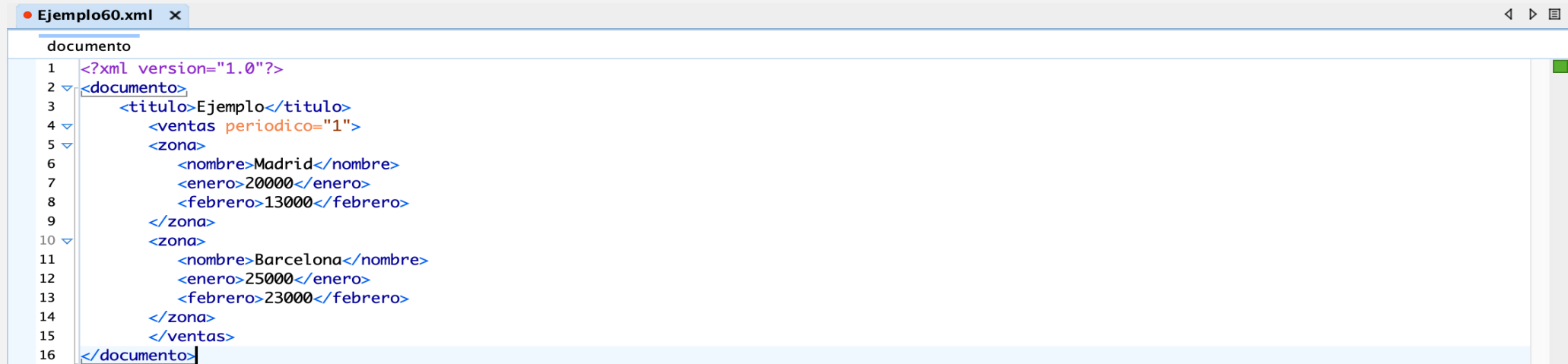
Tipo de nodo	Código numérico	Descripción
NODE_ELEMENT	1	Elemento XML
NODE_ATTRIBUTE	2	Atributo de un elemento
NODE_TEXT	3	Texto contenido en un elemento
NODE_CDATA_SECTION	4	Sección CDATA
NODE_ENTITY_REFERENCE	5	Referencias a entidades
NODE_ENTITY	6	Declaración de entidad
NODE_PROCESSING_INSTRUCTION	7	Instrucción de procesamiento, por ejemplo, la utilizada para asociar un documento a una hoja de estilo
NODE_COMMENT	8	Comentario
NODE_DOCUMENT	9	Nodo documento Es el nodo raíz de la presentación Dom
NODE_DOCUMENT_TYPE	10	DTD indicada en la declaración de tipo de documento del documento XML
NODE_DOCUMENT_FRAGMENT	11	Fragmento de un documento
NODE_NOTATION	12	Notation

Como podemos ver, todos los objetos o componentes que podemos encontrar en un documento XML son tratados como nodos. Su tipo será uno de los indicados en esta tabla.

Cada tipo de nodo u objeto contará con propiedades y métodos específicos, si bien todos los nodos comparten unas propiedades y métodos comunes.

A nivel de desarrollo la implementación del DOM incorporar los elementos necesarios para leer y establecer el tipo de un nodo. Cuando usamos estos métodos podemos hacer referencia del tipo de nada con la constante indicada en la primera columna o con el identificador numérico que recoja la segunda columna.

EL ÁRBOL DEL DOM



Como se puede ver en la figura, el DOM estructura de los documentos XML en una jerarquía de nodos. Todos los nodos dependen de uno de la raíz de tipo documento. Éste no debe de confundirse con el elemento raíz del documento. El elemento raíz se representa por otro nodo de tipo node Element, El cual será un nodo hijo perteneciente al nuevo documento.

Los atributos de estos elementos también se representan mediante nodos. El contenido del texto de las ocurrencias de atributos y elementos es también nodo. Por ejemplo, en el documento anterior, el texto Barcelona es un nuevo hijo del nodo correspondiente a la segunda ocurrencia del elemento nombre.

RELACIONES VÁLIDAS ENTRE NODOS.

El DOM establece una relación jerárquica entre los nodos con los que se representan los componentes del documento XML. Se trata de relaciones padre-hijo entre nodos, De forma que un nodo será hijo de otro nodo.

Debemos conocer que relaciones son válidas entre distintos tipos de nodo. Así un nodo podrá ser hijo de otro modo, si es de un tipo determinado y cumple las restricciones indicadas en la tabla 11.2

NODE_ELEMENT	NODE_ELEMENT NODE_TEXT NODE_COMENT NODE_CDATASECTION NODE_ENTITYREFERENCE	NODE_Document NODE_DocumentFragment NODE_EntityReference NODE_Element	NODE_PROCESING_INSTRUCTION	Ninguno	NODE_Document NODE_DocumentFragment NODE_EntityReference NODE_Element
NODE_ATTRIBUTE	NODE_TEXT NODE_ENTITYREFERENCE	Ninguno		Ninguno	NODE_Document NODE_DocumentFragment NODE_EntityReference NODE_Element
NODE_TEXT	Ninguno	NODE_Attribute NODE_DocumentFragment NODE_Element NODE_EntityReference			
NODE_CDATA_SECCION	Ninguno	NODE_DocumentFragment NODE_EntityReference NODE_Element	NODE_DOCUMENT	NODE_Element NODE_Processing Instruction NODE_Coment NODE_DocumentType	Ninguno
NODE_ENTITY_REFERENCE	NODE_Element NODE_Processing Instruction NODE_Coment NODE_Text NODE_CDATASection NODE_EntityReference	NODE_Attribute NODE_DocumentFragment NODE_Element NODE_EntityReference	NODE_DOCUMENT_TYPE	NODE_Notation NODE_Entity	NODE_Document
NODE_ENTITY	NODE_Text NODE_EntityReference	NODE_DOCUMENT_TYPE	NODE_DOCUMENT_FRAGMENT	NODE_Element NODE_Processing Instruction NODE_Coment NODE_Text NODE_CDATASection NODE_EntityReference	Ninguno
			NODE_NOTATION	Ninguno	NODE_DOCUMENT_TYPE

OBJETOS, PROPIEDADES Y MÉTODOS.

El DOM ofrece una interfaz orientada a objetos para acceder y procesar documentos XML. La orientación a objetos es un paradigma de la programación en el que las entidades se representan mediante objetos. En el caso del DOM, los documentos XML, nodos, elementos, atributos, etc. corresponden a objetos.

A su vez, para cada objeto contamos con propiedades y métodos.

Las propiedades contienen información sobre el objeto. Por ejemplo, para un objeto nodo tendremos propiedades que recojan su nombre, tipo, el nombre de su nodo padre, etc. El valor de las propiedades se puede leer, o leer y escribir. Algunas propiedades serán de sólo escritura y otras de sólo lectura y escritura.

Los métodos corresponden a las acciones que se pueden realizar con un objeto. Por ejemplo, a un objeto de tipo nodo se le puede añadir o eliminar un nodo hijo, renombrarlo, etc. Para cada una de estas acciones tendremos un método. Un método no contiene ningún valor. Sin embargo, un método puede recibir parámetros. Los parámetros son valores que uno no puede recibir para realizar la acción que implementa.

En el momento en el que se ejecuta un método, se le deben pasar los parámetros que se hayan especificada en su definición.