# Apuntes de DTD

{Abrirllave.com – Tutoriales de informática

---

Contenidos del tutorial de DTD www.abrirllave.com/dtd/

(Estos apuntes incluyen enlaces a los apartados del tutorial en la Web)

# 1. Qué es DTD

**DTD** (*Document Type Definition*, Definición de Tipo de Documento) sirve para definir la estructura de un documento SGML o XML, permitiendo su validación.

- **SGML** (*Standard Generalized Markup Language*, Lenguaje de Marcado Generalizado Estándar). Véase: http://www.w3.org/MarkUp/SGML/.
- **XML** (*eXtensible Markup Language*, Lenguaje de Marcado eXtensible) es un lenguaje desarrollado por **W3C** (*World Wide Web Consortium*) que está basado en SGML.

En "http://www.w3.org/TR/xml/" se puede consultar la *W3C Recommendation* de XML, en la cual se fundamenta este tutorial, donde se explica –de forma introductoria a través de ejemplos– **cómo escribir y utilizar DTD para validar documentos XML**.

Un documento XML es válido (*valid*) cuando, además de estar bien formado, no incumple ninguna de las normas establecidas en su estructura.

Existen otros métodos que también permiten validar documentos XML, como por ejemplo *XML Schema* o *RELAX NG*.

# 2. Declaración de tipo de documento

Una DTD se puede escribir tanto interna como externamente a un archivo XML. Ahora bien, en ambos casos hay que escribir una definición **DOCTYPE** (*Document Type Declaration*, Declaración de Tipo de Documento) para asociar el documento XML a la DTD. Asimismo, un archivo XML se puede asociar simultáneamente a una DTD interna y externa.

## 2.1. Documento XML asociado a una DTD interna

La sintaxis para escribir una DTD interna es:

```
<!DOCTYPE elemento-raíz [ declaraciones ]>
```

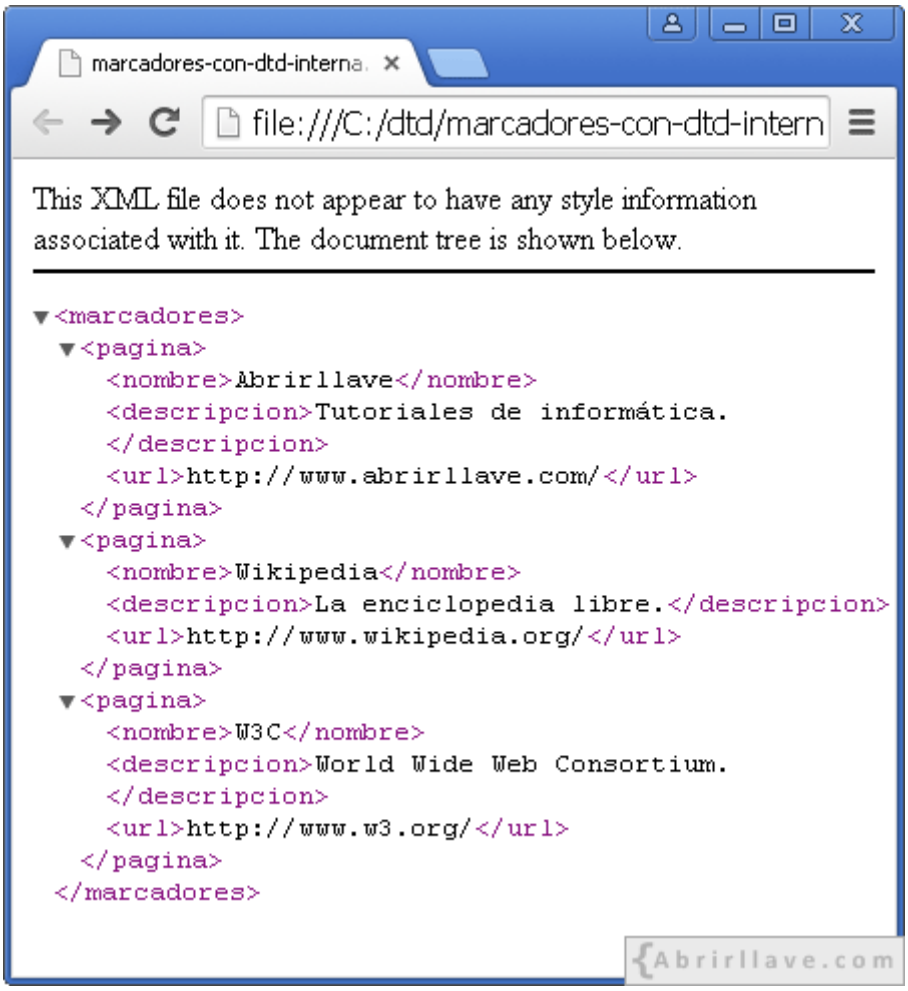
**EJEMPLO** En un documento XML se quiere guardar una lista de marcadores de páginas web, almacenando de cada uno de ellos su nombre, una descripción y su URL. Para ello, se puede escribir, por ejemplo, el archivo ***"marcadores-con-dtd-interna.xml"*** siguiente, que contiene una DTD interna:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE marcadores [
   <!ELEMENT marcadores (pagina)*>
   <!ELEMENT pagina (nombre, descripcion, url)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT descripcion (#PCDATA)>
   <!ELEMENT url (#PCDATA)>
]>

<marcadores>
   <pagina>
      <nombre>Abrirllave</nombre>
      <descripcion>Tutoriales de informática.</descripcion>
      <url>http://www.abrirllave.com/</url>
   </pagina>
   <pagina>
      <nombre>Wikipedia</nombre>
      <descripcion>La enciclopedia libre.</descripcion>
      <url>http://www.wikipedia.org/</url>
   </pagina>
   <pagina>
      <nombre>W3C</nombre>
      <descripcion>World Wide Web Consortium.</descripcion>
      <url>http://www.w3.org/</url>
   </pagina>
</marcadores>
```

- Obsérvese que, en la DTD se ha indicado que **marcadores** es el elemento raíz del documento XML, el cual puede contener cero o más páginas. Para indicar esto último, se ha escrito: **(pagina)***
- Escribiendo **pagina (nombre, descripcion, url)** se especifica que, cada elemento "pagina" tiene que contener tres elementos (hijos): "nombre", "descripcion" y "url".
- Con **#PCDATA** (*Parsed Character Data*) escrito entre paréntesis "**()**" se indica que los elementos "nombre", "descripcion" y "url" pueden contener texto (cadenas de caracteres) analizable por un procesador XML.

**Nota:** al nombrar a los elementos "pagina" y "descripcion" no se han utilizado los caracteres (á) y (ó), respectivamente, para evitar posibles incompatibilidades con programas que puedan no reconocerlos.

Al ver el archivo ***"marcadores-con-dtd-interna.xml"*** en un navegador web, como por ejemplo *Google Chrome*, se visualizará algo similar a:




Como se puede observar, la DTD no se muestra en el navegador.

Por otro lado, para comprobar que el documento XML escrito en este ejemplo es válido se pueden utilizar distintos programas. Por ejemplo, véase cómo validar un documento XML asociado a una DTD con XML Copy Editor.

## 2.2. Documento XML asociado a una DTD externa

Existen dos tipos de DTD externa: privada y pública. Para las privadas se utiliza **SYSTEM** y para las públicas **PUBLIC**. La sintaxis en cada caso es:

```
<!DOCTYPE elemento-raíz SYSTEM "URI">
```

```
<!DOCTYPE elemento-raíz PUBLIC "identificador-público" "URI">
```

## DTD externa privada - **SYSTEM**

**EJEMPLO** Si en un archivo llamado ***"marcadores.dtd"*** se escribiese la siguiente DTD:

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

El siguiente documento XML llamado ***"marcadores-con-dtd-externa.xml"***, sería válido:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE marcadores SYSTEM "marcadores.dtd">
<marcadores>
   <pagina>
      <nombre>Abrirllave</nombre>
      <descripcion>Tutoriales de informática.</descripcion>
      <url>http://www.abrirllave.com/</url>
   </pagina>
   <pagina>
      <nombre>Wikipedia</nombre>
      <descripcion>La enciclopedia libre.</descripcion>
      <url>http://www.wikipedia.org/</url>
   </pagina>
   <pagina>
      <nombre>W3C</nombre>
      <descripcion>World Wide Web Consortium.</descripcion>
      <url>http://www.w3.org/</url>
   </pagina>
</marcadores>
```

- En este documento XML, haciendo uso de una DTD externa privada, se han escrito una lista de marcadores de páginas web, guardando de cada uno de ellos su nombre, una descripción y su URL.

## DTD externa pública - **PUBLIC**

**EJEMPLO** El siguiente documento XML está asociado a una DTD externa pública:

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

- **-//W3C//DTD XHTML 1.0 Strict//EN** es un FPI (*Formal Public Identifier*, Identificador Público Formal).

# 2.3. Cuándo utilizar una DTD interna o una DTD externa

Para validar más de un documento XML con la misma DTD, escribir esta en un archivo externo proporciona la ventaja de no tener que repetir la DTD internamente en cada documento XML.

En el caso de que la DTD solo se utilice para validar un único documento XML, la DTD es habitual escribirla internamente.

# 2.4. Uso combinado de DTD interna y externa en un documento XML

Para asociar un documento XML a una DTD interna y externa simultáneamente, se pueden utilizar las siguientes sintaxis:

```
<!DOCTYPE elemento-raíz SYSTEM "URI" [ declaraciones ]>
```

```
<!DOCTYPE elemento-raíz PUBLIC "identificador-público" "URI" [
declaraciones ]>
```

**EJEMPLO** Si en un documento XML llamado ***"marcadores-con-dtd-interna-y-externa.xml"*** se quiere almacenar una lista de marcadores de páginas web, guardando de cada uno de ellos su nombre, una descripción y su URL. En dicho documento se podría escribir:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE marcadores SYSTEM "marcadores.dtd" [
   <!ELEMENT marcadores (pagina)*>
   <!ELEMENT pagina (nombre, descripcion, url)>
]>

<marcadores>
   <pagina>
      <nombre>Abrirllave</nombre>
      <descripcion>Tutoriales de informática.</descripcion>
      <url>http://www.abrirllave.com/</url>
   </pagina>
   <pagina>
      <nombre>Wikipedia</nombre>
      <descripcion>La enciclopedia libre.</descripcion>
      <url>http://www.wikipedia.org/</url>
   </pagina>
   <pagina>
      <nombre>W3C</nombre>
      <descripcion>World Wide Web Consortium.</descripcion>
      <url>http://www.w3.org/</url>
   </pagina>
</marcadores>
```

De tal forma que, el contenido del archivo ***"marcadores.dtd"*** podría ser:

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

# 3. Estructura de un documento XML

En una DTD se pueden declarar:

- Elementos
- Atributos
- Entidades
- Notaciones

Por tanto, **un documento XML será válido si** –además de no tener errores de sintaxis– **cumple lo indicado en las declaraciones de elementos, atributos, entidades y notaciones, de la DTD a la que esté asociado**.

# 4. Declaración de elementos

Para declarar un elemento en una DTD se utiliza la siguiente sintaxis:

```
<!ELEMENT nombre-del-elemento tipo-de-contenido>
```

En el **tipo-de-contenido** se especifica el contenido permitido en el elemento, pudiendo ser:

- Texto, **(#PCDATA)**.
- Otros elementos (hijos).
- Estar vacío, **EMPTY**.
- Mixto (texto y otros elementos), **ANY**.

## El contenido de un elemento puede ser texto - **(#PCDATA)**

**EJEMPLO** En el siguiente documento XML, el elemento "ciudad" puede contener cualquier texto (cadena de caracteres):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ciudad [
   <!ELEMENT ciudad (#PCDATA)>
]>
<ciudad>Roma</ciudad>
```

- Escribiendo **#PCDATA** (*Parsed Character Data*) entre paréntesis "**()**", se ha indicado que el elemento "ciudad" puede contener una cadena de caracteres analizable.

## Un elemento puede contener a otros elementos

**EJEMPLO** En el siguiente ejemplo, el elemento "ciudad" contiene a los elementos "nombre" y "país":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ciudad [
   <!ELEMENT ciudad (nombre, pais)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT pais (#PCDATA)>
]>

<ciudad>
   <nombre>Roma</nombre>
   <pais>Italia</pais>
</ciudad>
```

## Un elemento puede no contener contenido (estar vacío) - **EMPTY**

**EJEMPLO** En la DTD interna del siguiente documento XML, se ha declarado el elemento "mayor_de_edad" como vacío, **EMPTY**. Por tanto, debe escribirse **<mayor_de_edad/>**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, mayor_de_edad, ciudad)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT mayor_de_edad EMPTY>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Elsa</nombre>
   <mayor_de_edad/>
   <ciudad>Pamplona</ciudad>
</persona>
```

**Nota:** los elementos vacíos no pueden tener contenido, pero sí pueden tener atributos.

## Un elemento puede definirse para contener contenido mixto - **ANY**

**EJEMPLO** En la DTD interna del siguiente documento XML, se ha indicado que el elemento "persona" puede contener texto y otros elementos, es decir, contenido mixto, **ANY**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona ANY>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>
```

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

Obsérvese que, por ejemplo, también sería válido el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona ANY>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

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

O el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona ANY>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

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

Incluso, si el elemento "persona" estuviese vacío, el documento también sería válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona ANY>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona/>
```

# 4.1. Elementos vacíos - EMPTY

Para declarar un elemento vacío en una DTD, hay que indicar que su contenido es **EMPTY**. Un ejemplo de ello podría ser el elemento "br" del HTML, el cual sirve para hacer un salto de línea y no tiene contenido:

```
<!ELEMENT br EMPTY>
```

Dada la declaración anterior, en un documento XML el elemento "br" podría escribirse como:

```
<br/>
```

O también:

```
<br></br>
```

Por ejemplo, el siguiente documento XML sería válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE etiquetas_html [
   <!ELEMENT etiquetas_html (br)>
   <!ELEMENT br EMPTY>
]>

<etiquetas_html>
   <br/>
</etiquetas_html>
```

## Un elemento vacío puede tener atributos

**EJEMPLO** Aunque un elemento se declare vacío, no pudiendo contener texto ni otros elementos, sí puede tener atributos:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE etiquetas_html [
   <!ELEMENT etiquetas_html (br)>
   <!ELEMENT br EMPTY>
   <!ATTLIST br descripcion CDATA #REQUIRED>
]>

<etiquetas_html>
   <br descripcion="Salto de línea"/>
</etiquetas_html>
```

- En este ejemplo, para el elemento "br" se ha declarado el atributo **descripcion** de tipo **CDATA** (*Character DATA*), es decir, su valor puede ser una cadena de caracteres. Además, se ha indicado que el atributo es obligatorio escribirlo, **#REQUIRED**.

# 4.2. Elementos con cualquier contenido - ANY

Cuando en una DTD se quiere declarar un elemento que pueda contener cualquier contenido –bien sea texto, otros elementos o una mezcla de ambos– esto se puede hacer indicando que su contenido es de tipo **ANY**:

```
<!ELEMENT cualquier_contenido ANY>
```

**EJEMPLO** En el siguiente documento XML, el elemento "cualquier_contenido" contiene tres elementos "texto":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
   <!ELEMENT ejemplo (cualquier_contenido)>
   <!ELEMENT cualquier_contenido ANY>
   <!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
   <cualquier_contenido>
      <texto>Texto1</texto>
      <texto>Texto2</texto>
      <texto>Texto3</texto>
   </cualquier_contenido>
</ejemplo>
```

Fíjese que, definiendo la misma DTD, también sería válido el siguiente documento XML donde el elemento "cualquier_contenido" solo contiene texto:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
   <!ELEMENT ejemplo (cualquier_contenido)>
   <!ELEMENT cualquier_contenido ANY>
   <!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
   <cualquier_contenido>Texto1. Texto2. Texto3</cualquier_contenido>
</ejemplo>
```

Asimismo, el elemento "cualquier_contenido" podría contener una mezcla de texto y uno o más elementos.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
   <!ELEMENT ejemplo (cualquier_contenido)>
   <!ELEMENT cualquier_contenido ANY>
   <!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
   <cualquier_contenido>Texto1<texto>Texto2</texto>Texto3</cualquier_contenido>
</ejemplo>
```

Por otra parte, si el elemento "cualquier_contenido" estuviese vacío, el documento XML seguiría siendo válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
   <!ELEMENT ejemplo (cualquier_contenido)>
   <!ELEMENT cualquier_contenido ANY>
   <!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
   <cualquier_contenido></cualquier_contenido>
</ejemplo>
```

- En vez de **<cualquier_contenido></cualquier_contenido>**, también se puede escribir **<cualquier_contenido/>**.

## 4.3. Elementos con contenido de tipo texto - (#PCDATA)

Para declarar en una DTD un elemento que pueda contener texto analizable, se tiene que indicar que su contenido es **(#PCDATA)**, (*Parsed Character Data*):

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

**EJEMPLO** En el siguiente documento XML, el elemento "texto" contiene caracteres:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
   <!ELEMENT ejemplo (texto)>
   <!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
   <texto>Este elemento solo contiene caracteres.</texto>
</ejemplo>
```

Ahora bien, el elemento "texto" podría estar vacío y el documento XML seguiría siendo válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
   <!ELEMENT ejemplo (texto)>
   <!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
   <texto></texto>
</ejemplo>
```

En vez de **<texto></texto>**, también se puede escribir **<texto/>**.

# 4.4. Secuencias de elementos

En una DTD, un elemento (padre) puede ser declarado para contener a otro u otros elementos (hijos). En la sintaxis, los hijos –también llamados sucesores– tienen que escribirse entre paréntesis "**()**" y separados por comas "**,**".

## Elemento con varios hijos

**EJEMPLO** Para declarar un elemento (padre) que contenga tres elementos (hijos), se puede escribir:

```
<!ELEMENT padre (hijo1, hijo2, hijo3)>
```

**EJEMPLO** En el siguiente documento XML, el elemento "persona" contiene a los elementos "nombre", "fecha_de_nacimiento" y "ciudad":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, fecha_de_nacimiento, ciudad)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT fecha_de_nacimiento (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Iker</nombre>
   <fecha_de_nacimiento>26-12-1997</fecha_de_nacimiento>
   <ciudad>Valencia</ciudad>
</persona>
```

A su vez, los hijos también pueden tener sus propios hijos. Así, el elemento "fecha_de_nacimiento" puede contener, por ejemplo, a los elementos "dia", "mes" y "anio":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, fecha_de_nacimiento, ciudad)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT fecha_de_nacimiento (dia, mes, anio)>
   <!ELEMENT dia (#PCDATA)>
   <!ELEMENT mes (#PCDATA)>
   <!ELEMENT anio (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Iker</nombre>
   <fecha_de_nacimiento>
      <dia>26</dia>
      <mes>12</mes>
      <anio>1997</anio>
   </fecha_de_nacimiento>
   <ciudad>Valencia</ciudad>
</persona>
```

## Orden de los hijos de un elemento

En un documento XML, los elementos (hijos) de un elemento (padre), deben escribirse en el mismo orden en el que han sido declarados en la DTD.

**EJEMPLO** El siguiente documento XML no es válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, fecha_de_nacimiento, ciudad)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT fecha_de_nacimiento (dia, mes, anio)>
   <!ELEMENT dia (#PCDATA)>
   <!ELEMENT mes (#PCDATA)>
   <!ELEMENT anio (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Iker</nombre>
   <fecha_de_nacimiento>
      <anio>1997</anio>
      <mes>12</mes>
      <dia>26</dia>
   </fecha_de_nacimiento>
   <ciudad>Valencia</ciudad>
</persona>
```

- El documento no es válido porque los elementos sucesores (hijos) del elemento "fecha_de_nacimiento" no se han escrito en el mismo orden que en la DTD.

# 4.5. Cardinalidad de los elementos

En una DTD, para definir el número de veces que pueden aparecer los elementos de un documento XML, se pueden utilizar los *operadores de cardinalidad* mostrados en la siguiente tabla:

| **Operadores de cardinalidad en DTD** | | |
|---|---|---|
| ***Operador*** | ***Cardinalidad*** | ***Significado*** |
| **?** (*interrogación*) | 0-1 | El elemento es opcional, pudiendo aparecer una sola vez o ninguna. |
| ***** (*asterisco*) | 0-n | El elemento puede aparecer cero, una o más veces. |
| **+** (*signo más*) | 1-n | El elemento tiene que aparecer, obligatoriamente, una o más veces. |

**Nota:** los elementos declarados en una DTD sobre los que no actúe ningún operador de cardinalidad, tendrán que aparecer obligatoriamente una única vez, en el o los documentos XML a los que se asocie.

## Operador de cardinalidad "**+**" (signo más)

**EJEMPLO** En el siguiente documento XML, el elemento "nombre" tiene que aparecer una o más veces. En este caso, aparece tres veces:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE personas [
   <!ELEMENT personas (nombre+)>
   <!ELEMENT nombre (#PCDATA)>
]>

<personas>
   <nombre>Ana</nombre>
   <nombre>Iker</nombre>
   <nombre>Elsa</nombre>
</personas>
```

- Si sobre **nombre** no actuase el operador (**+**) el documento no sería válido, ya que, el elemento "personas" solo tendría que contener un elemento "nombre".
- En vez de **(nombre+)**, también se puede escribir **(nombre)+**.

## Operador de cardinalidad "*****" (asterisco)

**EJEMPLO** En la DTD interna del siguiente documento XML, se ha indicado que el elemento "nombre" tiene que aparecer una única vez. Ahora bien, el elemento "ingrediente" tiene cardinalidad (0-n), por tanto, puede aparecer cero, una o más veces:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE receta_de_cocina [
   <!ELEMENT receta_de_cocina (nombre, ingrediente*)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ingrediente (#PCDATA)>
]>

<receta_de_cocina>
   <nombre>Tortilla de patatas</nombre>
   <ingrediente>Huevo</ingrediente>
   <ingrediente>Patata</ingrediente>
   <ingrediente>Aceite</ingrediente>
   <ingrediente>Sal</ingrediente>
</receta_de_cocina>
```

## Operador de cardinalidad "**?**" (interrogación)

**EJEMPLO** En la DTD del siguiente documento XML, la cardinalidad del elemento "mayor_de_edad" es (0-1), siendo opcional su aparición:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, mayor_de_edad?)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT mayor_de_edad EMPTY>
]>

<persona>
   <nombre>Iker</nombre>
   <mayor_de_edad/>
</persona>
```

Así pues, el siguiente documento también es válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, mayor_de_edad?)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT mayor_de_edad EMPTY>
]>

<persona>
   <nombre>Pedro</nombre>
</persona>
```

# 4.6. Elementos opcionales

En la DTD asociada a un documento XML, se pueden declarar elementos que contengan elementos opcionales. Para ello, se utiliza el *operador de elección*, representado por una barra vertical (**|**).

## Operador de elección "**|**" (barra vertical)

**EJEMPLO** En el siguiente documento XML el elemento "articulo" puede contener un elemento "codigo" o un elemento "id"; obligatoriamente uno de ellos, pero no ambos:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulo [
   <!ELEMENT articulo (codigo | id)>
   <!ELEMENT codigo (#PCDATA)>
   <!ELEMENT id (#PCDATA)>
]>

<articulo>
   <codigo>AF-33</codigo>
</articulo>
```

## Operador de elección "|" y operador "*"

**EJEMPLO** En la DTD del siguiente documento XML se indica que el elemento "articulos" puede contener varios elementos "codigo" e "id":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulos [
   <!ELEMENT articulos (codigo | id)*>
   <!ELEMENT codigo (#PCDATA)>
   <!ELEMENT id (#PCDATA)>
]>

<articulos>
   <codigo>AF-32</codigo>
   <id>3891</id>
   <codigo>AF-50</codigo>
   <codigo>AF-89</codigo>
</articulos>
```

Obsérvese que, con el operador "*", en este ejemplo se ha indicado que el contenido del elemento "articulos" tiene cardinalidad (0-n). Por tanto, el elemento "articulos" puede:

- Estar vacío.
- Contener un elemento "codigo".
- Contener un elemento "id".
- Contener un elemento "codigo" y un elemento "id".
- Contener un elemento "codigo" y varios elementos "id".
- Contener un elemento "id" y varios elementos "codigo".
- Contener varios elementos "codigo" y varios elementos "id".

Nótese también que, dentro del elemento "articulos" pueden aparecer elementos "codigo" e "id" en cualquier orden.

## Operador de elección "|" en una secuencia de elementos

**EJEMPLO** En el siguiente documento XML, pueden aparecer cero o más elementos "articulo" que contengan un elemento "codigo" o un elemento "id", y obligatoriamente un elemento "nombre":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulos [
   <!ELEMENT articulos (articulo)*>
   <!ELEMENT articulo ((codigo | id), nombre)>
   <!ELEMENT codigo (#PCDATA)>
   <!ELEMENT id (#PCDATA)>
   <!ELEMENT nombre (#PCDATA)>
]>
```

```
<articulos>
   <articulo>
      <codigo>AF-47</codigo>
      <nombre>Martillo</nombre>
   </articulo>
   <articulo>
      <id>2056</id>
      <nombre>Destornillador</nombre>
   </articulo>
</articulos>
```

## Secuencia de elementos en una lista de opciones

**EJEMPLO** En la DTD del siguiente documento XML se ha indicado que pueden aparecer cero o más elementos "localidad". En el caso de aparecer, cada uno de ellos contendrá los elementos "pais" y "ciudad", o alternativamente un elemento "codigo_postal":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE localidades [
   <!ELEMENT localidades (localidad)*>
   <!ELEMENT localidad ((pais, ciudad) | codigo_postal)>
   <!ELEMENT pais (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
   <!ELEMENT codigo_postal (#PCDATA)>
]>

<localidades>
   <localidad>
      <pais>España</pais>
      <ciudad>Valencia</ciudad>
   </localidad>
   <localidad>
      <codigo_postal>31015</codigo_postal>
   </localidad>
</localidades>
```

## **#PCDATA** en una lista de opciones permite contenido mixto

**EJEMPLO** Al utilizar el *operador de elección* (**|**) en una DTD, si una de las opciones es **#PCDATA**, esta debe escribirse en primer lugar:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulos [
   <!ELEMENT articulos (#PCDATA | codigo | id)*>
   <!ELEMENT codigo (#PCDATA)>
   <!ELEMENT id (#PCDATA)>
]>
```

```
<articulos>
   <id>8608</id>
   Teclado
   <codigo>AF-18</codigo>
   <codigo>AF-45</codigo>
   Disquetera
   <id>7552</id>
   <id>4602</id>
</articulos>
```

- Fíjese que, el elemento "articulos" de este documento, puede contener contenido mixto, es decir, texto y otros elementos.

**EJEMPLO** Véase, en este último ejemplo, que el elemento "provincia" puede aparecer cero o más veces, pudiendo contener contenido mixto:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE localidades [
   <!ELEMENT localidades (provincia*)>
   <!ELEMENT provincia (#PCDATA | ciudad | codigo_postal)*>
   <!ELEMENT ciudad (#PCDATA)>
   <!ELEMENT codigo_postal (#PCDATA)>
]>

<localidades>
   <provincia>
      Navarra
      <ciudad>Estella</ciudad>
      <codigo_postal>31015</codigo_postal>
      <ciudad>Tafalla</ciudad>
   </provincia>
   <provincia>
      Valencia
      <codigo_postal>46520</codigo_postal>
   </provincia>
</localidades>
```

# 5. Declaración de atributos

La sintaxis básica para declarar un atributo en una DTD es:

```
<!ATTLIST nombre-del-elemento nombre-del-atributo tipo-de-atributo
valor-del-atributo>
```

## Declaración de un atributo indicando un **`valor`** por defecto

**EJEMPLO** En la DTD del siguiente documento XML se ha indicado que el elemento "f1" puede tener el atributo "pais":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais CDATA "España">
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="Alemania">Sebastian Vettel</f1>
   <f1>Fernando Alonso</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

- Para el elemento "f1", **`pais`** es un atributo definido de tipo **`CDATA`** (*Character DATA*), es decir, su valor será una cadena de caracteres.
- Al no indicarse el país de *Fernando Alonso*, por defecto es **`"España"`**.
- Para *Sebastian Vettel*, al atributo **`pais`** se le ha asignado **`"Alemania"`**, que es un valor distinto al **`valor-del-atributo`**, que por defecto es **`"España"`**.

Al visualizar el documento XML en un navegador web, se verá algo parecido a:

## Declaración de varios atributos en un elemento

**EJEMPLO** En la DTD del siguiente documento XML se ha indicado que el elemento "f1" puede tener tres atributos (**pais**, **fecha_de_nacimiento** y **equipo**):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais CDATA "España">
      <!ATTLIST f1 fecha_de_nacimiento CDATA #IMPLIED>
      <!ATTLIST f1 equipo CDATA #REQUIRED>
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="Alemania" fecha_de_nacimiento="03/07/1987"
       equipo="Ferrari">Sebastian Vettel</f1>
   <f1 equipo="McLaren">Fernando Alonso</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

- Obsérvese que, en este ejemplo, el atributo **equipo** es obligatorio escribirlo, **#REQUIRED**. Mientras que, el atributo **fecha_de_nacimiento** es opcional, **#IMPLIED**.

En una DTD, cuando se declara más de un atributo para un elemento –como se ha hecho en este caso– no es necesario escribir varias veces **<!ATTLIST**, pudiéndose escribir, por ejemplo:

```
<!ATTLIST f1 pais CDATA "España"
             fecha_de_nacimiento CDATA #IMPLIED
             equipo CDATA #REQUIRED>
```

# 6. Tipos de declaración de atributos

En DTD, existen los siguientes tipos de declaración de atributos:

| **Tipos de declaración de atributos en DTD** | |
|---|---|
| ***Valor*** | ***Significado*** |
| **valor** entre comillas dobles (") o simples (').| El atributo tiene un valor por defecto. |
| **#REQUIRED** | El atributo es obligatorio escribirlo. |
| **#IMPLIED** | El atributo es opcional escribirlo. |
| **#FIXED valor** entre comillas dobles (") o simples ('). | El valor del atributo es fijo. |

En el apartado anterior de este tutorial *"5. Declaración de atributos"*, se muestra un ejemplo declaración de atributo con un valor por defecto.

Respecto a los otros tipos de declaración de atributos (**#REQUIRED**, **#IMPLIED** y **#FIXED valor**), en los siguientes apartados se muestran ejemplos:

- *6.1. Atributo obligatorio -* ***#REQUIRED***
- *6.2. Atributo opcional -* ***#IMPLIED***
- *6.3. Atributo con valor fijo -* ***#FIXED valor***

# 6.1. Atributo obligatorio - #REQUIRED

**EJEMPLO** En la DTD interna del siguiente documento XML se ha declarado un atributo indicando que es obligatorio, es decir, **#REQUIRED**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais CDATA #REQUIRED>
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="Alemania">Sebastian Vettel</f1>
   <f1>Fernando Alonso</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

- En este ejemplo, es obligatorio escribir el atributo **pais** en los elementos "f1". Por tanto, aunque el documento XML está bien formado, habría que indicar el **pais** de *Fernando Alonso* para que fuese válido.

```
   <f1 pais="España">Fernando Alonso</f1>
```

- Por otra parte, fíjese que, de *Rafael Nadal* no es obligatorio indicar su país, ni se puede hacer.

# 6.2. Atributo opcional - #IMPLIED

**EJEMPLO** En una DTD, para especificar que un atributo es opcional escribirlo o no, hay que indicarlo mediante **#IMPLIED**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais CDATA #IMPLIED>
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="Alemania">Sebastian Vettel</f1>
   <f1>Fernando Alonso</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

- En este caso, el atributo **`pais`** es opcional para los elementos "f1" que aparezcan en el documento XML. Así pues, obsérvese que, aunque no se ha indicado el país de *Fernando Alonso*, el documento es válido.

# 6.3. Atributo con valor fijo - `#FIXED valor`

**EJEMPLO** Cuando en una DTD, se quiere declarar un atributo que tome un valor fijo, esto se puede hacer con **`#FIXED valor`**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais CDATA #FIXED "España">
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="España">Carlos Sainz</f1>
   <f1>Fernando Alonso</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

- Según la DTD de este documento XML, todos los elementos "f1" que aparezcan tendrán el atributo **`pais`** con el valor **`"España"`**. Por tanto, no es necesario haberlo escrito para *Carlos Sainz*. De hecho, si se hubiese escrito otro valor, el documento no sería válido.

De modo que, para este caso, al visualizar el documento XML en un navegador web, se mostrará algo parecido a:


# 7. Tipos de atributos

En DTD, existen los siguientes tipos de atributos:

| **Tipos de atributos en DTD** | |
|---|---|
| ***Tipo*** | ***Descripción*** |
| **CDATA** | (*Character DATA*) El valor son datos de tipo carácter, es decir, texto. |
| **Enumerado** | El valor puede ser uno de los pertenecientes a una lista de valores escritos entre *paréntesis* "**()**" y separados por el carácter "**\|**". |
| **ID** | El valor es un identificador único. |
| **IDREF** | El valor es un identificador que tiene que existir en otro atributo **ID** del documento XML. |
| **IDREFS** | El valor es una lista de valores que existan en otros atributos **ID** del documento XML, separados por espacios en blanco. |
| **NMTOKEN** | El valor es una cadena de caracteres, pudiendo contener letras minúsculas, letras mayúsculas, números, puntos "**.**", guiones medios "**-**", guiones bajos "**_**" o el carácter dos puntos "**:**". |
| **NMTOKENS** | El valor puede contener uno o varios valores de tipo **NMTOKEN** separados por espacios en blanco. |
| **NOTATION** | El valor es el nombre de una notación. |
| **ENTITY** | El valor es el nombre de una entidad. |
| **ENTITIES** | El valor puede contener uno o varios valores de tipo **ENTITY** separados por espacios en blanco. |
| **Especiales** | Existen dos atributos especiales: **xml:lang** y **xml:space**. |

# 7.1. Atributos de tipo CDATA

En una DTD, un atributo de tipo **`CDATA`** (*Character DATA*), es aquel cuyo valor puede ser una cadena de caracteres (texto).

**EJEMPLO** En la DTD del siguiente documento XML, el atributo **`pais`** del elemento "ciudad" ha sido declarado de tipo **`CDATA`**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ciudades [
   <!ELEMENT ciudades (ciudad)*>
   <!ELEMENT ciudad (#PCDATA)>
      <!ATTLIST ciudad pais CDATA #REQUIRED>
]>

<ciudades>
  <ciudad pais="Italia">Roma</ciudad>
  <ciudad pais="Francia">París</ciudad>
  <ciudad pais="Alemania">Berlín</ciudad>
  <ciudad pais="">Viena</ciudad>
</ciudades>
```

- Obsérvese que, el valor del atributo **`pais`** puede estar vacío.

# 7.2. Atributos de tipo enumerado

En una DTD, se puede declarar un atributo de tipo enumerado indicando que **su valor puede ser uno de los pertenecientes a una lista de valores** escritos entre paréntesis "**`()`**" y separados por el carácter "**`|`**".

**EJEMPLO** En la DTD del siguiente documento XML, los valores posibles del atributo **`pais`** son **`"ESP"`**, **`"FRA"`**, **`"ITA"`** y **`"ALE"`**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais (ESP | FRA | ITA | ALE) "ESP">
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="ALE">Sebastian Vettel</f1>
   <f1>Fernando Alonso</f1>
   <f1 pais="ESP">Carlos Sainz</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

- Véase que, en este caso, se ha especificado **"ESP"** como valor por defecto, siendo obligatorio que esté en la lista de valores escritos entre *paréntesis* "**()**".

Al visualizar este documento en un navegador web, en pantalla se verá:


Si se quiere definir el atributo **pais** obligatorio, habría que escribir:

```
<!ATTLIST f1 pais (ESP | FRA | ITA | ALE) #REQUIRED>
```

Por tanto, para Fernando Alonso se tendría que escribir:

```
<f1 pais="ESP">Fernando Alonso</f1>
```

# 7.3. Atributos de tipo ID

En una DTD, los atributos declarados **ID** son aquellos que solo pueden tomar un valor único (identificador) para cada elemento.

**EJEMPLO** En la DTD del siguiente documento XML, el atributo **codigo** del elemento "f1" ha sido declarado de tipo **ID**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 codigo ID #REQUIRED>
]>
```

```
<deportistas>
   <f1 codigo="ALO">Fernando Alonso</f1>
   <f1 codigo="VET">Sebastian Vettel</f1>
</deportistas>
```

Hay que tener en cuenta que:

- Los valores de atributos **ID**, tienen que cumplir las mismas normas de sintaxis utilizadas para escribir nombres en XML.
- Cada elemento escrito en un documento XML, solo puede tener un atributo **ID**.
- En un documento XML, no pueden escribirse dos elementos que tengan el mismo valor en un atributo **ID**, aunque dicho atributo sea distinto.
- Todo atributo declarado de tipo **ID** tiene que ser **#IMPLIED** (opcional) o **#REQUIRED** (obligatorio).

# 7.4. Atributos de tipo IDREF

En una DTD, los atributos declarados **IDREF** son aquellos cuyo valor tiene que existir en otro atributo **ID** del documento XML.

**EJEMPLO** En la DTD del siguiente documento XML, se indica que los elementos "pelicula" que se escriban, deben incluir el atributo **direccion**, cuyo valor estará asignado a un atributo **ID** de otro elemento del documento. En este caso, el valor estará asignado a un atributo **coddir** de un elemento "director":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cine [
   <!ELEMENT cine (directores, peliculas)>
   <!ELEMENT directores (director)*>
      <!ELEMENT director (#PCDATA)>
         <!ATTLIST director coddir ID #REQUIRED>
   <!ELEMENT peliculas (pelicula)*>
      <!ELEMENT pelicula (#PCDATA)>
         <!ATTLIST pelicula direccion IDREF #REQUIRED>
]>

<cine>
   <directores>
      <director coddir="CE">Clint Eastwood</director>
      <director coddir="JC">James Cameron</director>
   </directores>
   <peliculas>
      <pelicula direccion="JC">Avatar</pelicula>
      <pelicula direccion="CE">Mystic River</pelicula>
      <pelicula direccion="JC">Titanic</pelicula>
   </peliculas>
</cine>
```

- Obsérvese que, por ejemplo, para la película *Titanic* se ha indicado en su atributo **direccion** el valor **"JC"**, que es el valor del atributo **coddir** del director *James Cameron*.
- En este documento XML, el atributo de tipo **IDREF** se ha definido obligatorio, **#REQUIRED**. Pero, a un atributo **IDREF** también se le puede especificar un valor por defecto, un valor fijo o que sea opcional escribirlo, **#IMPLIED**.

# 7.5. Atributos de tipo IDREFS

En una DTD, los atributos declarados **IDREFS** son aquellos cuyo valor puede ser una lista de valores que existan en otros atributos **ID** del documento XML.

**EJEMPLO** En la DTD del siguiente documento XML, se indica que el valor del atributo **filmografia** de un elemento "director", puede ser una lista de valores de atributos **ID**. En este caso, una lista de valores escritos en el atributo **codpel** de los elementos "pelicula" que aparezcan en el documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cine [
   <!ELEMENT cine (peliculas, directores)>
   <!ELEMENT peliculas (pelicula)*>
      <!ELEMENT pelicula (#PCDATA)>
         <!ATTLIST pelicula codpel ID #REQUIRED>
   <!ELEMENT directores (director)*>
      <!ELEMENT director (#PCDATA)>
         <!ATTLIST director filmografia IDREFS #REQUIRED>
]>

<cine>
   <peliculas>
      <pelicula codpel="P1">Avatar</pelicula>
      <pelicula codpel="P2">Mystic River</pelicula>
      <pelicula codpel="P3">The Terminator</pelicula>
      <pelicula codpel="P4">Titanic</pelicula>
   </peliculas>
   <directores>
      <director filmografia="P2">Clint Eastwood</director>
      <director filmografia="P1 P3 P4">James Cameron</director>
   </directores>
</cine>
```

- Obsérvese que, los valores de la lista de valores de un atributo **IDREFS**, se escriben separados por un espacio en blanco.

# 7.6. Atributos de tipo NMTOKEN

En una DTD, los atributos declarados **NMTOKEN** son aquellos cuyo valor será una cadena de caracteres, pudiendo contener letras minúsculas, letras mayúsculas, números, puntos "**.**", guiones medios "**-**", guiones bajos "**_**" o el carácter dos puntos "**:**".

**EJEMPLO** En la DTD del siguiente documento XML, el atributo **clave** del elemento "usuario" ha sido declarado de tipo **NMTOKEN**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE usuarios [
  <!ELEMENT usuarios (usuario)*>
  <!ELEMENT usuario (#PCDATA)>
  <!ATTLIST usuario clave NMTOKEN #REQUIRED>
]>

<usuarios>
  <usuario clave="123456789">Ana</usuario>
  <usuario clave="ab-c-d-fg">Iker</usuario>
  <usuario clave="A1_B2..C3">Elsa</usuario>
</usuarios>
```

- En el valor de un atributo **NMTOKEN** no se pueden escribir *espacios en blanco* ni caracteres especiales, tales como: *, $, %, &, ?, @...

# 7.7. Atributos de tipo NMTOKENS

En una DTD, los atributos declarados **NMTOKENS** son aquellos cuyo valor puede contener uno o varios valores de tipo **NMTOKEN** separados por espacios en blanco.

**EJEMPLO** En la DTD del siguiente documento XML, el atributo **codigos** del elemento "usuario" ha sido declarado de tipo **NMTOKENS**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE usuarios [
  <!ELEMENT usuarios (usuario)*>
  <!ELEMENT usuario (#PCDATA)>
  <!ATTLIST usuario codigos NMTOKENS #REQUIRED>
]>

<usuarios>
  <usuario codigos="1234 567 89">Ana</usuario>
  <usuario codigos="ab c-d fg">Iker</usuario>
  <usuario codigos="A1:B2">Elsa</usuario>
</usuarios>
```

- Obsérvese que, los valores escritos en el atributo **codigos**, se escriben separados por espacios en blanco.

# 7.8. Atributos de tipo NOTATION

En una DTD, los atributos declarados **NOTATION** son aquellos cuyo valor puede ser el nombre de una notación.

**EJEMPLO** En la DTD del siguiente documento XML, se indica que los elementos "animal" que se escriban, deben incluir opcionalmente el atributo **tipo_de_imagen**, cuyo valor será una notación (**gif**, **jpg** o **png**):

```
<?xml version = "1.0" encoding="UTF-8"?>
<!DOCTYPE animales [
   <!ELEMENT animales (animal)*>
   <!ELEMENT animal (nombre)>
   <!ELEMENT nombre (#PCDATA)>
   <!ATTLIST animal
      imagen CDATA #IMPLIED
      tipo_de_imagen NOTATION (jpg | gif | png) #IMPLIED>

   <!NOTATION gif SYSTEM "image/gif">
   <!NOTATION jpg SYSTEM "image/jpeg">
   <!NOTATION png SYSTEM "image/png">
]>

<animales>
    <animal imagen="ballena-azul.gif" tipo_de_imagen="gif">
        <nombre>Ballena</nombre>
    </animal>
    <animal imagen="leon-dormido.png" tipo_de_imagen="png">
        <nombre>Leon</nombre>
    </animal>
</animales>
```

- En este ejemplo, las notaciones **gif**, **jpg** y **png** son declaraciones de los siguientes tipos MIME (*Multipurpose Internet Mail Extensions*, Extensiones Multipropósito de Correo de Internet): *image/gif*, *image/jpeg* e *image/png*.

**EJEMPLO** En la DTD del siguiente documento XML, se indica que los elementos "programa" que se escriban, deben incluir obligatoriamente el atributo **lenguaje**, cuyo valor será una notación (**csharp** o **java**):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE programas [
   <!ELEMENT programas (programa)*>
   <!ELEMENT programa (#PCDATA)>
   <!ATTLIST programa lenguaje NOTATION (csharp|java) #REQUIRED>

   <!NOTATION csharp PUBLIC "CSharp 5.0">
   <!NOTATION java PUBLIC "Java 8.0">
]>
```

```
<programas>
   <programa lenguaje="java"><!-- Código fuente 1. --></programa>
   <programa lenguaje="java"><!-- Código fuente 2. --></programa>
   <programa lenguaje="csharp"><!-- Código fuente 3. --></programa>
</programas>
```

- *CSharp 5.0* y *Java 8.0* son identificadores públicos.

# 7.9. Atributos de tipo `ENTITY`

En una DTD, los atributos declarados **`ENTITY`** son aquellos cuyo valor puede ser el nombre de una entidad.

## Uso de `ENTITY` y `NOTATION`

**EJEMPLO** En la DTD del siguiente documento XML, se indica que los elementos "animal" que se escriban, tiene que incluir obligatoriamente el atributo **`imagen`**, cuyo valor será una entidad:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE animales [
   <!ELEMENT animales (animal)*>
   <!ELEMENT animal EMPTY>
   <!ATTLIST animal imagen ENTITY #REQUIRED>

   <!ENTITY ballena SYSTEM "ballena.gif" NDATA gif>
   <!ENTITY delfin SYSTEM "delfin.gif" NDATA gif>

   <!NOTATION gif SYSTEM "image/gif">
]>

<animales>
   <animal imagen="ballena"/>
   <animal imagen="delfin"/>
</animales>
```

- En la DTD de este ejemplo se está indicando que los valores –datos– de las entidades (**`ballena`** y **`delfin`**) van a ser cargados desde una URI (*Uniform Resource Identifier*, Identificador Uniforme de Recurso). En este caso, se hace referencia a los archivos externos *"ballena.gif"* y *"delfin.gif"*.
- Con **`NDATA`** (*Notation Data*) se ha asociado a las entidades **`ballena`** y **`delfin`** con la notación **`gif`**.
- La notación **`gif`** es una declaración del tipo *MIME image/gif*.

# 7.10. Atributos de tipo ENTITIES

En una DTD, los atributos declarados **ENTITIES** son aquellos cuyo valor puede contener uno o varios valores de tipo **ENTITY** separados por espacios en blanco.

## Uso de ENTITIES y NOTATION

**EJEMPLO** En la DTD del siguiente documento XML, el atributo **imagenes** del elemento "grupos" ha sido declarado de tipo **ENTITIES**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE animales [
   <!ELEMENT animales (grupos)*>
   <!ELEMENT grupos EMPTY>
   <!ATTLIST grupos imagenes ENTITIES #REQUIRED>

   <!ENTITY ballena SYSTEM "ballena.gif" NDATA gif>
   <!ENTITY delfin SYSTEM "delfin.gif" NDATA gif>
   <!ENTITY elefante SYSTEM "elefante.gif" NDATA gif>
   <!ENTITY leon SYSTEM "leon.gif" NDATA gif>
   <!ENTITY oso SYSTEM "oso.gif" NDATA gif>

   <!NOTATION gif SYSTEM "image/gif">
]>

<animales>
   <grupos imagenes="ballena"/>
   <grupos imagenes="ballena delfin"/>
   <grupos imagenes="elefante leon oso"/>
   <grupos imagenes="ballena elefante"/>
</animales>
```

- En la DTD de este ejemplo se está indicando que los valores –datos– de las entidades (**ballena**, **delfin**, **elefante**, **leon** y **oso**) van a ser cargados desde una URI (*Uniform Resource Identifier*, Identificador Uniforme de Recurso). En este caso, se hace referencia a los archivos externos *"ballena.gif"*, *"delfin.gif"*, *"elefante.gif"*, *"leon.gif"* y *"oso.gif"*.
- Con **NDATA** (*Notation Data*) se ha asociado a las entidades **ballena**, **delfin**, **elefante**, **leon** y **oso** con la notación **gif**.
- La notación **gif** es una declaración del tipo *MIME image/gif*.

# 7.11. Atributos especiales

En DTD existen dos tipos de atributos especiales (predefinidos), llamados: **xml:lang** y **xml:space**.

## Uso del atributo **xml:lang**

En una DTD, el atributo **xml:lang** permite indicar el idioma del contenido y de los valores de los atributos de un elemento declarado. De forma que, cuando se utiliza **xml:lang** en un elemento, el idioma especificado afecta a todos los valores de sus posibles atributos y a todo su contenido, incluyendo a sus posibles sucesores a menos que se indique lo contrario con otra instancia de **xml:lang**.

**EJEMPLO** En la DTD del siguiente documento XML, con el atributo **xml:lang** se ha indicado el idioma de los elementos "sigla" y "traduccion":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE siglas [
   <!ELEMENT siglas (sigla)*>
   <!ELEMENT sigla (significado, traduccion)>
      <!ATTLIST sigla letras CDATA #REQUIRED>
      <!ATTLIST sigla xml:lang CDATA "en">
   <!ELEMENT significado (#PCDATA)>
   <!ELEMENT traduccion (#PCDATA)>
      <!ATTLIST traduccion xml:lang CDATA #FIXED "es">
]>

<siglas>
   <sigla letras="ANSI">
      <significado>American National Standards Institute</significado>
      <traduccion>Instituto Nacional Estadounidense de Estándares</traduccion>
   </sigla>
   <sigla letras="ISO">
      <significado>International Organization for Standardization</significado>
      <traduccion>Organización Internacional de Normalización</traduccion>
   </sigla>
   <sigla letras="CERN" xml:lang="fr">
      <significado>Conseil Européen pour la Recherche Nucléaire</significado>
      <traduccion>Organización Europea para la Investigación Nuclear</traduccion>
   </sigla>
</siglas>
```

- Inicialmente, para el elemento "sigla" se ha indicado el idioma inglés, **"en"**, por defecto.
- No obstante, después se ha fijado el valor **"es"**, del español, para el atributo **xml:lang** del elemento "traduccion".
- Por otra parte, para el CERN se ha especificado que el idioma es el francés, **"fr"**.

## Uso del atributo **xml:space**

En una DTD, el atributo **xml:space** permite indicar que los espacios en blanco, las tabulaciones y los retornos de carro que aparezcan en el contenido (texto) de un elemento –y sus sucesores a menos que se indique lo contrario con otra instancia de **xml:space**– tienen que ser preservados. Este atributo siempre tiene que ser declarado de tipo enumerado, siendo **"default"**, **"preserve"** o ambos, los posibles valores pertenecientes a la lista de valores que se indiquen entre paréntesis **"()"**.

**EJEMPLO** En la DTD del siguiente documento XML, con el atributo **xml:space** se ha indicado que, por defecto, los espacios que se escriban en el contenido de los elementos "programa" del documento, deben preservarse. Ahora bien, en la declaración de **xml:space** se ha indicado que su valor podría ser también **"default"**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE programas [
  <!ELEMENT programas (programa)*>
  <!ELEMENT programa (#PCDATA)>
  <!ATTLIST programa xml:space (default|preserve) "preserve">
]>
<programas>
  <programa>/* Programa: Hola mundo */

#include &lt;conio.h&gt;
#include &lt;stdio.h&gt;

int main()
{
    printf( &quot;Hola mundo.&quot; );

    getch(); /* Pausa */

    return 0;
}</programa>
  <programa>/* Programa: Calificación según nota */

#include &lt;conio.h&gt;
#include &lt;stdio.h&gt;

int main()
{
    float nota;

    printf( &quot;\n   Introduzca nota (real): &quot; );
    scanf( &quot;%f&quot;, &amp;nota );

    if ( nota &amp;gt;= 5 )
        printf( &quot;\n   APROBADO&quot; );
    else
        printf( &quot;\n   SUSPENDIDO&quot; );

    getch(); /* Pausa */

    return 0;
}</programa>
</programas>
```

- En este ejemplo, los espacios en blanco, las tabulaciones y los retornos de carro de los dos programas escritos tienen que preservarse.
- No obstante, tal y como está declarado el atributo **`xml:space`** del elemento "programa", se podría asignar el valor **`"default"`** a **`xml:space`** en cualquier **`programa`**. En tal caso, sería el programa que procese el documento, el que decidiese qué tratamiento hacer a los espacios en blanco, las tabulaciones y los retornos de carro.

# 8. Declaración de entidades

En una DTD se pueden declarar entidades generales y paramétricas (de parámetro). Las entidades generales pueden ser:

- Entidades generales internas analizables (*parsed*).
- Entidades generales externas analizables (*parsed*).
- Entidades generales externas no analizables (*unparsed*).

Por otro lado, las entidades paramétricas pueden ser:

- Entidades paramétricas internas analizables (*parsed*).
- Entidades paramétricas externas analizables (*parsed*).

Las entidades generales pueden utilizarse en el cuerpo de un documento XML y en su DTD. Sin embargo, las entidades paramétricas solo pueden utilizarse dentro de la DTD.

# 8.1. Entidades generales internas analizables

Para declarar una entidad general interna analizable (*parsed*) en una DTD, se utiliza la siguiente sintaxis:

```
<!ENTITY nombre-de-la-entidad "valor-de-la-entidad">
```

**EJEMPLO** En la DTD del siguiente documento XML, se han declarado tres entidades (**`escritor`**, **`obra`** y **`fecha`**):

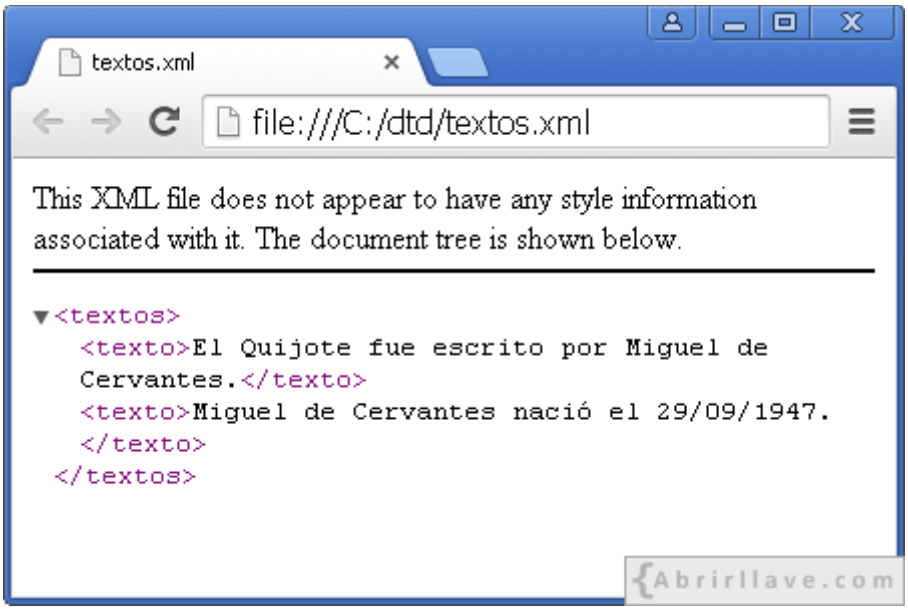
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE textos [
   <!ELEMENT textos (texto)+>
   <!ELEMENT texto (#PCDATA)>

   <!ENTITY escritor "Miguel de Cervantes">
   <!ENTITY obra "El Quijote">
   <!ENTITY fecha "29/09/1947">
]>
```

```
<textos>
   <texto>&obra; fue escrito por &escritor;.</texto>
   <texto>&escritor; nació el &fecha;.</texto>
</textos>
```

- Obsérvese que, para referenciar a las entidades, se ha utilizado la sintaxis:

```
&nombre-de-la-entidad;
```

Si este documento XML se visualizase en un navegador web, se vería algo parecido a:




# 8.2. Entidades generales externas analizables

En una DTD se pueden declarar dos tipos de entidades generales externas analizables (*parsed*): privadas y públicas. Para las privadas se utiliza **SYSTEM**, y para las públicas **PUBLIC**. La sintaxis en cada caso es:

```
<!ENTITY nombre-de-la-entidad SYSTEM "URI">
```

```
<!ENTITY nombre-de-la-entidad PUBLIC "identificador-público" "URI">
```

## Entidades generales externas analizables privadas - SYSTEM

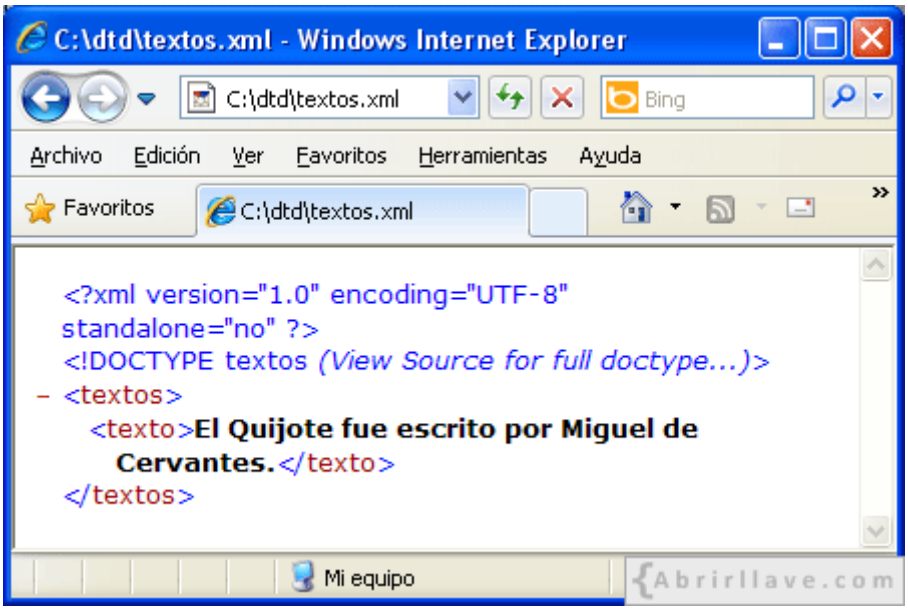
**EJEMPLO** En la DTD del siguiente documento XML, se ha declarado la entidad **escritor**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE textos [
   <!ELEMENT textos (texto)+>
   <!ELEMENT texto (#PCDATA)>

   <!ENTITY escritor SYSTEM "escritor.txt">
]>

<textos>
   <texto>El Quijote fue escrito por &escritor;.</texto>
</textos>
```

Suponiendo que el archivo ***"escritor.txt"*** contenga:

```
Miguel de Cervantes
```

En un navegador web (por ejemplo en *Internet Explorer 8*) se podrá ver:




## Entidades generales externas analizables públicas - **PUBLIC**

**EJEMPLO** Para declarar **escritor** como entidad general externa analizable pública, se puede escribir:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE textos [
   <!ELEMENT textos (texto)+>
   <!ELEMENT texto (#PCDATA)>

   <!ENTITY escritor PUBLIC "-//W3C//TEXT escritor//EN"
   "http://www.abrirllave.com/dtd/escritor.txt">
]>
```

```
<textos>
   <texto>El Quijote fue escrito por &escritor;.</texto>
</textos>
```

# 8.3. Entidades generales externas no analizables

En una DTD, al igual que ocurre con las entidades generales externas analizables, se pueden declarar dos tipos de entidades generales externas no analizables (*unparsed*): privadas y públicas. Para las privadas se utiliza **SYSTEM**, y para las públicas **PUBLIC**. La sintaxis en cada caso es:

```
<!ENTITY nombre-de-la-entidad SYSTEM "URI" NDATA notación>
```

```
<!ENTITY nombre-de-la-entidad PUBLIC "identificador-público" "URI"
NDATA notación>
```

Las entidades no analizables pueden contener cualquier tipo de datos (no XML). Por tanto, pueden hacer referencia a datos que un procesador XML no tiene porqué analizar, como por ejemplo una imagen.

## Entidades generales externas no analizables privadas - **SYSTEM**

**EJEMPLO** En la DTD del siguiente documento XML, se indica que el elemento "imagen" que se escriba, tiene que incluir obligatoriamente el atributo **fuente**, cuyo valor será una entidad:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE imagen [
   <!ELEMENT imagen EMPTY>
      <!ATTLIST imagen fuente ENTITY #REQUIRED>

   <!ENTITY logo SYSTEM "logo.gif" NDATA gif>

   <!NOTATION gif SYSTEM "image/gif">
]>

<imagen fuente="logo"/>
```

- En la DTD de este ejemplo se está indicando que el valor –datos– de la entidad **logo** va a ser cargado desde una URI. En este caso, se hace referencia al archivo *"logo.gif"*.
- Con **NDATA** (*Notation Data*) se indica que la entidad no es analizable y, en este caso, se ha asociado a la entidad **logo** con la notación **gif**.
- La notación **gif** es una declaración del tipo *MIME image/gif*.

## Entidades generales externas no analizables públicas - **PUBLIC**

**EJEMPLO** Para declarar **logo** como entidad pública, se puede escribir:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE imagen [
   <!ELEMENT imagen EMPTY>
      <!ATTLIST imagen fuente ENTITY #REQUIRED>

   <!ENTITY logo PUBLIC "-//W3C//GIF logo//EN"
   "http://www.abrirllave.com/dtd/logo.gif" NDATA gif>

   <!NOTATION gif SYSTEM "image/gif">
]>

<imagen fuente="logo"/>
```

- Véase que, se referencia al archivo "http://www.abrirllave.com/dtd/logo.gif".

# 8.4. Entidades paramétricas internas analizables

Para declarar una entidad paramétrica (de parámetro) interna analizable (*parsed*) en una DTD, se utiliza la siguiente sintaxis:

```
<!ENTITY % nombre-de-la-entidad "valor-de-la-entidad">
```

**EJEMPLO** La DTD del siguiente documento XML es externa, habiéndose escrito esta en el archivo ***"persona.dtd"***:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona SYSTEM "persona.dtd">
<persona>
   <nombre>Iker</nombre>
   <mayor_de_edad/>
   <ciudad>Pamplona</ciudad>
</persona>
```

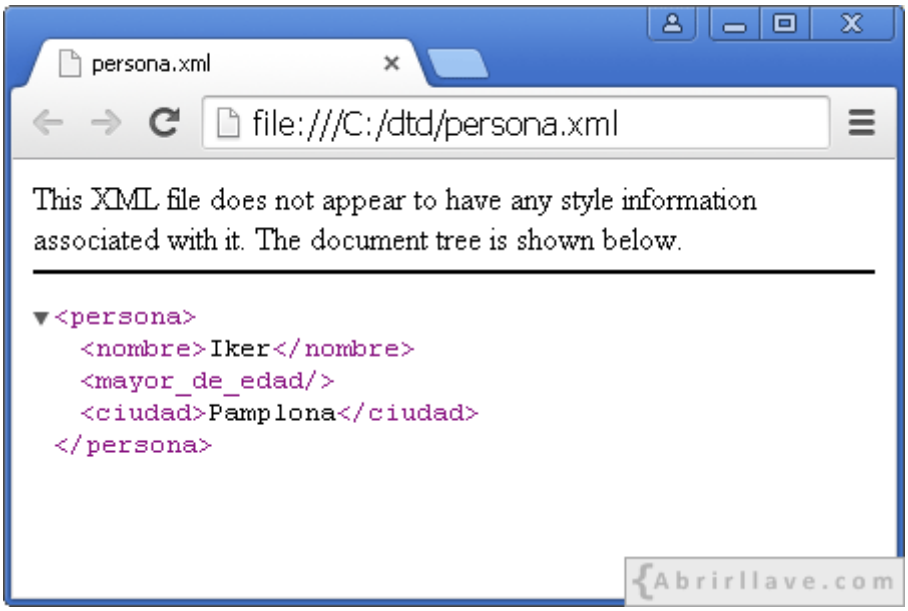
El contenido del archivo ***"persona.dtd"*** podría ser:

```
<!ENTITY % p "(#PCDATA)">
<!ELEMENT persona (nombre, mayor_de_edad?, ciudad)>
<!ELEMENT nombre %p;>
<!ELEMENT mayor_de_edad EMPTY>
<!ELEMENT ciudad %p;>
```

- Obsérvese que, en la DTD se ha declarado la entidad paramétrica **p** y, para referenciarla, se utiliza la sintaxis:

```
%nombre-de-la-entidad;
```

Si este documento XML se visualizase en un navegador web, se vería algo parecido a:




## Las entidades de parámetro se declaran antes de referenciarlas

En una DTD las entidades paramétricas tienen que declararse antes de ser referenciadas. Por tanto, no sería correcto haber escrito, por ejemplo:

```
<!ELEMENT persona (nombre, mayor_de_edad?, ciudad)>
<!ELEMENT nombre %p;>
<!ELEMENT mayor_de_edad EMPTY>
<!ELEMENT ciudad %p;>

<!ENTITY % p "(#PCDATA)">
```

## A una entidad paramétrica interna no se le puede referenciar en una DTD interna

Las entidades paramétricas internas pueden declararse en DTD internas o externas. Sin embargo, no pueden referenciarse desde una DTD interna. En consecuencia, el siguiente documento no sería válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ENTITY % p "(#PCDATA)">

   <!ELEMENT persona (nombre, mayor_de_edad?, ciudad)>
   <!ELEMENT nombre %p;>
   <!ELEMENT mayor_de_edad EMPTY>
   <!ELEMENT ciudad %p;>
]>
```

```
<persona>
   <nombre>Iker</nombre>
   <mayor_de_edad/>
   <ciudad>Pamplona</ciudad>
</persona>
```

## Declaración de una entidad paramétrica en la DTD interna de un documento XML y referenciada en la DTD externa

Ahora bien, sí sería válido el siguiente documento XML, donde internamente se declara la entidad paramétrica **p**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona SYSTEM "persona.dtd" [
   <!ENTITY % p "(#PCDATA)">
]>

<persona>
   <nombre>Iker</nombre>
   <mayor_de_edad/>
   <ciudad>Pamplona</ciudad>
</persona>
```

En este caso, el contenido del archivo ***"persona.dtd"*** podría ser:

```
<!ELEMENT persona (nombre, mayor_de_edad?, ciudad)>
<!ELEMENT nombre %p;>
<!ELEMENT mayor_de_edad EMPTY>
<!ELEMENT ciudad %p;>
```

# 8.5. Entidades paramétricas externas analizables

En una DTD se pueden declarar dos tipos de entidades paramétricas externas analizables (*parsed*): privadas y públicas. Para las privadas se utiliza **SYSTEM**, y para las públicas **PUBLIC**. La sintaxis en cada caso es:

```
<!ENTITY % nombre-de-la-entidad SYSTEM "URI">
%nombre-de-la-entidad;
```

```
<!ENTITY % nombre-de-la-entidad PUBLIC "identificador-público" "URI">
%nombre-de-la-entidad;
```

## Entidades paramétricas externas analizables privadas - **SYSTEM**

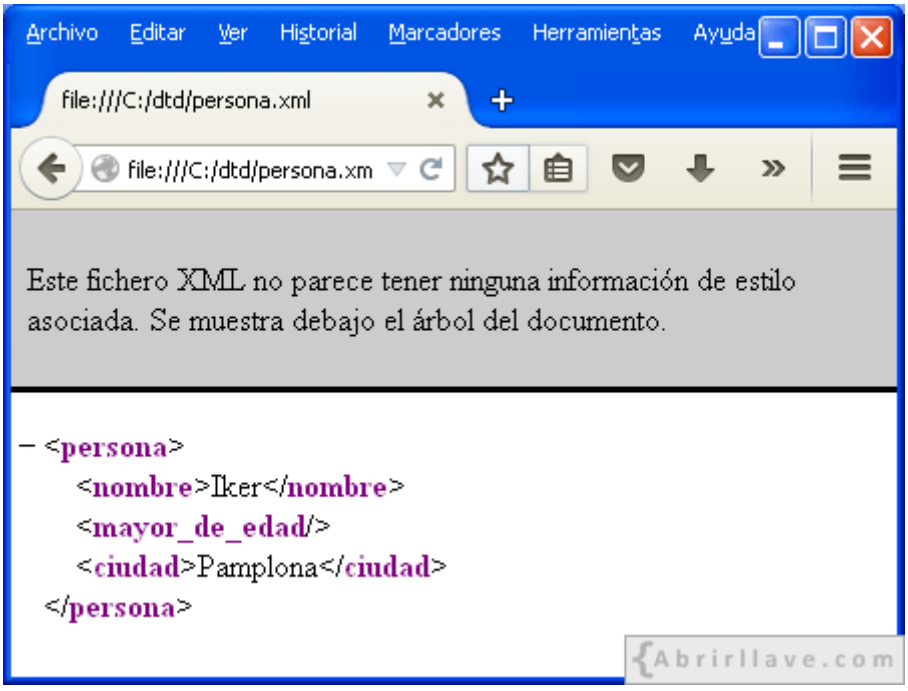
**EJEMPLO** En la DTD del siguiente documento XML, se ha declarado la entidad **persona**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona [
   <!ENTITY % persona SYSTEM "persona.dtd">
   %persona;
]>

<persona>
   <nombre>Iker</nombre>
   <mayor_de_edad/>
   <ciudad>Pamplona</ciudad>
</persona>
```

Suponiendo que el archivo ***"persona.dtd"*** contenga:

```
<!ELEMENT persona (nombre, mayor_de_edad?, ciudad)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT mayor_de_edad EMPTY>
<!ELEMENT ciudad (#PCDATA)>
```

En un navegador web (por ejemplo en *Mozilla Firefox*) se podrá ver:




## Entidades paramétricas externas analizables públicas - **PUBLIC**

**EJEMPLO** Para declarar **persona** como entidad paramétrica externa analizable pública, se puede escribir:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona [
   <!ENTITY % persona PUBLIC "-//W3C//TEXT persona//EN"
   "http://www.abrirllave.com/dtd/persona.dtd">
   %persona;
]>

<persona>
   <nombre>Iker</nombre>
   <mayor_de_edad/>
   <ciudad>Pamplona</ciudad>
</persona>
```

# 8.6. Uso de una entidad dentro de otra

**EJEMPLO** En la DTD del siguiente documento XML, se han declarado dos entidades generales internas analizables (**color** y **frase**):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE frase [
   <!ELEMENT frase (#PCDATA)>

   <!ENTITY color "azul">
   <!ENTITY frase "El cielo es &color;.">
]>

<frase>&frase;</frase>
```

- Obsérvese que, la entidad **color** ha sido referenciada en el valor de la entidad **frase**. De forma que, si este documento XML se visualizase en un navegador web, se vería:

## Referencia circular o recursiva de entidades

**EJEMPLO** La DTD del siguiente documento XML no es correcta, ya que, la entidad **frase1** ha sido referenciada en el valor de la entidad **frase2**, y al revés también:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE frase [
   <!ELEMENT frase (#PCDATA)>

   <!ENTITY frase1 "Esta frase incluye a la &frase2;.">
   <!ENTITY frase2 "Esta frase incluye a la &frase1;.">
]>

<frase>&frase1;</frase>
```

Para que dicha DTD fuese correcta, habría que quitar una de las dos referencias a entidades. Por ejemplo escribiendo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE frase [
   <!ELEMENT frase (#PCDATA)>

   <!ENTITY frase1 "Esta frase incluye a la &frase2;.">
   <!ENTITY frase2 "segunda frase">
]>

<frase>&frase1;</frase>
```

# 9. Declaración de notaciones

En una DTD se pueden declarar dos tipos de notaciones: privadas y públicas. Para las privadas se utiliza **SYSTEM**, y para las públicas **PUBLIC**, pudiéndose utilizar las siguientes sintaxis:

```
<!NOTATION nombre-de-la-notación SYSTEM "identificador-del-
sistema">
```

```
<!NOTATION nombre-de-la-notación PUBLIC "identificador-público">
```

```
<!NOTATION nombre-de-la-notación PUBLIC "identificador-público"
"identificador-del-sistema">
```

## Notaciones para indicar el formato de entidades externas - Uso de **SYSTEM**

En la DTD de un documento XML, las notaciones se pueden utilizar para especificar el formato de entidades externas (datos no XML), como por ejemplo un archivo que contenga una imagen. Dichas entidades externas no las analizará un procesador XML, sino que serán tratadas por el programa que procese el documento.

**EJEMPLO** En la DTD del siguiente documento XML, se indica que los elementos "fruta" que se escriban, tienen que incluir obligatoriamente el atributo **foto**, cuyo valor será una entidad y, para indicar el formato de dicha entidad, se usa la notación **gif**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE frutas [
   <!ELEMENT frutas (fruta)*>
   <!ELEMENT fruta EMPTY>
   <!ATTLIST fruta foto ENTITY #REQUIRED>

   <!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
   <!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>

   <!NOTATION gif SYSTEM "image/gif">
]>

<frutas>
   <fruta foto="manzana"/>
   <fruta foto="naranja"/>
</frutas>
```

- En la DTD de este ejemplo se está indicando que los valores –datos– de las entidades (**manzana** y **naranaja**) van a ser cargados desde una URI (*Uniform Resource Identifier*, Identificador Uniforme de Recurso). En este caso, se hace referencia a los archivos externos *"manzana.gif"* y *"naranja.gif"*.
- Con **NDATA** (*Notation Data*) se ha asociado a las entidades **manzana** y **naranaja** con la notación **gif**.
- La notación **gif** es una declaración del tipo *MIME image/gif*.

**EJEMPLO** Si en el sistema existe, por ejemplo, un programa llamado *"procesadorGIF.exe"* en la carpeta *"aplicaciones"* capaz de procesar imágenes GIF (*Graphics Interchange Format*, Formato de Intercambio de Gráficos), también se podría escribir:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE frutas [
   <!ELEMENT frutas (fruta)*>
   <!ELEMENT fruta EMPTY>
   <!ATTLIST fruta foto ENTITY #REQUIRED>

   <!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
   <!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>

   <!NOTATION gif SYSTEM "aplicaciones/procesadorGIF.exe">
]>
```

```
<frutas>
   <fruta foto="manzana"/>
   <fruta foto="naranja"/>
</frutas>
```

## Notación pública - **PUBLIC**

**EJEMPLO** En la declaración de una notación se puede indicar un identificador público estándar, como por ejemplo, *GIF 1.0*:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE frutas [
   <!ELEMENT frutas (fruta)*>
   <!ELEMENT fruta EMPTY>
   <!ATTLIST fruta foto ENTITY #REQUIRED>

   <!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
   <!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>

   <!NOTATION gif PUBLIC "GIF 1.0">
]>

<frutas>
   <fruta foto="manzana"/>
   <fruta foto="naranja"/>
</frutas>
```

**EJEMPLO** En la notación escrita en la DTD del siguiente documento XML, se ha declarado el tipo *MIME imagen/gif* e indicado el identificador público estándar *GIF 1.0*:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE frutas [
   <!ELEMENT frutas (fruta)*>
   <!ELEMENT fruta EMPTY>
   <!ATTLIST fruta foto ENTITY #REQUIRED>

   <!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
   <!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>

   <!NOTATION gif PUBLIC "GIF 1.0" "image/gif">
]>

<frutas>
   <fruta foto="manzana"/>
   <fruta foto="naranja"/>
</frutas>
```

## Atributos cuyo valor es el nombre de una notación

En una DTD, pueden existir elementos con atributos cuyo valor sea el nombre de una notación.

**EJEMPLO** En la DTD del siguiente documento XML, se indica que los elementos "documento" que se escriban, tienen que incluir obligatoriamente el atributo **`version`**, cuyo valor será una notación (**`h4`** o **`h5`**):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE documentos [
   <!ELEMENT documentos (documento)*>
   <!ELEMENT documento (#PCDATA)>
   <!ATTLIST documento version NOTATION (h4|h5) #REQUIRED>

   <!NOTATION h5 PUBLIC "HTML 5">
   <!NOTATION h4 PUBLIC "HTML 4.01">
]>

<documentos>
   <documento version="h4"><!-- Código del documento 1. --></documento>
   <documento version="h5"><!-- Código del documento 2. --></documento>
   <documento version="h5"><!-- Código del documento 3. --></documento>
   <documento version="h4"><!-- Código del documento 4. --></documento>
</documentos>
```

- *HTML 5* y *HTML 4.01* son identificadores públicos.

# 10. Secciones condicionales

En DTD externas se pueden definir las secciones **`IGNORE`** e **`INCLUDE`**, para ignorar o incluir declaraciones. Las sintaxis empleadas para ello son:

```
<![ IGNORE [ declaraciones ]]>
```

```
<![ INCLUDE [ declaraciones ]]>
```

El uso de las secciones condicionales suele estar ligado a entidades paramétricas.

**EJEMPLO** Si en un archivo llamado ***"persona.dtd"*** se ha escrito:

```
<![ %datos_basicos; [
   <!ELEMENT persona (nombre, edad)>
]]>

<![ %datos_ampliados; [
   <!ELEMENT persona (nombre, apellidos, edad, ciudad)>
]]>

<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT edad (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
```

El siguiente documento XML sería válido:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona SYSTEM "persona.dtd" [
   <!ENTITY % datos_basicos "INCLUDE">
   <!ENTITY % datos_ampliados "IGNORE">
]>

<persona>
   <nombre>Elsa</nombre>
   <edad>23</edad>
</persona>
```

También sería válido el documento:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona SYSTEM "persona.dtd" [
   <!ENTITY % datos_basicos "IGNORE">
   <!ENTITY % datos_ampliados "INCLUDE">
]>

<persona>
   <nombre>Ana</nombre>
   <apellidos>Sanz Tin</apellidos>
   <edad>19</edad>
   <ciudad>Pamplona</ciudad>
</persona>
```

- Obsérvese que, en este ejemplo, en los dos documentos XML asociados a la DTD externa escrita en el archivo ***"persona.dtd"***, se ha indicado –por medio de **IGNORE** e **INCLUDE**– si el elemento "persona" tiene que contener 2 ó 4 hijos, es decir, ("nombre" y "edad") o ("nombre", "apellidos", "edad" y "ciudad").

# 11. Espacios de nombres en DTD

**EJEMPLO** Dado el siguiente documento XML (visto en el apartado espacios de nombres del tutorial de XML de Abrirllave) bien formado, pero no validado, donde se utilizan dos espacios de nombres (*XML Namespaces*):

```
<?xml version="1.0" encoding="UTF-8"?>
<e1:ejemplo xmlns:e1="http://www.abrirllave.com/ejemplo1">

   <e1:carta>
      <e1:palo>Corazones</e1:palo>
      <e1:numero>7</e1:numero>
   </e1:carta>
```

```
    <e2:carta xmlns:e2="http://www.abrirllave.com/ejemplo2">
       <e2:carnes>
          <e2:filete_de_ternera precio="12.95"/>
          <e2:solomillo_a_la_pimienta precio="13.60"/>
       </e2:carnes>
       <e2:pescados>
          <e2:lenguado_al_horno precio="16.20"/>
          <e2:merluza_en_salsa_verde precio="15.85"/>
       </e2:pescados>
    </e2:carta>

</e1:ejemplo>
```

Se podría escribir dicho documento XML con una DTD interna como se muestra a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE e1:ejemplo [
   <!ELEMENT e1:ejemplo (e1:carta, e2:carta)>
      <!ATTLIST e1:ejemplo xmlns:e1 CDATA #FIXED "http://www.abrirllave.com/ejemplo1">
   <!ELEMENT e1:carta (e1:palo, e1:numero)>
   <!ELEMENT e1:palo (#PCDATA)>
   <!ELEMENT e1:numero (#PCDATA)>

   <!ELEMENT e2:carta (e2:carnes, e2:pescados)>
      <!ATTLIST e2:carta xmlns:e2 CDATA #FIXED "http://www.abrirllave.com/ejemplo2">
   <!ELEMENT e2:carnes (e2:filete_de_ternera, e2:solomillo_a_la_pimienta)>
   <!ELEMENT e2:pescados (e2:lenguado_al_horno, e2:merluza_en_salsa_verde)>
   <!ELEMENT e2:filete_de_ternera EMPTY>
      <!ATTLIST e2:filete_de_ternera precio CDATA #REQUIRED>
   <!ELEMENT e2:solomillo_a_la_pimienta EMPTY>
      <!ATTLIST e2:solomillo_a_la_pimienta precio CDATA #REQUIRED>
   <!ELEMENT e2:lenguado_al_horno EMPTY>
      <!ATTLIST e2:lenguado_al_horno precio CDATA #REQUIRED>
   <!ELEMENT e2:merluza_en_salsa_verde EMPTY>
      <!ATTLIST e2:merluza_en_salsa_verde precio CDATA #REQUIRED>
]>

<e1:ejemplo xmlns:e1="http://www.abrirllave.com/ejemplo1">

   <e1:carta>
      <e1:palo>Corazones</e1:palo>
      <e1:numero>7</e1:numero>
   </e1:carta>

   <e2:carta xmlns:e2="http://www.abrirllave.com/ejemplo2">
      <e2:carnes>
         <e2:filete_de_ternera precio="12.95"/>
         <e2:solomillo_a_la_pimienta precio="13.60"/>
      </e2:carnes>
      <e2:pescados>
         <e2:lenguado_al_horno precio="16.20"/>
         <e2:merluza_en_salsa_verde precio="15.85"/>
      </e2:pescados>
   </e2:carta>

</e1:ejemplo>
```

# 12. Comentarios

En una DTD asociada a un documento XML, se pueden escribir comentarios entre los caracteres "**<!--**" y "**-->**". Por ejemplo:

```
<!-- Esto es un comentario escrito en una DTD -->
```

**EJEMPLO** En la DTD interna del siguiente documento se han escrito dos comentarios:

```
<!-- Ejemplo de documento XML con comentarios en su DTD
interna, del Tutorial de DTD de Abrirllave.com -->
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ciudades [
   <!ELEMENT ciudades (ciudad*)>
   <!ELEMENT ciudad (#PCDATA)>
      <!-- pais es atributo del elemento ciudad -->
      <!ATTLIST ciudad pais CDATA #REQUIRED>
]>

<ciudades>
   <ciudad pais="Italia">Roma</ciudad>
   <ciudad pais="Francia">París</ciudad>
   <ciudad pais="Alemania">Berlín</ciudad>
   <ciudad pais="">Viena</ciudad>
</ciudades>
```

# 13. Recursos de DTD

- ***Tutorial de DTD***
  *http://www.abrirllave.com/dtd/*
- ***Chuleta de DTD***
  *http://www.abrirllave.com/dtd/chuleta-de-dtd.php*
- ***Ejercicios resueltos de DTD***
  *http://www.abrirllave.com/dtd/ejercicios-resueltos.php*
- ***Cómo validar con XML Copy Editor un documento XML asociado a una DTD***
  *http://www.abrirllave.com/dtd/como-validar-con-xml-copy-editor-un-documento-xml-asociado-a-una-dtd.php*