Validación por DTD

[1.1]introducción

Es indudablemente la más utilizada, pero también la menos coherente con las reglas XML. Su éxito se debe a que ya era una forma de validación reconocida antes de la aparición de XML, por lo que muchísimos productos software la reconocen desde hace mucho. La enorme compatibilidad que posee, ha determinado su éxito.

[1.2]cómo indicar que un XML cumple las reglas de un DTD

Si queremos que un documento XML cumpla las reglas de validación establecidas mediante el lenguaje DTD, tenemos varias opciones.

en el propio documento

Se puede definir la estructura que debe cumplir un documento XML mediante código DTD insertado en el propio documento. La desventaja evidente, es que estemos definiendo reglas que solo se aplican a un documento.

Un documento XML que defina internamente su DTD, simplemente escribe instrucciones DTD dentro del propio documento dentro de una etiqueta **DOCTYPE**. La sintaxis es:

Ejemplo:

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

Ese documento es válido ya que cumple las reglas establecidas directamente en la etiqueta **DOCTYPE**.

en un documento externo privado

En este caso la validación se crea en un documento DTD externo e independiente. De modo que cuando un documento queremos que cumpla las reglas de ese, se debe indicar la ruta (sea relativa o absoluta) al mismo.

La sintaxis de la etiqueta DOCTYPE que permite asignar un DTD privado a un documento XML es:

<!DOCTYPE raíz **SYSTEM "ruta_URL_al_DTD">**

Salvo que se desee crear un único documento con una validación DTD, lo lógico es utilizar la forma de DTD externa ya que de esa forma se pueden validar varios documentos a la vez. La ruta puede ser absoluta y entonces se indica su URL:

```
<!DOCTYPE raíz SYSTEM "http://www.empresa.com/docs.dtd">
```

Pero puede ser relativa:

```
<!DOCTYPE raíz SYSTEM "docs.dtd">
```

Entonces se busca al archivo DTD desde el directorio donde se encuentra el archivo XML que queremos validar (en el ejemplo, el archivo **docs.dtd** debe encontrarse en el mismo directorio que el archivo XML).

DTD externo de tipo PUBLIC

Se entiende que la palabra SYSTEM se utiliza cuando el documento DTD es privado. Si se trata de un documento de uso público, entonces se usa PUBLIC. La sintaxis, en este caso, es:

```
<!DOCTYPE raíz PUBLIC "nombreDTD" "DTD_URL">
```

La *raíz* sigue siendo el nombre del elemento raíz. El **nombreDTD** es el nombre público que se le da al DTD en cuestión.

[1.3]definiciones en un DTD

En un código DTD (tanto externo como interno) se pueden definir:

- Los **elementos** que se pueden utilizar en un documento XML. En esta definición se indica además que pueden contener dichos elementos.
- Los **atributos** que pueden poseer los elementos incluso indicando sus posibles valores válidos.
- Las **entidades** que puede utilizar el documento XML.

Definición de elementos mediante DTD

Los elementos que se pueden utilizar en un XML se definen en su DTD mediante una etiqueta **!ELEMENT**. La sintaxis de la misma es:

```
<!ELEMENT nombre tipo>
```

Donde:

- El *nombre* es el identificador que tendrá el elemento en el documento XML (hay que recordar que se distingue entre mayúsculas y minúsculas).
- El *tipo* indica el funcionamiento del elemento, relativo al contenido que puede tener.

[2.1]tipos de contenido en los elementos

EMPTY

Ejemplo de definición de elemento vacío:

```
<!ELEMENT línea EMPTY >
```

Cuando se indica como tipo la palabra EMPTY, se indica que el elemento no puede tener contenido. Eso, por cierto, no quiere decir que no podrán contener atributos. Los elementos vacíos pueden tener atributos si así se especifican en el DTD.

Ejemplos de elementos vacíos son:

```
<línea></línea>
```

ANY

Permite cualquier contenido en el elemento, sin restricciones de ningún tipo. Puede contener texto, otro tipo de datos y cualquier etiqueta. Además puede tener atributos.

Ejemplo:

```
<?xml version="1.0"?>
<!DOCTYPE persona [
        <!ELEMENT persona (nombre, apellidos)>
        <!ELEMENT nombre (#PCDATA)>
        <!ELEMENT apellidos ANY>
]>
<persona>
        <nombre>Jorge</nombre>
        <apellidos>Sánchez Asenjo
                <nombre>Jorge</nombre>
        </apellidos>
</persona>
```

Al definir *apellidos* como elemento ANY, permite incluso que dentro haya una etiqueta *nombre*.

Puesto que un DTD se usa para restringir la escritura de un tipo de documentos XML, el uso de ANY debe de ser muy cauteloso.

contenido concreto

En los elementos se puede indicar que, dentro del mismo, tienen que aparecer un contenido concreto. Dicho elemento se indica entre paréntesis. Ejemplo:

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

En el ejemplo, dentro de un elemento *persona* **obligatoriamente** debe existir una etiqueta *nombre*(**una y sólo una)**.

No sólo se pueden indicar nombres de elementos como contenido concreto, la indicación **#PCDATA** significa que el elemento podrá contener texto literal (tan largo como se desee).

El texto de tipo #PCDATA tiene que cumplir las reglas de texto XML. Eso implica que no puede contener símbolos como prohibidos como **<** o **>**. En su lugar hay que utilizar entidades.

secuencias

En el caso de indicar una lista de elementos separados por comas, por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
        <!ELEMENT persona (nombre, apellidos, edad)>
        <!ELEMENT nombre (#PCDATA)>
        <!ELEMENT apellidos (#PCDATA)>
        <!ELEMENT edad (#PCDATA)>
]>
<persona>
        <nombre>Antonio</nombre>
        <apellidos>Pérez</apellidos>
        <edad>35</edad>
</persona>
```

Indica que el elemento contendrá la lista de elementos indicada, la cual deberá estar en el mismo orden especificado en el DTD. Es decir, en el ejemplo los *apellidos* no podrían aparecer antes que el *nombre*.

elecciones u opciones

Los elementos pueden contener elementos opcionales (puede aparecer uno u otro). Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE artículo [
        <!ELEMENT artículo (id | serie)>
        <!ELEMENT id (#PCDATA)>
        <!ELEMENT serie (#PCDATA)>
]>
<artículo>
        <id>16</id>
</artículo>
```

La barra vertical indica que el elemento puede contener una u otra opción (pero sólo una). Es decir, también sería válido:

```
<!DOCTYPE artículo [
    <!ELEMENT artículo (id | serie)>
    <!ELEMENT id (#PCDATA)>
    <!ELEMENT serie (#PCDATA)>
]>
<
```

<artículo>

```
        <serie>X1238H</serie>
</artículo>
```

combinaciones

Por supuesto puede haber combinaciones de definiciones, si tenemos un documento DTD llamado *coordenada.dtd* con este contenido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT coordenada ((longitud, latitud) | coordUniversal)>
<!ELEMENT longitud (#PCDATA)>
<!ELEMENT latitud (#PCDATA)>
<!ELEMENT coordUniversal (#PCDATA)>
```

Sería válido este documento:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE coordenada SYSTEM "coordenada.dtd">
<coordenada>
        <longitud>234</longitud>
        <latitud>-23</latitud>
</coordenada>
```

Y sería válido también:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE coordenada SYSTEM "coordenada.dtd">
<coordenada>
        <coordUniversal>1232332</coordUniversal>
</coordenada>
```

cardinalidad

La cardinalidad es el número de veces que puede aparecer un determinado contenido en un elemento. Se realiza mediante estos símbolos:

- **?** Contenido opcional, puede aparecer (una sola vez) o no aparecer
- ***** Contenido opcional y repetible. Es decir puede no aparecer y puede incluso aparecer varias veces
- **+** Contenido obligatorio y repetible. Tiene que aparecer e incluso puede aparecer varias veces

Ejemplo:

```
<!ELEMENT película (título, dirección+,
       argumento?, actor*)>
```

Según la instrucción anterior el elemento película consta de un título, uno o más elementos de dirección, puede o no tener argumento, y de varios a ningún actor (además se tendría que respetar ese orden). Otro ejemplo (*polígono.dtd*):

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT polígono ((coordX,coordY)+ | nombre)>
<!ELEMENT coordX (#PCDATA)>
<!ELEMENT coordY (#PCDATA)>
<!ELEMENT nombre (#PCDATA)>
```

Con esa DTD sería válido el documento:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE polígono SYSTEM "polígono.dtd">
<polígono>
        <coordX>12</coordX>
        <coordY>13</coordY>
        <coordX>17</coordX>
        <coordY>23</coordY>
        <coordX>34</coordX>
        <coordY>56</coordY>
</polígono>
```

Pero también:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE polígono SYSTEM "polígono.dtd">
<polígono>
        <nombre>Triángulo</nombre>
</polígono>
```

Declaración de atributos mediante DTD

Los atributos permiten añadir información a un elemento. Un atributo no puede constar de más atributos y cada atributo sólo puede aparecer una vez en cada elemento.

[3.1]sintaxis de la declaración de atributos

Los atributos se declaran mediante la etiqueta **!ATTLIST**, sintaxis:

```
<!ATTLIST elemento nombreAtributo tipo presencia
    valorPorDefecto>
```

Donde:

- *elemento*. Es el nombre del elemento que podrá utilizar el atributo
- *nombreAtributo*. Es el identificador del atributo que estamos declarando (y que debe de cumplir las reglas de identificadores de XML)
- *tipo*. Es el tipo de valores que podemos asignar al atributo
- *presencia*. Indica las características de los valores que puede tomar el atributo: si es obligatorio, si hay valor por defecto,...
- *valorPorDefecto*. Permite especificar un valor que el atributo tomará si no especifica otro explícitamente en el documentoXML.
-

[3.2]CDATA

Para indicar el tipo de valores de un atributo, la palabra **CDATA** sirve para indicar que el atributo contiene texto (CDATA es el acrónimo de *Character DATA*).

A diferencia de **#PCDATA**, su contenido no es procesado, lo que significa que puede contener cualquier valor (incluidos símbolos prohibidos en los #PCDATA como **<**, **>**, **&**, etc)

[3.4.3]declarar atributos

Esta declaración:

```
<!ATTLIST persona nacionalidad CDATA>
```

Significa que hemos definido el atributo *nacionalidad* correspondiente al elemento *persona*. Que será de tipo *CDATA*, es decir texto normal. Así en un XML que se valide con el DTD anterior, podremos indicar:

<persona nacionalidad=”española”**>**

[3.4.4]valores

Al declarar un atributo, lo último que se indica es la propiedad relativa al valor por defecto del atributo. Se comentan a continuación como indicar valores.

valor por defecto concreto

Si al final de la declaración de un atributo aparece un valor concreto, se entiende que ese será el valor por defecto. Es decir, que se podría no utilizar el atributo en un elemento y entonces dicho atributo tomaría dicho valor.

Por ejemplo supongamos que éste es el archivo *directorio.dtd*:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!ELEMENT  directorio (persona)+>
<!ELEMENT  persona (#PCDATA)>
<!ATTLIST persona nacionalidad CDATA “Española”>
```

Se define en él el atributo *nacionalidad,* para el elemento *persona,* como un atributo que contendrá texto de todo tipo, pero que por defecto toma el valor *Española* (nacionalidad por defecto en dicho archivo).

Entonces este archivo XML será válido:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE directorio SYSTEM “directorio.dtd”>
<directorio>
    <persona nacionalidad=”Francesa”>
        Vivian Maret
```

```
    </persona>
    <persona>Juan Martín</persona>
</directorio>
```

Entonces para *Vivian Maret* se ha indicado explícitamente la *nacionalidad*, pero no se ha indicado ese atributo para *Juan Martín*, por lo que Juan Martían tendrá nacionalidad española.

valores fijos

Se puede utilizar el término **#FIXED** para definir un valor fijo para un atributo. Un valor fijo de un atributo no se puede modificar. Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT  directorio (persona)+>
<!ELEMENT  persona (#PCDATA)>
<!ATTLIST persona nacionalidad CDATA #FIXED "Española">
```

El atributo *nacionalidad* no podrá tomar ningún valor que no sea el valor *Española*, impidiendo tomar otra nacionalidad.

En la práctica este tipo de atributos no se usa demasiado, sólo se usa en el caso de atributos que indican alguna información basada en la propia existencia o no de dicho atributo. Es el caso del atributo **noshade** del elemento **hr** del lenguaje HTML. **hr** sirve para dibujar una línea en una página web, la línea se dibuja en relieve salvo que aparezca el atributo *noshade*, el cual indica la línea tenga formato *aplanado*. En HTML compatible con XML (lo que se conoce como XHTML) a *noshade* sólo se le puede dar un valor: la propia palabra *noshade*:

```
<hr noshade="noshade" />
```

valores requeridos

En este caso se usa la palabra **#REQUIRED** indicando con ello que siempre hay que dar valor al atributo. Ejemplo:

```
<!ATTLIST persona nacionalidad CDATA #REQUIRED>
```

Un documento XML que utilice el elemento *persona* deberá especificar obligatoriamente la *nacionalidad*.

Obviamente, el uso de #REQUIRED no permite indicar un valor por defecto, al no poder dejarse sin especificar el atributo.

valor opcional

La palabra **#IMPLIED** especificada en el atributo indicaría que dicho atributo puede quedarse sin valor; es decir no posee valor por defecto, pero puede quedarse sin especificar (quedaría nulo por tanto).

<!ATTLIST persona nacionalidad **CDATA #IMPLIED>**

En el ejemplo, el atributo nacionalidad no es obligatorio especificarle, puede quedar sin valor. Nuevamente en este caso no se puede especificar un valor por defecto (sería absurdo).

[3.4.5]tipos de atributo

CDATA

Como se comentó antes, los atributos de tipo CDATA permiten indicar como valor cualquier texto. A diferencia de los datos **PCDATA** de los elementos, los CDATA admiten cualquier carácter del tipo que sea.

ID

Sirve para especificar que el atributo contendrá un identificador de elemento. Un identificador es un valor único que tendrá cada elemento y son muy usados en XML.

El valor de un atributo de tipo ID cumple estas reglas:

- El valor tiene que cumplir las mismas reglas que para especificar nombres XML. Es decir: nada de espacios, no pueden comenzar con un número y sólo admite letras, números y el carácter de subrayado (_).
- No puede haber dos elementos con el mismo ID en un mismo documento XML

- En el DTD, para cada elemento, sólo puede indicarse un atributo como ID. No puede haber dos atributos distintos en el mismo elemento de tipo ID.
- Los atributos ID solo pueden indicar **#IMPLIED** o **#REQUIRED** en el apartado del valor por defecto.

Los IDs son especialmente útiles para las herramientas de maquetación, análisis y programación de aplicaciones XML. Ya que permiten diferenciar de manera única a cada elemento.

IDREF

Los atributos IDREF contienen como valor el identificador de otro elemento. Es decir será una referencia a otro elemento. Las reglas de los IDREF son:

- El valor de un IDREF debe cumplir las reglas para especificar nombres XML (es lógico ya que contienen valores de tipo ID)
- Debe existir un atributo ID en el documento XML cuyo valor coincida con el especificado en un IDREF (de otro modo se haría referencia a un elemento inexistente y esto no está permitido)

La idea es poder relacionar elementos a través de atributos de tipo ID e IDREF. Ejemplo de uso:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Archivo directorio.dtd -->
<!ELEMENT directorio (persona)+ >
<!ELEMENT persona (#PCDATA) >
<!ATTLIST persona id ID #REQUIRED madre IDREF #IMPLIED
                                padre IDREF #IMPLIED>
<?xml version="1.0" encoding="UTF-8"?>
<!-- Archivo directorio1.xml-->
<!DOCTYPE directorio SYSTEM "directorio.dtd">
<directorio>
    <persona id="p1">Pedro</persona>
    <persona id="p2">Marisa</persona>
    <persona id="p3" madre="p2" padre="p1">Carmen</persona>
</directorio>
```

Carmen es la hija de Pedro y Marisa, según el código anterior, ya que los atributos *padre* y *madre* de tipo **IDREF** contienen los **ID** de *Pedro* y *Marisa*.

IDREFS

Igual que el anterior, solo que, en este caso, se permite indicar varias referencias (que deben existir en el documento XML) a otros ID, **separadas por espacios**. Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Archivo directorio.dtd -->
<!ELEMENT directorio (persona)+ >
<!ELEMENT persona (#PCDATA) >
<!ATTLIST persona id ID #REQUIRED
                                padres IDREFS #IMPLIED >
<?xml version="1.0" encoding="UTF-8"?>
<!-- Archivo directorio1.xml-->
<!DOCTYPE directorio SYSTEM "directorio.dtd">
<directorio>
    <persona id="p1">Pedro</persona>
    <persona id="p2">Marisa</persona>
    <persona id="p3" padres="p1 p2">Carmen</persona>
</directorio>
```

NMTOKEN

El valor del atributo será un texto que cumple reglas estrictas. Concretamente, cumplirá las reglas de los nombres de elementos XML. Se usa en atributos donde se entiende que CDATA permite demasiadas libertades.

Los atributos NMTOKEN contienen texto donde solo existirán letras, números y el símbolo _, es decir un texto que cumple las reglas para nombres XML.

NMTOKENS

El atributo puede contener varios valores de tipo **NMTOKEN** separados por espacios.

ENTITY

El valor de un atributo será una entidad de la cual se indica el nombre. Más adelante se explica el uso de las entidades.

ENTITIES

El valor del atributo será una lista de nombres de entidades separadas por espacios.

enumeraciones

Se trata de atributos a los que se indica una serie de posibles valores. El valor del atributo debe coincidir con alguno de la lista. Los valores posibles se indican entre paréntesis separados por el símbolo |.

Ejemplo:

```
<!ATTLIST persona sexo (Hombre | Mujer) #REQUIRED >
```

Las personas sólo podrán especificar como sexo *"Hombre"* o *"Mujer"* y nada más:

```
<persona sexo="Varón">Javier Ruiz</persona>
```

El código anterior XML fallaría ya que el atributo sexo no admite el valor *"Varón"*.

[3.4.6]declaración de varios atributos en la misma etiqueta

Se usa muy habitualmente para indicar de forma cómoda todos los atributos de un determinado elemento:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT  directorio (persona)+>
<!ELEMENT  persona (#PCDATA)>
<!ATTLIST persona   nacionalidad CDATA "Española"
                    sexo (Hombre | Mujer) #IMPLIED
                    id ID #REQUIRED>
```

Para ese documento DTD, sería válido este XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE directorio SYSTEM "directorio.dtd">
<directorio>
    <persona nacionalidad="Francesa" id="A1234">
          Vivian Maret
    </persona>
    <persona  id="A789">Juan Martín</persona>
    <persona sexo="Hombre" id="A12">Rafael Díaz</persona>
</directorio>
```

[3.5] definición de entidades en DTD

Las entidades son elementos XML que permiten indicar abreviaturas de texto (o referencias a elementos externos abreviadas) o utilizar caracteres que de otra forma serían inválidos en el documento.

A continuación se explican sus posibilidades

[3.5.1]entidades ya existentes

En XML están definidas las siguientes entidades:

entidad	significado
<	El símbolo de menor (<)
>	El símbolo de mayor (>)
&	El ampersand: &
'	La comilla simple (‘)
"	La comilla doble (“)

Estas entidades no hay que declararlas en ningún DTD, todos los analizadores de XML estándar conocen estas entidades.

Ejemplo:

```
<autor>Leopoldo Alas &apos;Clarín&apos;</autor>
```

El texto **PCDATA** del autor es *Leopoldo Alas ‘Clarín’* (así se visualizará en el navegador).

[3.5.2]entidades para referencias a caracteres especiales

La etiqueta inicial **<?xml** permite indicar (entre otras cosas) el juego de caracteres que utiliza un documento XML (normalmente **Unicode**, **UTF8**).

Si deseamos indicar un carácter especial que no está contenido en nuestro teclado, conociendo su código en el juego de

caracteres que utiliza el documento, podemos especificarle con la sintaxis:

```
&#número;
```

Donde el número es el código del carácter en decimal. En hexadecimal se puede hacer con:

```
&#xnúmero;
```

Ejemplo:

```
<calle>Kantstra&#223;e, Berlín</calle>
```

En el navegador este elemento aparecería como:

```
<calle>Kantstraße, Berlín</calle>
```

El número se puede poner en hexadecimal si se antecede una *x* al nombre, por ejemplo sería equivalente:

```
<calle>Kantstra&#EF;e, Berlín</calle>
```

[3.5.3]entidades generales

Se usan como abreviaturas que aparecerán en el documento XML. La razón de su uso es facilitar la escritura de nombres repetitivos (nombres de la empresa, direcciones muy utilizadas,…). La sintaxis para declarar una entidad de este tipo es:

```
<!ENTITY nombre "texto">
```

Para usar en un documento XML la entidad declarada, se usa el formato habitual:

```
&nombre;
```

Ejemplo de declaración de una entidad:

```
<!ENTITY mayor "Calle Mayor Principal" >
```

Su uso en un documento XML, sería:

```
<dirección>&mayor; 18</dirección>
```

La dirección indicada es *Calle Mayor Principal 18*.

Incluso se pueden indicar símbolos que no son **PCDATA** al definir entidades:

```
<!ENTITY negCursiva "<strong><em></em></strong>">
```

El documento XML que utilice dicha entidad incluirá todos los símbolos (y por lo tanto estará especificando etiquetas en el código).

Un uso muy interesante es usar entidades que hacen referencia a archivos externos (mediante su dirección URL), por ejemplo:

```
<!ENTITY direcciónCompleta SYSTEM "direccion.txt" >
```

Es la palabra **SYSTEM** la que indica que la entidad no es un texto sino que es el contenido de un archivo. El uso de la entidad *&direcciónCompleta;* en un documento XML provocará que en dicho documento se añada el contenido del archivo *dirección.txt* (en la posición exacta en la que esté colocada la referencia a la entidad).

[3.5.4]entidades de parámetros

Solo se pueden utilizar dentro del DTD (no en el documento XML). Su uso más habitual es construir DTD utilizando las entidades definidas a fin de ahorrar trabajo al crear el propio DTD.

Su uso es similar a las entidades generales sólo que utilizan el símbolo **%** en lugar del símbolo **&**. Al igual que las generales, deben de ser declaradas antes de poder usarse:

```
<!ENTITY % mayor "Calle Mayor Principal" >
```

Y su uso (dentro del DTD), por ejemplo:

```
<!ATTLIST persona dirección CDATA "%mayor;">
```

En este caso las comillas dobles son obligatorias porque los valores por defecto van entrecomillados (como se ha visto anteriormente).

Las entidades de parámetros pueden utilizar archivos externos, ejemplo de DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % directorio SYSTEM "directorio.dtd" >
%directorio;
<!ELEMENT empresa (razónSocial, dirección)  >
<!ELEMENT razónSocial (#PCDATA) >
```

De esta forma se construye un DTD con el contenido ya especificado en otro DTD. En el ejemplo las *empresas* constan de elementos *razónSocial* y de *directorio*. El elemento directorio no se define, sino que su descripción está especificada en *directorio.dtd*.

No obstante el uso más habitual es definir una entidad para utilizar código común en el propio DTD, por ejemplo supongamos que dos elementos, profesor y alumno comparten atributos comunes. Entonces este código simplifica la definición de los atributos de esos elementos.