UT5. XML

UT5. XML

5.1 Introducció
5.2 Documentos XML
5.3. Estructura jerárquica de un documento XML
5.4. Modelo de datos de un documento XML. Nodos
5.5. Corrección sintáctica: documento XML bien formado
5.6 Documentos XML válidos
5.7. Validación de documentos XML con DTD
5.7.1. Estructura de un DTD. Elementos

5.1 Introducció

En este tema se van a tratar, de manera casi exclusiva, aspectos relativos a una tecnología aparecida hace ya algunos años: el **XML**. Se trata de un formato de almacenamiento de información a base de etiquetas o marcas definidas por el usuario.

Por el hecho de ser un lenguaje de marcas, deberá cumplir una serie de reglas que harán que **un documento XML esté bien formado**: la forma en la que se abren y cierran las etiquetas, la forma en la que se escriben sus atributos, la existencia de un elemento que los contenga a todos, la aparición de comentarios y su formato, etc.

5.1 Introducció

Adicionalmente, se aprenderán mecanismos para **validar un documento XML**. Esto significa "crear" un lenguaje de marcas para un uso específico, ya que se indica qué elementos y atributos pueden aparecer (vocabulario), el orden en el que aparecen, qué elemento contiene a cual, qué atributos tiene un elemento, qué elementos o atributos son optativos y cuáles son obligatorios, etc.

Aquí se revisarán fundamentalmente dos técnicas: los **DTD** y los **esquemas XML**. La primera es una técnica algo obsoleta pero muy extendida. La segunda, que es una técnica más sofisticada que permite unos niveles de definición mucho más precisos.

5.2 Documentos XML

El **XML** (e**X**tensible **M**arkup **L**anguage - Lenguaje de Marcado eXtensible) es un estándar (o norma), no una implementación concreta.

Es un **metalenguaje de marcas**, lo que significa que no dispone de un conjunto fijo de etiquetas (como sucedía con HTML) que todo el mundo debe conocer. Por el contrario, XML permite definir a los desarrolladores los elementos que necesiten y con la estructura que mejor les convenga.

Define una sintaxis general para maquetar datos con etiquetas sencillas y comprensibles al ojo humano (a cualquier humano). Provee, así mismo, un formato estándar para documentos informáticos. Es un formato flexible, de manera que puede ser adaptado al campo de aplicación que se desee.

5.2 Documentos XML

Ejemplo:

En el campo de la **química**, tendría sentido la existencia de etiquetas como **<átomo>**, **<molécula>** o **<enlace>**.

En el campo de la composición musical, tendría sentido que hubiera etiquetas como <entera>, <negra> O <semicorchea>.

De ahí viene el extensible del nombre XML: se pueden crear nuevas etiquetas en función de las necesidades.

5.2 Documentos XML

Lo que no es XML

- XML **no es un lenguaje de programación**, de manera que no existen compiladores de XML que generen ejecutables a partir de un documento XML.
- XML **no es un protocolo de comunicación**, de manera que no enviará datos por nosotros a través de internet (como tampoco lo hace HTML). Protocolos de comunicación son HTTP (Hyper-Text Transfer Protocol- Protocolo de Transferencia de Hipertexto), FTP (File Transfer Protocol- Protocolo de Transferencia de Ficheros), etc. Estos y otros protocolos de comunicación pueden enviar documentos con formato XML.
- XML **no es un sistema gestor de bases de datos**. Una base de datos relacional puede contener campos del tipo XML. Existen, incluso, bases de datos XML nativas, que todo lo que almacenan son documentos con formato XML. Pero XML en sí mismo no es una base de datos.
- **No es propietario**, es decir, no pertenece a ninguna compañía, como sucede con otros formatos.

5.2 Documentos XML

Formato adecuado para el almacenamiento y la transmisión

Puesto que es un **formato de texto plano** (no se trata de archivos binarios) es adecuado para almacenar información y transmitirla. Con el más simple editor de textos se puede editar un documento XML.

Asimismo, los documentos XML son relativamente ligeros para ser almacenados y enviados, puesto que ocupan lo que los datos que contienen más las etiquetas que los delimitan. Suelen tener extensión. xml, aunque no es imprescindible.

Estas etiquetas (y sus atributos) son meta-información, es decir, información sobre la información (sobre los datos). Nos permiten estructurar el documento y facilitan su procesamiento, pero no son información en sí mismas.

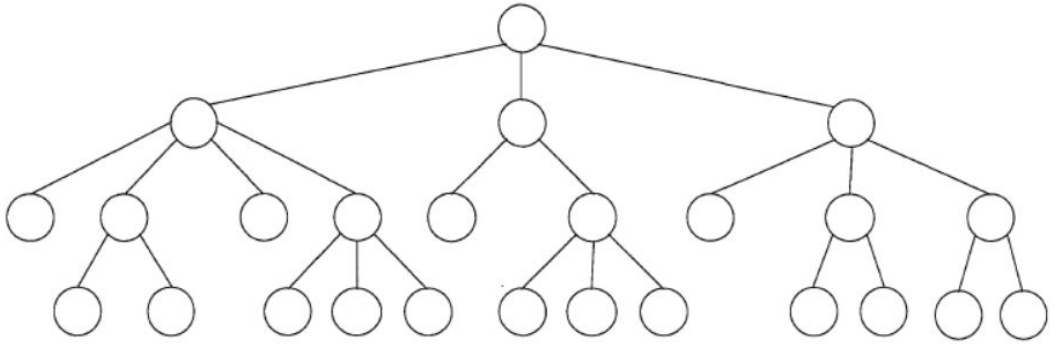
5.3. Estructura jerárquica de un documento XML

En un **documento XML** la información se organiza de forma **jerárquica**, de manera que los elementos del documento se relacionan entre sí mediante relaciones de **padres**, **hijos**, **hermanos**, **ascendentes**, **descendentes**, etc.

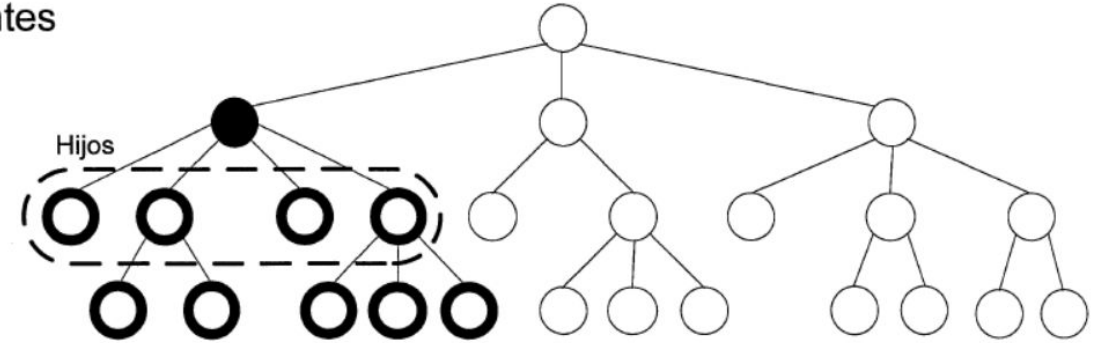
A esta estructura jerárquica se la denomina **árbol del documento XML.** A las partes del árbol que tienen hijos se las denomina **nodos intermedios o ramas**, mientras que a las que no tienen se conocen como **nodos finales u hojas.**

5.3. Estructura jerárquica de un documento XML

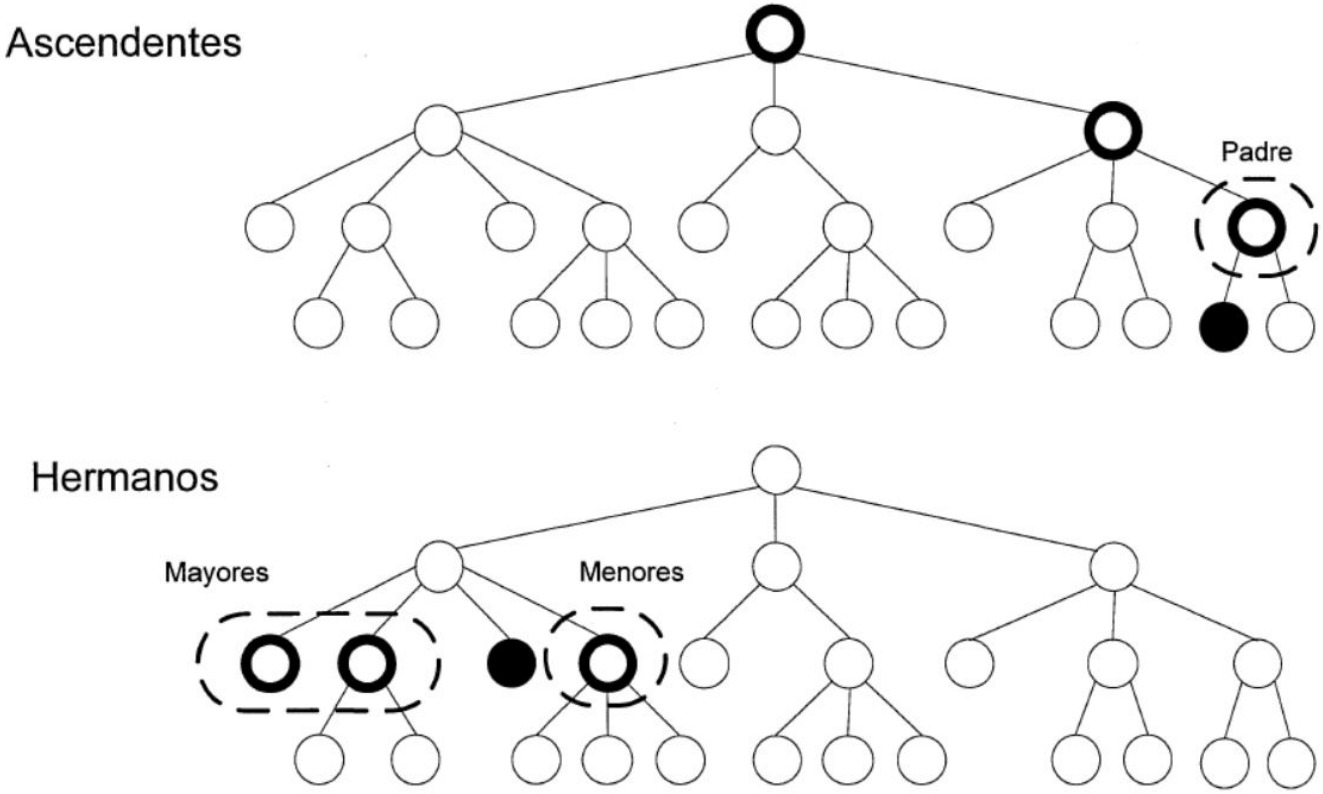
Original



Descendentes



5.3. Estructura jerárquica de un documento XML



5.3. Estructura jerárquica de un documento XML

En un **documento XML** la información se organiza de forma **jerárquica**, de manera que los elementos del documento se relacionan entre sí mediante relaciones de **padres**, **hijos**, **hermanos**, **ascendentes**, **descendentes**, etc.

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

5.3. Estructura jerárquica de un documento XML

Ejemplo: El elemento <persona> es "padre" (contiene) a los elementos <nombre> y <apellido>, que son "hermanos" entre sí.

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

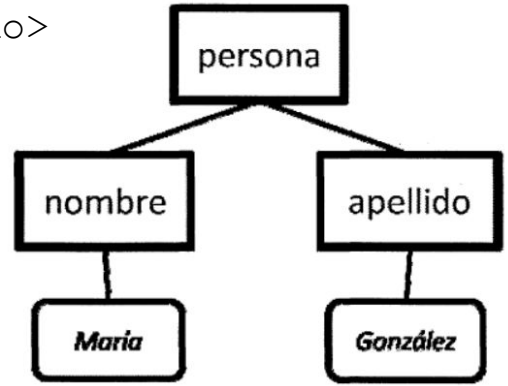


diagrama de árbol

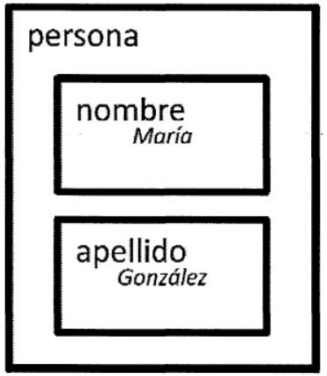


diagrama de cajas

5.3. Estructura jerárquica de un documento XML

Visualización de un documento XML

No dispone de una visualización concreta en un navegador puesto que el documento no refleja una apariencia, sino unos datos. Existen varias maneras para representar visualmente los datos de un documento XML:

- Una forma es mediante una hoja de estilo CSS que indique al navegador cómo convertir cada elemento del documento XML en un elemento visual. Se lograría con la siguiente instrucción de procesamiento que se verá más adelante:
    - <?xml-stylesheet type="text/css" href="estilos.css"?>
- Otra manera es mediante el uso de una hoja de transformaciones XSLT. La instrucción equivalente para asociar a un documento XML una hoja de transformaciones XSLT es:
    - <?xml-stylesheet type="text/xsl" href="transforma.xsl"?>
- También se podría hacer mediante el uso de un lenguaje de programación, como Java o JavaScript, que procese el documento XML.

5.4. Modelo de datos de un documento XML. Nodos

Como se ha comentado, un documento XML consta de una determinada estructura, formada por los siguientes tipos de componentes o nodos:

- **Raíz:** Por encima de cualquier elemento se ubica el nodo raíz, que se designa como "/". No es un componente que tenga representación dentro del documento XML, pero se utilizará más adelante como punto de partida para recorrer el árbol XML y ubicar el resto de nodos.

- **Elementos:** Es la unidad básica de un documento XML. Son delimitadores/contenedores de información.

Al igual que los elementos de HTML, se identifican por una etiqueta de apertura (como <persona») y una de cierre (como </persona»). Lo que se ubica entre ambas es el contenido de ese elemento, que puede ser textual, otros elementos o vacío.

5.4. Modelo de datos de un documento XML. Nodos

<persona> ... </persona>

Apertura Contenido Cierre

Algunos tipos de elementos especiales son:

- **Elemento raíz:** todo documento XML bien formado debe contener un único elemento raíz que contiene a todos los demás (no tiene ascendentes ni hermanos). También se le llama elemento documento.
- **Elementos sin contenido:** aunque puede tener atributos, se abre y se cierra con una sola etiqueta.

Ejemplo:

Elemento sin contenido y sin atributos. `< separador />`

Elemento sin contenido pero con atributos. `<separador cantidad=" 7" />`

5.4. Modelo de datos de un documento XML. Nodos

Atributos: Son como los atributos de HTML. Son pares nombre-valor que permiten especificar datos adicionales de un elemento. Se ubican en la etiqueta de apertura del elemento. Para asignar un valor a un atributo se utiliza el signo igual. Todos los atributos, independientemente del tipo de datos que representen, se tratarán como texto y aparecerán entre comillas simples o dobles.

Se usan fundamentalmente para almacenar información sobre la información (metainformación) contenida en el documento.

Ejemplo: `<distancia unidades="km">70</distancia>`

También se emplean para recoger información identificativa del elemento que permita distinguirlo de otro elemento.

Ejemplo: `<persona nif="12345678Z"> ... </persona>`

5.4. Modelo de datos de un documento XML. Nodos

Texto: El texto, como tipo de nodo que representa los datos del documento XML, puede aparecer, bien como contenido de un elemento, bien como valor de un atributo. No puede aparecer en ningún otro lugar.

Un tratamiento especial tienen los espacios en blanco. Hay cuatro tipos de caracteres de espaciado en XML:

Tabulador:	\t		TAB
Nueva línea:	\n
LF
Retomo de carro:	\r	&#13 ;	CR
Espacio:	\s		NO-BREAK SPACE

Dentro del contenido textual de un elemento se mantendrán como están y así serán tratados por el procesador. Como valor de un atributo, los espacios en blanco adyacentes se condensan en uno solo. Los espacios en blanco entre elementos serán ignorados.

5.4. Modelo de datos de un documento XML. Nodos

Ejemplo:

<A> Dato </A>	≠	<A>Dato</A>
<A b="otro dato" />	=	<A b="otro dato" />
<A>Más datos</A> <B>Y más</B>	=	<A>Más datos</A><B>Y más</B>

Comentarios: Son iguales que los de HTML. Empiezan por los caracteres <!-- y se cierran con los caracteres -->. Dentro de ellos se puede escribir cualquier signo (sin necesidad de caracteres de escape) menos el doble guión (--) que confundiría al analizador con un posible cierre del comentario.

Pueden ubicarse en cualquier lugar de un documento, menos dentro de una etiqueta de apertura, ni dentro de una etiqueta de cierre.

5.4. Modelo de datos de un documento XML. Nodos

Espacio de nombres: Es un mecanismo para distinguir etiquetas cuando se mezclan distintos vocabularios. Se verán en detalle más adelante.

Instrucciones de procesamiento: Las instrucciones de procesamiento empiezan por <? Y terminan por ?>. Son instrucciones para el procesador XML, de manera que son dependientes de él. No forman parte del contenido del documento XML. Se utilizan para dar información a las aplicaciones que procesan el documento XML.

Ejemplo: <?xml versi6n="1.0" encoding="UTF-8"?>

5.4. Modelo de datos de un documento XML. Nodos

Entidades predefinidas: Representan caracteres especiales de marcado, pero son interpretados como texto por parte del procesador XML.

Entidad	Carácter
&amp;	&
&lt;	<
&gt;	>
&apos;	‘
&quot;	“

Ejemplo: `<libreria>Barnes &amp; Noble</libreria>`

El contenido del elemento <libreria> será interpretado por el analizador como Barnes & Noble.

5.4. Modelo de datos de un documento XML. Nodos

Secciones CDATA: Son conjuntos de caracteres que el procesador no debe analizar (parecidos a un comentario).

La definición de estas secciones permite agilizar el análisis del documento y deja libertad al autor del documento para introducir libremente en ellas caracteres como < y &.

No pueden aparecer antes del elemento raíz ni después de su cierre. No pueden contener el propio signo delimitador de final de sección CDATA, es decir, la combinación de caracteres]]>

Ejemplo: Se quiere introducir como contenido de un elemento <codigo> un trozo de código HTML y se desea que el analizador XML no 10 tome como etiquetas, sino que 10 deje sin procesar.

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

5.4. Modelo de datos de un documento XML. Nodos

Definición de Tipo de Documento (DTD): Permite definir reglas que fuercen ciertas restricciones sobre la estructura de un documento XML aunque su existencia no es obligatoria. Un documento XML bien formado que tiene asociado un documento de declaración de tipos y cumple con las restricciones allí declaradas se dice que es válido. Más adelante se verán en profundidad éste y otros mecanismos de validación de documentos XML.

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

5.4. Modelo de datos de un documento XML. Nodos

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?> ❶
<!DOCTYPE document system "elementos.dtd"> ❷
<!-- Aquí viene un comentario --> ❸
<?xml-stylesheet type="text/css" href="elementos.css"?> ❹
<elementos> ❺
  <elemento> ❻
    <nombre>Agua</nombre>
    <color>Azul</color>
  </elemento>
  <elemento> ❻
    <nombre>Fuego</nombre>
    <color>Rojo</color>
  </elemento>
</elementos> ❼
```

❶ Instrucción de procesamiento que declara que el documento está en formato XML.

❷ Inclusión de un DTD externo, ubicado en el archivo elementos.dtd.

❸ Un comentario.

❹ Instrucción de procesamiento que vincula al documento XML una hoja de estilos ubicada en el archivo elementos.css.

❺ Elemento raíz (también llamado elemento documento). En este punto se abre.

❻ Diversos elementos descendientes del elemento raíz y los descendientes de éstos.

❼ Cierre del elemento raíz.

5.4. Modelo de datos de un documento XML. Nodos

Nombres XML

En XML se utilizan una serie de reglas para definir nombres correctos de elementos. Son las mismas que para los atributos.

Un nombre XML, que así se denomina:

- Puede empezar con una letra (con o sin tilde), que podría ser de un alfabeto no latino, subrayado o dos puntos (este último carácter es desaconsejado ya que se reserva su uso para los espacios de nombres).
- Los siguientes caracteres pueden ser letras, dígitos, subrayados, guiones bajos, comas y dos puntos.
- Los nombres que empiezan por las letras XML, en cualquier combinación de mayúsculas y minúsculas, se reservan para estandarización.
- No pueden contener:
    - Ningún carácter de espaciado.
    - Ningún otro carácter de puntuación que los ya citados como válidos. Esto incluye: comillas simples o dobles, signo de dólar, acento circunflejo, signo de porcentaje, punto y coma.

5.4. Modelo de datos de un documento XML. Nodos

Ejemplo:

Nombres correctos:

```
<Número_Seguridad_Social>50-12345678</Número_Seguridad_Social>
<primerApellido>Rodríguez</primerApellido>
<_cuenta_Tweeter>@follower</_cuenta_Tweeter>
```

Nombres incorrectos:

```
<O'Donnell>General</O'Donnell>  ❶
<día/mes/año></día/mes/año>  ❷
<fecha nacimiento>2011-02-02 </fecha nacimiento>  ❸
```

❶ No es válido por el carácter apóstrofe

❷ No es válido por el signo de dividir

❸ No es válido por el espacio

5.4. Modelo de datos de un documento XML. Nodos

Uso de elementos frente a uso de atributos

- Los elementos:
  - Se emplean para representar jerarquías o contenido de unos dentro de otros.
  - Se pueden extender con otros elementos en su interior.
  - El orden en el que aparecen es representativo.
  - Pueden tener atributos.
  - Puede haber múltiples ocurrencias de un elemento.
- Los atributos:
  - Van asociados a los elementos.
  - Son modificadores de la información.
  - Se suelen usar para registrar metadatos.
  - El orden en que aparecen dentro del elemento al que van asociados no es representativo.
  - No se pueden extender con otros elementos contenidos en su interior.
  - No puede haber múltiples ocurrencias de un atributo dentro de un mismo elemento.

5.4. Modelo de datos de un documento XML. Nodos

Espacios de nombres

Es un mecanismo para evitar conflictos de nombres, de manera que se puedan diferenciar elementos o atributos dentro de un mismo documento XML que tengan idénticos nombres pero diferentes definiciones. No aparecieron en la primera especificación de XML, pero sí pronto después.

Se declaran como atributo de elementos de la forma:

`<nombre_elemento xmlns:prefijo="URI_del_espacio_de nombres">`

y se usan anteponiendo a elementos y atributos con el prefijo asociado al espacio de nombres además del carácter dos puntos ":".

Ejemplo:

```
<info:pedido xmlns:info="empresa:espacios:info">
  <info:item info:id="i_13">Afeitadora eléctrica</info:item>
  ...  …
</info:pedido>
```

5.4. Modelo de datos de un documento XML. Nodos

Cualquier cadena de texto puede ser usada como prefijo del espacio de nombres. El URI del espacio de nombres sí debe ser único, aunque realmente no se comprueba mediante conexión alguna. El URI no es más que un nombre lógico del espacio de nombres. A veces se usa como URI http://~.

El ámbito de declaración o uso de un espacio de nombres, incluye los lugares donde se puede referenciar este espacio de nombres. Cubre el elemento donde se ha declarado y sus elementos descendientes. A todos estos elementos se les puede anteponer el prefijo del espacio de nombres.

Asimismo, se puede declarar un espacio de nombres diferente para un elemento descendiente de otro, en el cual ya se declarado otro espacio de nombres.

5.4. Modelo de datos de un documento XML. Nodos

Ejemplo:
En un documento XML se declaran dos espacios de nombres, `empresa:espacios:dept` y `empresa:espacios:emp` , de prefijos dept y emp respectivamente.

```
<dept:departamentos xmlns:dept="empresa:espacios:dept"
                    xmlns:emp="empresa:espacios:emp">
  <dept:departamento dept:deptno="10">
    <emp:empleado emp:empno="7654">
      <emp:nombre>Roberto</sec:nombre>
      <emp:apellido>Mate</emp:apellido>
    </emp:empleado>
    <emp:empleado emp:empno="7891">
      <emp:nombre>Lucía</emp:nombre>
      <emp:apellido>Palmito</emp:apellido>
    </emp:empleado>
  </dept:departamento>
</dept:departamentos>
```

5.4. Modelo de datos de un documento XML. Nodos

Los atributos pueden pertenecer al mismo espacio de nombres al que pertenece el elemento en el que están asociados o a otro diferente. La definición es la misma que con los elementos, anteponiendo el prefijo del espacio de nombres y los dos puntos al nombre el atributo. Si no aparece ningún prefijo de espacio de nombres, el atributo no pertenecerá a ningún espacio de nombres.

```
<emp:empleado xmlns:emp="empresa:espacios:emp">
  <emp:datosPersonales trabajo:empno="7583"
          xmlns:trabajo="empresa:espacios:trabajo">
    <emp:departamento deptno="10">Ventas</emp:departamento>
    <emp:nombre>Roberto</emp:nombre>
    <emp:apellido>López</emp:apellido>
  </emp:datosPersonales>
</emp:empleado>
```

5.4. Modelo de datos de un documento XML. Nodos

```
<emp:empleado xmlns:emp="empresa:espacios:emp">
  <emp:datosPersonales trabajo:empno="7583"
          xmlns:trabajo="empresa:espacios:trabajo">
    <emp:departamento deptno="10">Ventas</emp:departamento>
    <emp:nombre>Roberto</emp:nombre>
    <emp:apellido>López</emp:apellido>
  </emp:datosPersonales>
</emp:empleado>
```

Ejemplo:
El atributo trabajo:empno del elemento <emp:datosPersonales> pertenece a un espacio de nombres diferente que éste. El prefijo del espacio de nombres del atributo es trabajo. El atributo deptno del elemento <emp: departamento> no pertenece a ningún espacio de nombres.

5.4. Modelo de datos de un documento XML. Nodos

Dos atributos con el mismo nombre pero con distinto prefijo de espacio de nombres son diferentes, lo que significa que pueden estar asociados al mismo elemento.

Ejemplo: Los atributos trabajo:empno y emp:empno son diferentes, luego pueden aparecer asociados al mismo elemento.

```
<emp:empleado xmlns:emp="empresa:espacios:emp"
              xmlns:trabajo="empresa:espacios:trabajo">
  <emp:datosPersonales trabajo:empno="7583" emp:empno="e_7583">
    <emp:departamento deptno="10">Ventas</emp:departamento>
    <emp:nombre>Mercedes</emp:nombre>
    <emp:apellido>López</emp:apellido>
  </emp:datosPersonales>
</emp:empleado>
```

5.4. Modelo de datos de un documento XML. Nodos

Actividad

Indica a qué espacio de nombres pertenece el elemento `<info:venta>` del siguiente documento XML y justifica tu respuesta.

```
<?xml version="1.0" ?>
<info:venta xmlns:info="empresa:espacios:info"
                xmlns:prod="empresa:espacios:prod">
  <prod:producto prod:id="p_987">Mesa de despacho</prod:producto>
  <prod:precio prod:mondeda="Euro">300</prod:precio>
  <prod:unidades>200</prod:unidades>
</info:venta>
```

a. A `empresa:espacios:info`

b. A `empresa:espacios:prod`

c. A `empresa:espacios:info` y `empresa:espacios:prod`

d. A ningún espacio de nombres

5.4. Modelo de datos de un documento XML. Nodos

Espacio de nombres por defecto

Un espacio de nombres por defecto es aquel en el que no se define un prefijo. El ámbito de aplicación de ese espacio de nombres es el del elemento en el que se ha declarado y sus elementos descendientes, **pero no a sus atributos**.

Si se tuviera que añadir un espacio de nombres con prefijo a un documento XML grande ya creado, habría que ir añadiendo el prefijo a los elementos y atributos, algo tedioso y tendente a producir errores. De esta manera, se puede evitar tener que escribir el prefijo.

El mayor inconveniente de esta medida, citado ya antes, es que el espacio de nombres por defecto sólo afecta al elemento en el que está declarado y a sus descendientes, no a los atributos.

5.4. Modelo de datos de un documento XML. Nodos

Ejemplo: El espacio de nombres por defecto, empresa:espacios:emp, no afectará al atributo empno del elemento <datosPersonales> ni al atributo deptno del elemento <departamento>.

```
<empleado xmlns="empresa:espacios:emp"
                xmlns:trabajo="empresa:espacios:trabajo">
  <datosPersonales trabajo:empno="7583" empno="e_7583">
    <departamento deptno="10">Ventas</departamento>
    <nombre>Mercedes</nombre>
    <apellido>López</apellido>
  </datosPersonales>
</empleado>
```

Se puede desasignar a un elemento de un espacio de nombres por defecto con la orden:

<nombre elemento xmlns="">

5.4. Modelo de datos de un documento XML. Nodos

Actividad

Indica a qué espacio de nombres pertenece el atributo `moneda` del elemento `<precio>` del siguiente documento XML y justifica tu respuesta.

```
<?xml version="1.0" ?>
<venta xmlns="empresa:espacios:info"
          xmlns:venta="empresa:espacios:venta">
    <producto id="p_987">Mesa de despacho</producto>
    <precio moneda="Euro">300</precio>
    <unidades>200</unidades>
</venta>
```

a. A empresa:espacios:info

b. A `empresa:espacios:venta`

c. A `empresa:espacios:info` y `empresa:espacios:venta`

d. A ningún espacio de nombres

5.4. Modelo de datos de un documento XML. Nodos

Atributos especiales

Hay algunos atributos especiales en XML:

`xml:space`: le indica a la aplicación que usa el XML si los espacios en blanco del contenido textual de un elemento son significativos.

Ejemplo: Se le indica al XML que los espacios se mantengan (preserve) o se eliminen (default):

```
<!ATTLIST quitaEspacios xml:space #FIXED "default">
<!ATTLIST dejaEspacios xml:space #FIXED "preserve">
```

5.4. Modelo de datos de un documento XML. Nodos

`xml:lang`: permite especificar el idioma en el que está escrito el contenido textual de todo un documento o de sus elementos individuales. Los posibles valores son:

- un código de dos letras (ISO 639). Ejemplo: en para English (existen subcódigos, como en-GB y en-US para inglés británico y americano respectivamente).
- Un identificador de idioma registrado por el IANA (Internet Assigned Numbers Authority - Autoridad de Números Asignados en Internet). Estos identificadores empiezan por el prefijo i- o I-.
- Un identificador definido por el usuario. Estos identificadores empiezan por el prefijo x- o X--.

`:base`: permite definir una URI distinta a la del documento.

5.4. Modelo de datos de un documento XML. Nodos

ParserXML

Un analizador XML (llamado en inglés parser), es un procesador que lee un documento XML y determina la estructura y propiedades de los datos en él contenidos. Un analizador estándar lee el documento XML y genera el árbol jerárquico asociado, lo que permite ver los datos en un navegador o ser tratados por cualquier aplicación.

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

Estos analizadores también comprueban la semántica del documento e informan de los errores existentes.

Existen validadores XML en línea, como XML Validation (http://www.xmlvalidation.com).

5.5. Corrección sintáctica: documento XML bien formado

La especificación de XML define la sintaxis que el lenguaje debe seguir:

- Cómo se delimitan los elementos con etiquetas.
- Qué formato puede tener una etiqueta.
- Qué nombres son aceptables para los elementos.
- Dónde se colocan los atributos.

5.5. Corrección sintáctica: documento XML bien formado

Un documento XML se dice que está **bien formado** si cumple las reglas establecidas por el W3C en las especificaciones para XML.

- El documento puede (el W3C lo recomienda) empezar por una instrucción de procesamiento xml, que indica la versión del XML y, opcionalmente, la codificación de caracteres (encoding), que por defecto es UTF-8, y si está listo para procesarse independientemente o requiere de otros archivos externos para dicha tarea (standalone), que por defecto vale no.
  La instrucción de procesamiento correcta más simplificada es:

  ```
  <?xml version = "1.0"?>
  ```

  Otra más extensa e igualmente correcta:

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

5.5. Corrección sintáctica: documento XML bien formado

Un documento XML se dice que está **bien formado** si cumple las reglas establecidas por el W3C en las especificaciones para XML.

- El documento puede (el W3C lo recomienda) empezar por una instrucción de procesamiento xml, que indica la versión del XML y, opcionalmente, la codificación de caracteres (encoding), que por defecto es UTF-8, y si está listo para procesarse independientemente o requiere de otros archivos externos para dicha tarea (standalone), que por defecto vale no.

  La instrucción de procesamiento correcta más simplificada es:

  ```
  <?xml version = "1.0"?>
  ```

  Otra más extensa e igualmente correcta:

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

- Debe existir un único elemento raíz, que "cuelga" del nodo raíz (/). Este elemento tendrá como descendientes a todos los demás elementos. El documento XML bien formado más simple contendría sólo un elemento raíz, sin ningún descendiente.

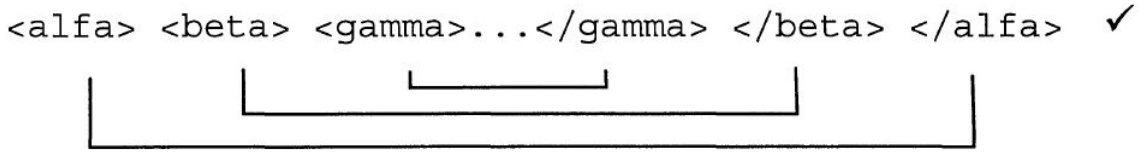
- Los elementos que no sean vacíos deben tener una etiqueta de apertura y otra de cierre.

5.5. Corrección sintáctica: documento XML bien formado

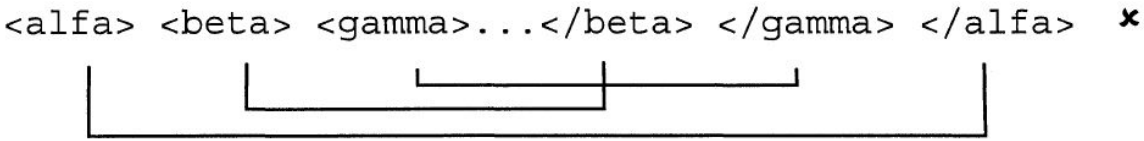
Los elementos vacíos deber cerrarse con />.

Los elementos deben aparecer correctamente anidados en cuanto a su apertura y su cierre, no solaparse. Es decir, los elementos se cierran en orden inverso al que se abren.

Ejemplo: Se abre <alfa>, se abre <beta> y se abra <gamma>. Se cerrarán en orden inverso al de apertura. Primero se cierra </gamma>, luego </beta> y, por último, </alfa>.



En este caso se ha cerrado </beta> antes que </gamma>.



5.5. Corrección sintáctica: documento XML bien formado

- Los nombres de elementos y atributos son sensibles a mayúsculas/minúsculas.
- Los valores de los atributos deben aparecer entre comillas simples o dobles, pero del mismo tipo.

  Ejemplo:

  ```
  <libro signatura="SIL001" />   ✓
  <libro signatura='SIL001" />   ✗
  ```

- No puede haber dos atributos con el mismo nombre asociados al mismo elemento.
- No se pueden introducir ni instrucciones de procesamiento ni comentarios en ningún lugar del interior de las etiquetas de apertura y cierre de los elementos.

  Ejemplo: Un comentario incorrecto:

  ```
  <persona <!-- comentario incorrecto --> >...</persona>   ✗
  ```

5.5. Corrección sintáctica: documento XML bien formado

- No puede haber nada antes de la instrucción de procesamiento <? xml ... ?>
- No puede haber texto antes ni después del elemento documento.
- No pueden aparecer los signos < ni & en el contenido textual de elementos ni atributos.

Una manera de verificar que un documento XML está bien formado es abriéndolo en un navegador como Mozilla Firefox o Internet Explorer. Si se muestra el árbol de nodos, significa que está bien formado.

5.5. Corrección sintáctica: documento XML bien formado

Actividad: Identifica cuáles de los siguientes documentos están bien formados y cuáles no. Cuando no lo estén, explica el (los) motivo(s). Para comprobar el resultado se puede usar un editor XML, como el XML Copy Editor, para que valide por nosotros los documentos.

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

```
<documento>Texto de prueba</documento>
```

```
<?xml?>
<documento>Texto de prueba</documento>
```

```
<?xml version="1.0"?>
<DOCUMENTO/>
```

5.5. Corrección sintáctica: documento XML bien formado

Actividad: Identifica cuáles de los siguientes documentos están bien formados y cuáles no. Cuando no lo estén, explica el (los) motivo(s). Para comprobar el resultado se puede usar un editor XML, como el XML Copy Editor, para que valide por nosotros los documentos.

```
<?XML version="1.0"?>
<Documento codigo="135">
  <nombre>Artículo</nombre>
  <amplitud>Media</amplitud>
</Documento>
```

```
<?xml version="1.0"?>
<El documento>
  <nombre>Artículo</nombre>
  <amplitud>Media</amplitud>
</El documento>
```

5.6 Documentos XML válidos

Hasta ahora se ha visto qué componentes forman un documento XML (elementos, atributos, comentarios ...) y qué reglas deben de cumplir para que el documento sea bien formado, es decir, sea sintácticamente correcto.

Nada se ha dicho sobre qué elementos o atributos concretos pueden aparecer, en qué orden, qué valores pueden tomar, etc.

Si se hiciera esto, se estaría definiendo un lenguaje XML concreto, con su vocabulario (elementos y atributos permitidos), las relaciones entre elementos y atributos, los valores permitidos para ellos, etc.

Se dice que **un documento XML es válido si existen unas reglas de validación** (por ejemplo en forma de definición de tipo de documento (DTD)) **asociadas a él**, de manera que el documento deba cumplir con dichas reglas. Estas reglas especifican la estructura gramatical y sintaxis que debe tener el documento XML.

Todo documento XML válido está bien formado, pero no a la inversa.

5.7. Validación de documentos XML con DTD

Un **DTD** (Document Type Definition - **Definición de Tipo de Documento**) es una descripción de la estructura de un documento XML. Se especifica qué elementos tienen que aparecer, en qué orden, cuáles son optativos, cuáles obligatorios, qué atributos tienen los elementos, etc.

Es un mecanismo de validación de documentos que existía antes de la aparición de XML. Se usaba para validar documentos SGML (Standard Generalizad Markup Language - Lenguaje de Marcas estandar Generalizado), que es el precursor de todos los lenguajes de marcas actuales.

Cuando apareció XML, se integró en su especificación como modelo de validación gramatical.

La técnica de validación con DTDs ha quedado algo obsoleta, si bien se usó de manera intensiva en los años 1990 y principios de los 2000.

Actualmente, se usan más otras técnicas como los **esquemas XML**, también conocidos por las siglas **XSD** (XML Schema Document - Documento de Esquema XML).

5.7.1. Estructura de un DTD. Elementos

Al igual que ya sucedía con las hojas de estilo en cascada (CSS), el DTD puede aparecer integrado en el propio documento XML, en un archivo independiente (habitualmente con extensión .dtd) e incluso puede tener una parte interna y otra externa.

La opción del documento externo permitirá reutilizar el mismo DTD para distintos documentos XML, facilitando las posibles modificaciones de una manera centralizada.

Siempre que se quiera declarar un DTD, se hará al comienzo del documento XML. Las reglas que lo constituyen son las que podrán aparecer a continuación de la declaración (dentro del propio documento XML) o en un archivo independiente.

Declaración del DTD:

```
<!DOCTYPE>
```

Es la instrucción donde se indica qué DTD validará el XML. Aparece al comienzo del documento XML. El primer dato que aparece es el nombre del elemento raíz del documento XML.

5.7.1. Estructura de un DTD. Elementos

En función del tipo de DTD la sintaxis varía. Las características que definen el tipo son:

- Ubicación: dónde se localizan las reglas del DTD.
  - Interno: las reglas aparecen en el propio documento XML.
  - Externo: las reglas aparecen en un archivo independiente.
  - Mixto: mezcla de los anteriores, las reglas aparecen en ambos lugares. Las reglas internas tienen prioridad sobre las externas.
- Carácter: si es un DTD para uso privado o público.
  - o Para uso privado: se identifica por la palabra SYSTEM.
  - o Para uso público: se identifica por la palabra PUBLIC. Debe ir acompañado del FPI (Formal Public Identifier - Identificador Público Formal), una etiqueta que identifica al DTD de manera "universal".

Las distintas combinaciones son:

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

5.7.1. Estructura de un DTD. Elementos

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

Atributos:

- **Nombre del elemento raíz del documento XML**: aparece justo a continuación de la palabra DOCTYPE.

- **Declaración de privacidad/publicidad**: aparece a continuación del nombre del elemento raíz e indica si el DTD es de uso público (PUBLIC) o de uso interno de la organización que 10 desarrolla (SYSTEM).

- **Identificador**: sólo existe cuando el DTD es PUBLIC e indica el FPI por el que se conoce el DTD.

5.7.1. Estructura de un DTD. Elementos

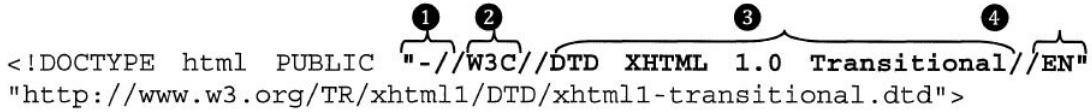
Estructura del FPI:

Está compuesto de 4 campos separados por el carácter //.

- Primer campo: norma formal o no formal
  - Si el DTD no ha sido aprobado por una norma formal, como por ejemplo un DTD escrito por uno mismo, se escribe un signo menos (-).
  - Si ha sido aprobado por un organismo no oficial, se escribe un signo más (+).
  - Si ha sido aprobado por un organismo oficial, se escribe directamente una referencia al estándar.
- Segundo campo: nombre del organismo responsable del estándar.
- Tercer campo: tipo del documento que se describe, suele incluir un número de versión.
- Cuarto campo: idioma del DTD.

5.7.1. Estructura de un DTD. Elementos

Ejemplo: Se va a declarar un DTD respecto a un documento XML cuyo elemento raíz se llama html y que es de carácter público.



❶ Al aparecer un signo menos (-) significa que el DTD no ha sido aprobado por una norma formal.

❷ El nombre del organismo responsable del DTD es el W3C.

❸ El tipo de documento es XHTML Transicional, en su versión 1.0.

❹ El idioma del DTD es el inglés (EN).

Atributos (continuación):

- **url**: sólo existe cuando el DTD (en parte o en su totalidad) se encuentra declarado en un archivo externo, del que da su ubicación. Como ya se ha comentado, puede darse el caso que un DTD esté definido en parte en un archivo externo y en parte en el documento XML.

5.7.1. Estructura de un DTD. Elementos

Ejemplo:

Se declara el tipo de un documento XML cuyo elemento raíz se llama <planetas>.

Se trata de un DTD de uso privado (SYSTEM) y la ubicación de las reglas del DTD es externa, en un archivo llamado planetas.dtd que se encuentra en el mismo directorio que el archivo XML.

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

5.7.1. Estructura de un DTD. Elementos

Ejemplo: Un DTD mixto y privado tendrá sus reglas repartidas entre la cabecera del documento XML y un archivo externo. El documento XML, charlas.xml, será:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<!DOCTYPE charlas SYSTEM "charlas.dtd" [
  <!ELEMENT charlas (charla)+>
  <!ELEMENT charla (nombre, lugar, despedida)>
  <!ENTITY despedidaInglesa "Thank you, good bye!">
  <!ENTITY despedidaFrancesa "Merci, au revoir!">
]>
<charlas>
  ...
</charlas>
```

El archivo de reglas externo, charlas.dtd, será:

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

El archivo externo no incluye la declaración del tipo de documento (DOCTYPE), puesto que sólo aparece en la cabecera del documento XML. Conviene resaltar también que el atributo standalone de la instrucción de procesamientoe <?xml ... ?> tiene el valor no, lo que significa que para el correcto procesado del documento necesitamos del uso de otros documentos externos (en este caso, de charlas.dtd).

5.7.1. Estructura de un DTD. Elementos

Componentes del DTD:

Hay cuatro tipos posibles de componentes que se pueden declarar en un DTD:

- Elemento
- Atributo
- Entidad
- Notación

5.7.1. Estructura de un DTD. Elementos

<!ELEMENT>

Es una declaración de tipo de elemento. Indica la existencia de un elemento en el documento XML.

Sintaxis general: `<!ELEMENT nombre_elemento modelo_contenido>`

El nombre del elemento tiene que ser el mismo que el correspondiente del documento XML, sin los signos de menor (<) y mayor (>) propios de las etiquetas. El modelo de contenido puede indicar:

- Una regla, en cuyo caso será:
    - ANY (cualquier cosa): se puede utilizar al construir el DTD para dejar la descripción de un elemento como válida en cualquier caso, eliminando cualquier comprobación sintáctica. Es un comodín que no debe aparecer en el DTD definitivo.
    - EMPTY (elemento vacío): describe un elemento sin descendientes. Ejemplo: La descripción del elemento <br/> de HTML será: `<!ELEMENT br EMPTY>`

5.7.1. Estructura de un DTD. Elementos

<!ELEMENT>

- Una regla, en cuyo caso será: (continuación)
  - Datos (caracteres), sean textuales, numéricos o cualquier otro formato que no contenga marcas (etiquetas). Se describe como #PCDATA (Parsed Character Data - Datos de Caracteres Procesados) y debe aparecer entre paréntesis.
  - Ejemplo: Una regla de un DTD puede ser:

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

    Que se corresponde con un elemento llamado <titulo> con contenido textual.

    ```
    <titulo>Lenguajes de marcas</titulo>
    ```

5.7.1. Estructura de un DTD. Elementos

<!ELEMENT>

- Una regla, en cuyo caso será: (continuación)
  - Elementos descendientes. Su descripción debe aparecer entre paréntesis y se basa en las siguientes reglas:

Cardinalidad de los elementos: Para indicar el número de veces que puede aparecer un elemento o una secuencia de elementos existen ciertos símbolos:

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

Secuencias de elementos: En las secuencias de elementos se utilizan símbolos para indicar el orden en que un elemento debe aparecer, o bien si aparece como alternativa a otro:

Símbolo	Significado
A, B	El elemento B aparecerá a continuación del elemento A
A | B	Aparecerá el elemento A o el B, pero no ambos

Se pueden combinar el uso de los símbolos de cardinalidad de los elementos con los de las secuencias de elementos.

5.7.1. Estructura de un DTD. Elementos

Ejemplo: En un correo electrónico, se podría describir el elemento raíz <email> como una secuencia de elementos <para>, <cc> (optativo), <cco> (optativo), <asunto> y <cuerpo>. Las reglas que representan esto serán:

```
<!ELEMENT email (para, cc?, cco?, asunto, cuerpo)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT cc (#PCDATA)>
```

Ejemplo: Un contrato tiene una lista de cláusulas. Cada una de las cláusulas está compuesta de varios epígrafes y sus desarrollos asociados, y concluyen por un único epílogo:

```
<!ELEMENT clausulas (clausula)+>
<!ELEMENT clausula ((epígrafe, desarrollo)+, epilogo)>
```

5.7.1. Estructura de un DTD. Elementos

<!ELEMENT>

- Una regla, en cuyo caso será: (continuación)
  - Contenido mixto, mezcla de texto más elementos descendientes.

    Ejemplo: Se quiere describir un elemento <párrafo> que simule el párrafo de un editor de textos, de forma que pueda contener texto sin formato, texto en <negrita> (rodeado de esta etiqueta) o texto en <cursiva> (rodeado de esta etiqueta). Estas dos últimas etiquetas podrán a su vez contener, bien texto sin formato, bien la otra etiqueta. Por tanto, <párrafo>, <negrita> y <cursiva> son elementos de contenido mixto. Las reglas que describen esto serán:

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

    No se puede usar el cuantificador + con un contenido mixto, por eso se ha usado el *. Un documento XML válido con respecto a las reglas del anterior DTD:

```
<parrafo>
  Aquí un <cursiva>tema <negrita>importante</negrita></cursiva>
</parrafo>
```

5.7.1. Estructura de un DTD. Elementos

Actividad

Se quiere que el elemento <grupoSanguineo> tenga como único elemento descendiente a uno solo de los cuatro siguientes A, B, AB ó O. Indica cuál de las siguientes es una declaración correcta del citado elemento.

a. <!ELEMENT grupoSanguineo (A ? B ? AB ? O) >

b. <!ELEMENT grupoSanguineo (A , B , AB , O) >

c. <!ELEMENT grupoSanguineo (A | B | AB | O) >

d. <!ELEMENT grupoSanguineo (A + B + AB + O) >

5.7.1. Estructura de un DTD. Elementos

Actividad

¿Cuál de las siguientes afirmaciones es correcta respecto a la declaración del elemento?

a. <!ELEMENT contenido (alfa | beta*) >
   Tanto el elemento alfa como el elemento beta pueden aparecer 0 o más veces como descendientes del elemento contenido.

b. <!ELEMENT contenido (alfa | beta) >
   Tanto el elemento alfa como el elemento beta pueden aparecer una vez como descendientes del elemento contenido, sea cual sea el orden.

c. <!ELEMENT contenido (alfa | beta) >
   El elemento alfa y el elemento beta deben aparecer una vez cada uno como descendientes del elemento contenido.

d. <!ELEMENT contenido (alfa | beta*) >
   El elemento alfa debe aparecer una vez y a continuación el elemento beta debe aparecer 0 o más veces, ambos como descendientes del elemento contenido.

5.7.1. Estructura de un DTD. Elementos

<!ATTLIST>

Es una declaración de tipo de atributo. Indica la existencia de atributos de un elemento en el documento XML. En general se utiliza un solo ATTLIST para declarar todos los atributos de un elemento (aunque se podría usar un ATTLIST para cada atributo).

Sintaxis general:
```
<!ATTLIST nombre_elemento
                    nombre atributo tipo_atributo caracter
                    nombre atributo tipo_atributo caracter
                    ... >
```

El nombre del atributo tiene que ser un nombre XML válido.

El carácter del atributo puede ser:

- un valor textual entre comillas, que representa un valor por defecto para el atributo.
- #IMPLIED, el atributo es de carácter opcional y no se le asigna ningún valor por defecto.
- #REQUIRED, el atributo es de carácter obligatorio, pero no se le asigna un valor por defecto.
- #FIXED, el atributo es de carácter obligatorio y se le asigna un valor por defecto que además es el único valor que puede tener el atributo.

5.7.1. Estructura de un DTD. Elementos

Los posibles tipos de atributo son:

- CDATA: caracteres que no contienen etiquetas
- ENTITY: el nombre de una entidad (que debe declararse en el DTD)
- ENTITIES: una lista de nombres de entidades (que deben declararse en el DTD), separadas por espacios
- Enumerado: una lista de valores de entre los cuales, el atributo debe tomar uno

  Ejemplo: Se quiere definir una regla que valide la existencia de un elemento <semaforo>, de contenido vacío, con un único atributo color cuyos posibles valores sean rojo, naranja y verde. El valor por defecto será verde.

```
<!ELEMENT semaforo EMPTY>

<!ATTLIST semaforo

color (rojo | naranja | verde) "verde">
```

- ID: un identificador único. Se usa para identificar elementos, es decir, caracterizarlos de manera única. Por ello, dos elementos no pueden tener el mismo valor en atributos de tipo ID. Además, un elemento puede tener a lo sumo un atributo de tipo ID. El valor asignado a un atributo de este tipo debe ser un nombre XML válido.

5.7.1. Estructura de un DTD. Elementos

Los posibles tipos de atributo son: (continuación)

- IDREF: representa el valor de un atributo ID de otro elemento, es decir, para que sea válido, debe existir otro elemento en el documento XML que tenga un atributo de tipo ID y cuyo valor sea el mismo que el del atributo de tipo IDREF del primer elemento.

Ejemplo: Se quiere representar un elemento <empleado> que tenga dos atributos, idEmpleado e idEmpleadoJefe. El primero será de tipo ID y carácter obligatorio, y el segundo de tipo IDREF y carácter optativo. semaforo (nombre, apellido»

```
<!ATTLIST empleado
            idEmpleado ID #REQUIRED
            idEmpleadoJefe IDREF #IMPLIED>
...
```

Aquí tenemos un fragmento de XML válido con respecto a las reglas recién definidas. Cada elemento <empleado> tiene un atributo idEmpleado, con un valor válido (nombre XML válido) y el segundo elemento <empleado> tiene también un atributo idEmpleadoJefe, cuyo valor ha de ser el mismo que el del atributo de tipo ID de otro elemento existente en el documento. En este caso, este atributo idEmpleadoJefe del segundo <empleado> vale igual que el atributo idEmpleado del primer <empleado>:

```
<empleados>
  <empleado idEmpleado="e_111">...</empleado>
  <empleado idEmpleado="e_222" idEmpleadoJefe="e_111">
    ...
  </empleado>
</empleados>
```

✓

Un fragmento de XML no válido con respecto a las reglas recién definidas sería aquél en el cual el atributo idEmpleadoJefe del primer <empleado> tenga un valor que no exista para ningún atributo de tipo ID de otro elemento del documento:

```
<empleados>
  <empleado idEmpleado="e_111" idEmpleadoJefe="e_333">
    ...
  </empleado>
  <empleado idEmpleado="e_222" idEmpleadoJefe="e_111">
    ...
  </empleado>
</empleados>
```

✘

5.7.1. Estructura de un DTD. Elementos

Los posibles tipos de atributo son: (continuación)

- IDREFS: representa múltiples IDs de otros elementos, separados por espacios.
- NMTOKEN: cualquier nombre sin espacios en blanco en su interior. Los espacios en blanco anteriores o posteriores se ignorarán.

  Ejemplo: Se quiere declarar un atributo de tipo NMTOKEN de carácter obligatorio. En el siguiente DTD, la declaración del elemento <rio> y su atributo pais es:

```
<!ELEMENT rio (nombre)>
<!ATTLIST rio pais NMTOKEN #REQUIRED>
```

  En el siguiente fragmento XML, el valor del atributo pais es válido con respecto a la regla anteriormente definida:

```
<rio pais="EEUU">
  <nombre>Misisipi</nombre>
</rio>
```

✓

  En el siguiente documento XML, el valor del atributo pais no es válido con respecto a la regla anteriormente definida por contener espacios en su interior:

```
<rio pais="Estados Unidos">
  <nombre>Misisipi</nombre>
</rio>
```

✘

- NMTOKENS: una lista de nombres, sin espacios en blanco en su interior (los espacios en blanco anteriores o posteriores se ignorarán), separados por espacios.
- NOTATION: un nombre de notación, que debe estar declarada en el DTD.

5.7.1. Estructura de un DTD. Elementos

<!ENTITY>

Este es un elemento avanzado en el diseño de DTDs. Es una declaración de tipo de entidad.

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

- Referencia a entidades generales (internas o externas).
- Referencia a entidades parámetro (internas o externas).
- Entidades no procesadas (unparsed).

5.7.1. Estructura de un DTD. Elementos

Referencia a entidades generales, se utilizarán dentro del documento XML.

Sintaxis general: `<!ENTITY nombre_entidad definición_entidad>`

Ejemplo: En primer lugar, se declara una entidad en el DTD

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

A continuación, se usa en el XML anteponiendo al nombre de la entidad el carácter ampersand (&) y a continuación un carácter punto y coma (;). El programa analizador del documento realizará la sustitución.

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

Ejemplo: En el DTD de HTML se declaran diversas entidades para referenciar caracteres con acentos diversos, como é para referenciar el carácter é.

5.7.1. Estructura de un DTD. Elementos

Referencia entidades generales externas, ubicadas en otros archivos:

Sintaxis general: `<!ENTITY nombre_entidad tipo_uso url_archivo>`

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

Ejemplo: Se dispone de un archivo de texto, autores.txt, que contiene el siguiente texto plano "Juan Manuel y José Ramón".

Se crea un documento XML que hace referencia a ese archivo de texto, en forma de entidad externa. Al visualizar el documento, la referencia a la entidad general externa se sustituirá por el texto contenido en el archivo.

5.7.1. Estructura de un DTD. Elementos

Referencia entidades generales externas, ubicadas en otros archivos:

Sintaxis general: `<!ENTITY nombre_entidad tipo_uso url_archivo>`

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

Ejemplo: Se dispone de un archivo de texto, autores.txt, que contiene el siguiente texto plano "Juan Manuel y José Ramón".

Se crea un documento XML que hace referencia a ese archivo de texto, en forma de entidad externa. Al visualizar el documento, la referencia a la entidad general externa se sustituirá por el texto contenido en el archivo.

```
<?xml version="1.0"?>
<!DOCTYPE escritores [
  <!ELEMENT escritores (#PCDATA)>
  <!ENTITY autores SYSTEM "autores.txt">
]>
<escritores>&autores;</escritores>
```

5.7.1. Estructura de un DTD. Elementos

Referencia a entidades parámetro, que se usarán en el propio DTD y funcionan cuando la definición de las reglas del DTD se realiza en un archivo externo.

Sintaxis general: `<!ENTITY % nombre entidad definición entidad>`

Ejemplo: Se declara una entidad parámetro dimensiones y se referencia dentro del propio DTD.

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

Este código es equivalente a haber escrito:

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

5.7.1. Estructura de un DTD. Elementos

Referencia entidades parámetro externas, ubicadas en otros archivos:
Sintaxis general:

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

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

Si es PUBLIC, es necesario definir el FPI que, recordemos, es el nombre por el cual se identifica públicamente un determinado elemento (sea un DOCTYPE, un ELEMENT o una ENTITY), en este caso la entidad.

5.7.1. Estructura de un DTD. Elementos

Entidades no procesadas, referencian a datos que no deben ser procesados por el analizador XML, sino por la aplicación que 10 use (como una imagen).

Sintaxis general: `<!ENTITY % nombre entidad tipo_uso fpi valor_entidad NDATA tipo>`

Ejemplo: Se declara una notación (se comentará más adelante) de nombre JPG para el tipo MIME (Multipurpose Internet Mail Extensions - Extensiones Multipropósito de Correo de Internet).

Se declara una entidad no procesada de nombre mediterraneo, asociada al archivo de imagen mediterraneo.jpg. Por último, se declara un elemento <mar>, que cuenta con atributo imagen que es del tipo ENTITY recién declarado.

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

y en el XML, el valor del atributo imagen del elemento <mar> es la entidad no procesada declarada en el DTD:

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

5.7.1. Estructura de un DTD. Elementos

<!NOTATION>

Este es un elemento avanzado en el diseño de DTDs. Es una declaración del tipo de atributo NOTATION. Una notación se usa para especificar un formato de datos que no sea XML.

Se usa con frecuencia para describir tipos MIME, como image/gif o image/jpg. Se utiliza para indicar un tipo de atributo al que se le permite usar un valor que haya sido declarado como notación en el DTD.

Sintaxis general de la notación:

```
<!NOTATION nombre notación SYSTEM "identificador externo">
```

Sintaxis general del atributo que la usa:

```
<!ATTLIST nombre elemento nombre atributo NOTATION valor defecto>
```

5.7.1. Estructura de un DTD. Elementos

Ejemplo: Se declaran tres notaciones que corresponden a otros tantos tipos MIME de imágenes (gif, jpg Y png). También se declara un elemento <mar> y sus atributos imagen y formato_imagen, este último referenciando a las notaciones recién creadas. Por último, se declara una entidad no procesada que se asocia a un archivo de imagen.

```
<!NOTATION GIF SYSTEM "image/gif">
<!NOTATION JPG SYSTEM "image/jpeg">
<!NOTATION PNG SYSTEM "image/png">
<!ELEMENT mar ... >
<!ATTLIST mar
          imagen ENTITY #IMPLIED
          formato_imagen NOTATION (GIF | JPG | PNG) #IMPLIED>
<!ENTITY mediterraneo SYSTEM "mediterraneo.jpg">
```

y en el documento XML, el valor del atributo imagen del elemento <mar> es la entidad no procesada declarada en el DTD, y el valor del atributo formato_imagen el de una de sus alternativas válidas, la notación JPG:

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

5.7.1. Estructura de un DTD. Elementos

Secciones condicionales

Permiten incluir o excluir reglas en un DTD en función de condiciones. Sólo se pueden ubicar en DTDs externos. Su uso tiene sentido al combinarlas con referencias a entidades parámetro. Las secciones condicionales son **IGNORE** e **INCLUDE**, teniendo la primera precedencia sobre la segunda.

Ejemplo:

Supóngase dos estructuras diferentes para un mensaje:

- Una sencilla que incluye emisor, receptor y contenido otra extendida que incluye los datos de la estructura sencilla más el título y el número de palabras.

- Se quiere diseñar un DTD que, en función del valor de una entidad parámetro, incluya una estructura de mensaje o la otra. Por defecto se incluirá la larga.

5.7.1. Estructura de un DTD. Elementos

El DTD, que ha de ser externo, tendrá la siguiente estructura:

```
<!-- Mensaje corto -->
<![IGNORE[
   <!ELEMENT mensaje (emisor, receptor, contenido)>
]]>

<!-- Mensaje largo -->
<![INCLUDE[
   <!ELEMENT mensaje (titulo, emisor, receptor, contenido, palabras)>
   <!ELEMENT titulo (#PCDATA)>
   <!ELEMENT palabras (#PCDATA)>
]]>

<!-- Declaración de elementos y atributos comunes -->
<!ELEMENT emisor (#PCDATA)>
<!ELEMENT receptor (#PCDATA)>
<!ELEMENT contenido (#PCDATA)>
```

5.7.1. Estructura de un DTD. Elementos

Al añadir las referencias a entidad parámetro, el DTD quedaría:

```
<!ENTITY %corto "IGNORE">
<!ENTITY %largo "INCLUDE">

<!-- Mensaje corto -->
<![%corto[
   <!ELEMENT mensaje (emisor, receptor, contenido)>
]]>

<!-- Mensaje largo -->
<![%largo[
   <!ELEMENT mensaje (titulo, emisor, receptor, contenido, palabras)>
   <!ELEMENT titulo (#PCDATA)>
   <!ELEMENT palabras (#PCDATA)>
]]>
<!-- Declaración de elementos y atributos comunes -->
<!ELEMENT emisor (#PCDATA)>
<!ELEMENT receptor (#PCDATA)>
<!ELEMENT contenido (#PCDATA)>
```

5.7.1. Estructura de un DTD. Elementos

Para cambiar el tipo de mensaje que se va a incluir, basta con modificar la asociación de valores de %corto a INCLUDE y de %largo a IGNORE, de la forma:

```
<!ENTITY %corto "INCLUDE">
<!ENTITY %largo "IGNORE">
```

Se podrían dejar la declaración de las entidades en la DTD interna al documento XML, donde se determinaría qué tipo de mensaje se quiere incluir de manera específica para ese documento:

```
<?xml verion="1.0"?>
<!DOCTYPE mensaje SYSTEM "mensaje.dtd" [
  <!ENTITY %corto "INCLUDE">
  <!ENTITY %largo "IGNORE">
]>
 <mensaje>
   ...
 </mensaje>
```

5.7.2. Poniendo todo junto

Se quiere construir un DTD que valide el siguiente documento XML, persona.xml:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE persona SYSTEM "persona.dtd"]>
<persona dni="12345678-L" estadoCivil="Casado">
  <nombre>María Pilar</nombre>
  <apellido>Sánchez</apellido>
  <edad>60</edad>
  <enActivo/>
</persona>
```

5.7.2. Poniendo todo junto

Se quiere construir un DTD que valide el siguiente documento XML, persona.xml:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE persona SYSTEM "persona.dtd"]>
<persona dni="12345678-L" estadoCivil="Casado">
  <nombre>María Pilar</nombre>
  <apellido>Sánchez</apellido>
  <edad>60</edad>
  <enActivo/>
</persona>
```

El esquema XML asociado se quiere que cumpla ciertas restricciones semánticas:

- El atributo dni es un identificador obligatorio.
- El estado civil puede ser: Soltero, Casado o Divorciado. Por defecto es Soltero.
- El elemento <enActivo> es optativo.

El DTD que cumple con esto, ubicado en el archivo persona.dtd, es:

```
<!ELEMENT persona (nombre, apellido, edad, enActivo?)>
<!ATTLIST persona dni ID #REQUIRED
          estadoCivil (Soltero | Casado | Divorciado) "Soltero">
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellido (#PCDATA)>
<!ELEMENT edad (#PCDATA)>
<!ELEMENT enActivo (#PCDATA) EMPTY>
```

5.7.2. Poniendo todo junto

<!ENTITY>

Este es un elemento avanzado en el diseño de DTDs. Es una declaración de tipo de entidad.

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

- Referencia a entidades generales (internas o externas).
- Referencia a entidades parámetro (internas o externas).
- Entidades no procesadas (unparsed).

5.7.2. Poniendo todo junto

- Referencia a **entidades generales**, se utilizarán dentro del documento XML.

Sintaxis general: <!ENTITY nombre_entidad definición_entidad>

Ejemplo: En primer lugar, se declara una entidad en el DTD

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

A continuación, se usa en el XML anteponiendo al nombre de la entidad el carácter ampersand (&) y a continuación un carácter punto y coma (;). El programa analizador del documento realizará la sustitución.

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

Ejemplo: En el DTD de HTML se declaran diversas entidades para referenciar caracteres con acentos diversos, como `&eacute;` para referenciar el carácter `é`.

5.7.2. Poniendo todo junto

- Referencia **entidades generales externas**, ubicadas en otros archivos:

Sintaxis general: `<!ENTITY nombre_entidad tipo_uso url_archivo>`

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

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

5.7.2. Poniendo todo junto

- Referencia entidades generales externas, ubicadas en otros archivos:

Sintaxis general: `<!ENTITY nombre_entidad tipo_uso url_archivo>`

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

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

```
<?xml version="1.0"?>
<!DOCTYPE escritores [
  <!ELEMENT escritores (#PCDATA)>
  <!ENTITY autores SYSTEM "autores.txt">
]>
<escritores>&autores;</escritores>
```

5.7.2. Poniendo todo junto

- Referencia a **entidades parámetro**, que se usarán en el propio DTD y funcionan cuando la definición de las reglas del DTD se realiza en un archivo externo.

Sintaxis general: <!ENTITY % nombre_entidad definición_entidad>

Ejemplo: Se declara una entidad parámetro dimensiones y se referencia dentro del propio DTD.

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

Este código es equivalente a haber escrito:

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

5.7.2. Poniendo todo junto

- Referencia **entidades parámetro externas**, ubicadas en otros archivos:

Sintaxis general: `<!ENTITY % nombre_entidad tipo_uso fpi url_archivo>`

Siendo el tipo de uso privado (SYSTEM) o público (PUBLIC). Si es PUBLIC, es necesario definir el FPI que, recordemos, es el nombre por el cual se identifica públicamente un determinado elemento (sea un DOCTYPE, un ELEMENT o una ENTITY), en este caso la entidad.

5.7.2. Poniendo todo junto

- Entidades no procesadas, referencian a datos que no deben ser procesados por el analizador XML, sino por la aplicación que 10 use (como una imagen).

Sintaxis general: `<!ENTITY % nombre entidad tipo_uso fpi valor entidad NDATA tipo>`

Ejemplo: Se declara una notación (se comentará más adelante) de nombre JPG para el tipo MIME (Multipurpose Internet Mail Extensions - Extensiones Multipropósito de Correo de Internet).

Se declara una entidad no procesada de nombre mediterraneo, asociada al archivo de imagen mediterraneo.jpg. Por último, se declara un elemento <mar>, que cuenta con atributo imagen que es del tipo ENTITY recién declarado.

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

5.7.2. Poniendo todo junto

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

y en el XML, el valor del atributo imagen del elemento <mar> es la entidad no procesada declarada en el DTD:

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

5.7.2. Poniendo todo junto

<!NOTATION>

Este es un elemento avanzado en el diseño de DTDs. Es una declaración del tipo de atributo NOTATION. Una notación se usa para especificar un formato de datos que no sea XML.

Se usa con frecuencia para describir tipos MIME, como image/gif o image(jpg.

Se utiliza para indicar un tipo de atributo al que se le permite usar un valor que haya sido declarado como notación en el DTD.

Sintaxis general de la notación:

```
<!NOTATION nombre_notación SYSTEM "identificador externo">
```

Sintaxis general del atributo que la usa:

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

5.7.2. Poniendo todo junto

Ejemplo: Se declaran tres notaciones que corresponden a otros tantos tipos MIME de imágenes (gif, jpg y png). También se declara un elemento <mar> y sus atributos imagen y formato_imagen, este último referenciando a las notaciones recién creadas. Por último, se declara una entidad no procesada que se asocia a un archivo de imagen.

```
<!NOTATION GIF SYSTEM "image/gif">
<!NOTATION JPG SYSTEM "image/jpeg">
<!NOTATION PNG SYSTEM "image/png">
<!ELEMENT mar ... >
<!ATTLIST mar
          imagen ENTITY #IMPLIED
          formato_imagen NOTATION (GIF | JPG | PNG) #IMPLIED>
<!ENTITY mediterraneo SYSTEM "mediterraneo.jpg">
```

y en el documento XML, el valor del atributo imagen del elemento <mar> es la entidad no procesada declarada en el DTD, y el valor del atributo formato_imagen el de una de sus alternativas válidas, la notación JPG:

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

5.7.2. Poniendo todo junto

Secciones condicionales

Permiten incluir o excluir reglas en un DTD en función de condiciones. Sólo se pueden ubicar en DTDs externos. Su uso tiene sentido al combinarlas con referencias a entidades parámetro.

Las secciones condicionales son IGNORE e INCLUDE, teniendo la primera precedencia sobre la segunda.

Ejemplo: Supónganse dos estructuras diferentes para un mensaje: una sencilla que incluye emisor, receptor y contenido otra extendida que incluye los datos de la estructura sencilla más el título y el número de palabras.

Se quiere diseñar un DTD que, en función del valor de una entidad parámetro, incluya una estructura de mensaje o la otra. Por defecto se incluirá la larga.

5.7.2. Poniendo todo junto

El DTD, que ha de ser externo, tendrá la siguiente estructura:

```
<!-- Mensaje corto -->
<![IGNORE[
    <!ELEMENT mensaje (emisor, receptor, contenido)>
]]>

<!-- Mensaje largo -->
<![INCLUDE[
    <!ELEMENT mensaje (titulo, emisor, receptor, contenido, palabras)>
    <!ELEMENT titulo (#PCDATA)>
    <!ELEMENT palabras (#PCDATA)>
]]>

<!-- Declaración de elementos y atributos comunes -->
<!ELEMENT emisor (#PCDATA)>
<!ELEMENT receptor (#PCDATA)>
<!ELEMENT contenido (#PCDATA)>
```

5.7.2. Poniendo todo junto

Al añadir las referencias a entidad parámetro, el DTD quedaría:

```
<!ENTITY %corto "IGNORE">
<!ENTITY %largo "INCLUDE">

<!-- Mensaje corto -->
<![%corto[
   <!ELEMENT mensaje (emisor, receptor, contenido)>
]]>

<!-- Mensaje largo -->
<![%largo[
   <!ELEMENT mensaje (titulo, emisor, receptor, contenido, palabras)>
   <!ELEMENT titulo (#PCDATA)>
   <!ELEMENT palabras (#PCDATA)>
]]>
<!-- Declaración de elementos y atributos comunes -->
<!ELEMENT emisor (#PCDATA)>
<!ELEMENT receptor (#PCDATA)>
<!ELEMENT contenido (#PCDATA)>
```

5.7.2. Poniendo todo junto

Para cambiar el tipo de mensaje que se va a incluir, basta con modificar la asociación de valores de %corto a INCLUDE y de %largo a IGNORE, de la forma:

```
<!ENTITY %corto "INCLUDE">
<!ENTITY %largo "IGNORE">
```

Se podrían dejar la declaración de las entidades en la DTD interna al documento XML, donde se determinaría qué tipo de mensaje se quiere incluir de manera específica para ese documento:

```
<?xml verion="1.0"?>
<!DOCTYPE mensaje SYSTEM "mensaje.dtd" [
  <!ENTITY %corto "INCLUDE">
  <!ENTITY %largo "IGNORE">
]>
```

5.7.2. Poniendo todo junto

Limitaciones de los DTD

Algunas limitaciones de los DTD son:

1. Un DTD no es un documento XML, luego no se puede verificar si está bien formado.
2. No se pueden fijar restricciones sobre los valores de elementos y atributos, como su tipo de datos, su tamaño, etc.
3. No soporta espacios de nombres.
4. Sólo se puede enumerar los valores de atributos, no de elementos.
5. Sólo se puede dar un valor por defecto para atributos, no para elementos.
6. Existe un control limitado sobre las cardinalidades de los elementos, es decir, concretar el número de veces que pueden aparecer.