

Unidad 6 XML DTD (Document Type Definition)

Módulo: Lenguaje de Marcas y Sistemas de Gestión de Información
Ciclo: Desarrollo de Aplicaciones Web

Profesora: M^a Ángeles Calabuig Navarro

1.- DTD (Document Type Definition)

- Cuando un XML **no contiene un DTD**, cualquier etiqueta que aparezca en el mismo se considerara válida
- El analizador solo podrá comprobar que el **documento esta bien formado**
- La **existencia del DTD** permite asegurar que los documentos siguen las reglas del XML
- Es imprescindible la especificación de un DTD que defina formalmente el lenguaje de etiquetado requerido
- 1º paso antes de escribir cualquier documento XML
- Comparación con BDs: 1º se define la estructura y luego ya se puede trabajar con los datos correspondientes

1.- DTD (Document Type Definition)

- **Documento XML válido** es un documento “bien formado” que se ajusta a las reglas de un DTD
- Un **XML** con la sintaxis correcta es un documento “**bien formado**”
- Un XML validado contra un DTD es un **XML “válido”**
- Mediante el uso de los DTD podremos validar XML
- La validación de documentos consiste en comprobar que, además de ser bien formados, se corresponden con la estructura prevista para el contenido que aportan

1.- DTD (Document Type Definition)

○ Ejemplo:

- Venta de coches donde cada tienda o concesionario tiene que enviar su información a un sitio web
- En esa web se publicará las ofertas, por lo que entendemos que no puede ser viable que cada una de las empresas implemente su propia versión de XML
- Por el contrario, lo adecuado es que el sitio web defina el formato exacto que deben seguir los documentos se van a recibir
- Así se asegura que serán documentos XML **válidos** y todos con la misma estructura

2.- Declaración del DTD

- Definición de tipo de documento (DTD) es una descripción de la estructura y sintaxis de un documento XML
- DTD define las reglas que debe cumplir la información contenida en un documento XML, para que el documento sea válido
- Cuando creamos un DTD lo que estamos haciendo es crear nuestro propio lenguaje de marcas para nuestra aplicación concreta, de forma que el documento XML que se ajuste a esa DTD se pueda considerar **válido**

2.- Declaración del DTD

- En un DTD se describen los elementos :
 - nombres de los elementos,
 - atributos que pueden tener,
 - tipos de datos que pueden contener
 - Etc
 - que podrá contener el documento, su estructura y posibilidades de anidamiento
- DTD puede ser un fichero externo, con extension .dtd
- También puede estar contenida en el propio documento XML, incluido en la declaración de tipo de documento que forma parte del prólogo del documento

2.- Declaración del DTD

- NO hay que confundir ambas cosas, ya que:
- La **Definición (DTD)** **contendrá las especificaciones necesarias sobre los elementos**, pudiendo estar incluido o no en el propio documento
- La **Declaración** **siempre estará en el prologo del documento (incluyendo la definición** del etiquetado o simplemente haciendo referencia a su ubicación exterior)
- Esta declaración es opcional en el documento XML, pero es necesaria para poder validar los datos que contiene
- Situada en el prólogo a continuación de la declaración XML, en la 2ª línea, mediante **DOCTYPE** y **deberá contener siempre la especificación de elemento** raíz del documento

2.- Declaración del DTD

- **Opción A**

- **A) El formato de declaración, cuando incluye la definición en el propio documento, podría ser:**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE elemento_raiz [
```

```
    Declaraciones internas
```

```
]>
```

```
<elemento_raiz>
```

```
.....
```

```
</elemento_raiz>
```

- Estará incluida dentro de la declaración DOCTYPE, después del elemento_raiz y comprendida entre “[” y “]”

2.- Declaración del DTD

o Opción A: Ejemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE nota
[
<!ELEMENT nota (destinatario,remitente,cabecera,cuerpo)>
<!ELEMENT destinatario (#PCDATA)>
<!ELEMENT remitente (#PCDATA)>
<!ELEMENT cabecera (#PCDATA)>
<!ELEMENT cuerpo (#PCDATA)>
]>
<nota>
<destinatario>Tove</destinatario>
<remitente>Jani</remitente>
<cabecera>Recordatorio</cabecera>
<cuerpo>Llamame!</cuerpo>
</nota>
```

2.- Declaración del DTD

- El DTD anterior tiene el siguiente significado:
 - **!DOCTYPE nota**, indica que el elemento raíz de este documento es nota
 - **!ELEMENT nota**, indica que el elemento nota contiene 4 elementos:
 - destinatario, remitente, cabecera y cuerpo
 - **!ELEMENT destinatario**, indica que el elemento destinatario es de tipo #PCDATA, es decir, texto
 - **!ELEMENT remitente**, indica que el elemento remitente es de tipo #PCDATA
 - **!ELEMENT cabecera**, indica que el elemento cabecera es de tipo #PCDATA
 - **!ELEMENT cuerpo**, indica que el elemento cuerpo es de tipo #PCDATA

2.- Declaración del DTD

- **Opción B**
- Normalmente un DTD se usa para validar un gran nº de documentos XML
- La mayoría de veces tiene poco sentido que el DTD se incluya dentro del documento XML ya que se tendría que repetir en todos los documentos XML pertenecientes a un mismo lenguaje (DTD)
- Teniendo esto en cuenta se puede distinguir entre **2 tipos de referencias externas**:
 - Un documento DTD aun no publicado
 - Se especifica con la palabra SYSTEM seguida de la URL con la ubicación del documento:
 - `<!DOCTYPE elemento_raiz SYSTEM "archivo_declaraciones.dtd">`
 - Un DTD que ha sido publicado

2.- Declaración del DTD

- **Opción B**

- Las **2 tipos de referencias externas**:

- Un documento DTD aun no publicado
- Un DTD que ha sido publicado
 - Se usa la palabra PUBLIC seguida por el identificador publico asociado a este DTD
 - Sigue siendo necesario incluir la URL al fichero DTD que solo utilizará en caso de fallar la localización del fichero
 - Usando el identificador publico:
 - `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`
 - Que es la DTD publica utilizada para validar documentos XHTML estrictos
 - Los documentos XHTML no son mas que documentos HTML que utilizan DTD's para asegurar que el documento HTML es valido

3.- Declaración de tipos de elementos

- Los elementos son la base de las marcas XML
- Indican que etiquetas serán permitidas en el documento
- El DTD tiene que declarar cada uno de los elementos
- Las declaraciones de tipo de elemento deben empezar con "**<!ELEMENT**" seguidas por el identificador genérico del elemento que se declara

<!ELEMENT *nombre tipo_contenido* >

- El nombre del elemento debe ser un nombre XML válido y solo puede haber una declaración por elemento
- No podrá repetirse

3.- Declaración de tipos de elementos

● Ejemplo:

- Al cual le podría corresponder un DTD como el siguiente, que estaría contenido en el fichero identificacion.dtd

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE identificacion SYSTEM "identificacion.dtd">

<identificacion>
  <nombre_completo>
    <nombre>
      Pepe
    </nombre>

    <apellido1>
      Gonzalez
    </apellido1>

    <apellido2>
      Ribera
    </apellido2>
  </nombre_completo>

  <apodo>
    Pepito Grillo
  </apodo>
</identificacion>
```

```
<!ELEMENT identificacion (nombre_completo, apodo)>
<!ELEMENT nombre_completo (nombre, apellido1,apellido2)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellido1 (#PCDATA)>
<!ELEMENT apellido2 (#PCDATA)>
<!ELEMENT apodo (#PCDATA)>
```

3.- Declaración de tipos de elementos

- En el dtd lo primero que hacemos es definir el “elemento raíz” que llamamos “identificación”
- El elemento raíz está formado, a su vez, por 2 elementos:
 - “nombre_completo” y
 - “apodo”
- Por otro lado, “nombre_completo” está formado, a su vez por otros 3 elementos: “nombre”, “apellido1” y “apellido2”
- Los elementos que NO contienen a otros elementos, como “nombre”, “apellido1”, “apellido2” y “apodo” definen que el tipo de datos de esos elementos es #PCDATA (texto plano)
- También comprueba que el fichero xml es válido, es decir, sigue la estructura definida por el dtd

3.- Declaración de tipos de elementos

- Si ampliamos el DTD con más elementos:

```
<!ELEMENT identificacion (situacion?,nombre_completo, apodo*, mail?)>
  <!ELEMENT situacion (estudiante|trabajador)>
    <!ELEMENT estudiante (#PCDATA)>
    <!ELEMENT trabajador (#PCDATA)>
  <!ELEMENT nombre_completo (nombre+, apellido1,apellido2)>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT apellido1 (#PCDATA)>
    <!ELEMENT apellido2 (#PCDATA)>
  <!ELEMENT apodo (#PCDATA)>
  <!ELEMENT mail (#PCDATA)>
```

- El nuevo DTD podría validar documentos con contenidos similares al siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE identificacion SYSTEM "identificacion_ampliado.dtd">
<identificacion>
  <situacion>
    <estudiante>
      Universitat Jaume I Castelló
    </estudiante>
    <!-- esto es un comentario:
          estudiante y trabajador son alternativos,
          si lleva información de uno no tendrá del otro -->
  </situacion>

  <nombre_completo>
    <nombre>
      Pepe
    </nombre>

    <apellido1>
      Gonzalez
    </apellido1>

    <apellido2>
      Ribera
    </apellido2>
  </nombre_completo>

  <apodo>
    Pepito Grillo
  </apodo>

  <mail>
    elgrillito@correobasura.com
  </mail>
</identificacion>
```


3.- Declaración de tipos de elementos

- Según el contenido de los elementos tenemos diferentes tipos:

1. Elementos que sólo contienen elementos

- Tendremos que especificar entre paréntesis el identificador de cada uno de los elementos que anidará
- Hay 2 tipos de relación entre los elementos hijos:

- Secuenciales**, referenciándolos por su nombre, separados por comas:

<!ELEMENT nombre_elemento (elemento1,elemento2,elemento3....)>

- En nuestro ejemplo, el elemento raíz (que siempre incorporaremos en primer lugar), hemos visto que lo especificamos como:

<!ELEMENT identificacion (situacion?,nombre_completo, apodo*,mail?)>

- Alternativos**: cuando el elemento contiene uno y solo uno de los elementos hijos especificados, en cuyo caso los separaremos mediante “ | ”:

<!ELEMENT nombre_elemento (elemento1 | elemento2 | elemento3....)>

- En nuestro ejemplo tenemos:

<!ELEMENT situacion (estudiante | trabajador)>

- El elemento situación podrá tomar uno de los 2 valores especificados: estudiante o trabajador, pero solo uno de ellos

3.- Declaración de tipos de elementos

- Según el contenido de los elementos tenemos diferentes tipos:

1. Elementos que sólo contienen elementos

- Además de especificar qué elementos hijos puede contener el elemento y en qué orden
- Se puede establecer cuantas veces aparece cada uno de ellos, con un carácter que indique el factor de repetición, o un **indicador de frecuencia**:
 - El carácter '*': el elemento o grupo de elementos puede repetirse 0 o mas veces
 - El carácter '?': el elemento o grupo de elementos puede aparecer 0 o 1 veces.
 - El carácter '+': el elemento o grupo de elementos puede repetirse 1 o mas veces
 - Por defecto, si no ponemos nada, el elemento debe aparecer una vez

3.- Declaración de tipos de elementos

- Según el contenido de los elementos tenemos diferentes tipos:

1. Elementos que sólo contienen elementos

- Seguimos nuestro ejemplo:

<!ELEMENT identificacion (situacion?, nombre_completo, apodo*, mail?)>

- Si consideramos que nuestro elemento raíz (identificacion) está formado por los elementos “situacion” y “mail” que son opcionales, pueden aparecer una vez o no aparecer
- “nombre_completo” siempre debe aparecer, aunque solo una vez ya que aparece en la lista sin ningún indicador de frecuencia
- mientras que el apodo puede no estar presente o aparecer repetidamente, ya que una persona puede no tener ningún apodo o varios

<!ELEMENT nombre_completo (nombre+, apellido1, apellido2)>

- Así mismo, anidado en nombre_completo, tenemos el nombre que podemos considerar que es necesario una vez, es decir, todas las personas tienen como mínimo un nombre, aunque pueden tener mas,
- mientras tanto el apellido1 como el apellido2 serán únicos

3.- Declaración de tipos de elementos

- Según el contenido de los elementos tenemos diferentes tipos:

2. Elementos que sólo contienen datos

- En la declaración se especifican mediante **(#PCDATA)** e indica que pueden contener datos de tipo carácter (**Parser Character Data**)
- Debemos tener cuidado de que entre el identificador del elemento y el símbolo inicial del paréntesis haya un espacio de separación
- En nuestro ejemplo los siguientes elementos son de tipo texto:

<!ELEMENT nombre (#PCDATA)>

<!ELEMENT apellido1 (#PCDATA)>

<!ELEMENT apellido2 (#PCDATA)>

<!ELEMENT apodo (#PCDATA)>

<!ELEMENT mail (#PCDATA)>

3.- Declaración de tipos de elementos

- Según el contenido de los elementos tenemos diferentes tipos:

3. Elementos vacíos

- Aunque no es usual, los elementos pueden no tener ningún contenido pero pueden utilizarse para insertar los atributos
- Se declaran especificando la palabra **EMPTY**

<!ELEMENT nombre_elemento EMPTY>

- Un ejemplo es la declaración `
` de XHTML

<!ELEMENT br EMPTY>

4. Elementos mixtos

- No suelen utilizarse en XML ya que se puede especificar que elementos hijos podran aparecer, pero no dan indicación de frecuencia o si forman parte de una secuencia alternativa
- Su formato es muy rígido, siempre en primer lugar PCDATA, con una lista alternativa como un grupo

3.- Declaración de tipos de elementos

- Según el contenido de los elementos tenemos diferentes tipos:

4. Elementos mixtos

- No suelen utilizarse en XML ya que se puede especificar que elementos hijos podrán aparecer, pero no dan indicación de frecuencia o si forman parte de una secuencia alternativa
- Su formato es muy rígido, siempre en primer lugar PCDATA, con una lista alternativa como un grupo
- No se puede aplicar caracteres de repetición a los elementos hijos, solo la posibilidad del carácter de repetición * para el conjunto**
- Debe especificarse obligatoriamente el carácter de repetición '*' a todo el grupo**
- Se desaconseja su uso
- La declaración sería:

<!ELEMENT nombre elemento (#PCDATA | elem1 | elem2 | elem3)*>

4.- Declaración de tipos de atributos

- Los atributos permiten añadir información adicional a los elementos de un documento
- Las diferencias entre los elementos y los atributos son:
 - Los atributos no pueden contener sub-atributos y que los usamos para añadir información corta, sencilla y desestructurada
 - Cada uno de los atributos sólo se puede especificar una vez, y en cualquier orden**
 - Ejemplo:

`<mensaje prioridad="urgente">`

`<de>Alfredo Reino</de>`

`<a>Hans van Parijs`

`<texto idioma="holandés">`

Hallo Hans, hoe gaat het?

...

`</texto>`

`</mensaje>`

4.- Declaración de tipos de atributos

- Igual que ocurre con los elementos, cada uno de los distintos atributos identificados en la fase de diseño, debe declararse previamente en el DTD
- Pueden agruparse en una lista correspondiente para cada elemento
- Para cada atributo podremos tener también la especificación de su tipo y su valor por defecto o predeterminación
- Puede haber múltiples definiciones de listas de atributos para un mismo elemento
- Si se declara varias veces el mismo atributo solo prevalece el primero
- Su sintaxis es:

<!ATTLIST *nombre_elemento* *nombre_atributo* *tipo* *predeterminacion*>



Siempre se tiene que incluir el nombre del elemento al que pertenece el atributo



Identificador Atributo



CDATA
NMTOKEN
(VALOR1|VALOR2|....)
ID
.... etc



- #REQUIRED. El atributo es obligatorio.
- #IMPLIED. El atributo es opcional.
- #FIXED. Tiene un valor fijo declarado en el DTD.
- Valor por defecto. Si no se especifica.

4.- Declaración de tipos de atributos

- En el ejemplo anterior, para declarar la lista de atributos del elemento <mensaje> podríamos definir y utilizar la siguiente definición de atributos:

<!ELEMENT mensaje (de, a, texto)>

<!ATTLIST mensaje prioridad (normal | urgente)>

- Así pues, definimos para el elemento “mensaje” un atributo que llamamos “prioridad” y que puede tomar los valores “normal” o “urgente”
- Veamos a continuación cada parte del atributo con mas detalle.
- El tipo del atributo podrá tomar los siguientes valores:
 - CDATA
 - NMTOKEN
 - NMTOKENS
 - Tipos de atributos enumerados
 - Tipos de atributos ID

4.- Declaración de tipos de atributos

● Tipo de atributo **CDATA**, **NMTOKEN** Y **NMTOKENS**

- Si el valor del atributo está formado por un texto que puede incluir cualquier carácter imprimible, a excepción de los caracteres especiales, incluidos los espacios en blanco, entonces el tipo será **CDATA**
- Si pretendemos limitar el tipo de caracteres que pueden aparecer como valor en el atributo, debemos utilizar el tipo **NMTOKEN**
 - Solo permite que aparezcan los mismos caracteres que utilizamos para definir elementos y atributos
- Existe también la posibilidad de utilizar el tipo **NMTOKENS**. Esto indica que el atributo contendrá una lista de cadenas de tipo **NMTOKEN**

4.- Declaración de tipos de atributos

- Tipo de atributo CDATA, NMTOKEN Y NMTOKENS

- Ejemplos

- Los puntos indica que la definición del atributo esta incompleta

- **<!ATTLIST coche color CDATA ... >**

- Significa que el atributo color puede tomar cualquier valor, por ejemplo: blanco-rojo, rojo, beige claro, azul_celeste ...

- **<!ATTLIST coche color NMTOKEN...>**

- Significa que la propiedad color puede tomar solo valores que contengan letras, dígitos, puntos, guiones y subrayados
 - Deben comenzar por letra y no pueden contener espacios en blanco

- **<ATTLIST coche color NMTOKENS...>**

- La propiedad color será una lista de NMTOKENS
 - Por ejemplo: <coche color="blanco negro gris">

4.- Declaración de tipos de atributos

○ Tipo de atributos enumerados

- Se usan cuando el valor del atributo está restringido a un conjunto de valores
- En la declaración se usa el carácter ' | ' para separar los valores
- **<!ATTLIST coche color (blanco | negro | gris)>**
 - De esta forma, la propiedad color solo puede tomar los valores “blanco”, “negro” o “gris”, y solo uno de ellos
 - Cualquier otro valor hará que la validación del documento XML falle

○ Tipo de atributos ID

- Es frecuente que algunos elementos tengan algún valor que los identifica de forma unívoca
- Cuando un elemento contiene una propiedad de este tipo hay que asegurarse que esta no se repite en otro elemento
- Incluso con elementos diferentes
- Podemos poner:
 - **<!ATTLIST coche matricula ID ...>**

4.- Declaración de tipos de atributos

o Predeterminación de atributos

- o A continuación del nombre y el tipo del atributo debemos especificar si se requiere o no la presencia de un atributo, y el modo de gestionarlo en ese caso
- o Existen 4 posibles alternativas:
 - o #REQUIRED. El atributo es obligatorio.
 - o #IMPLIED. El atributo es opcional.
 - o #FIXED. Tiene un valor fijo declarado en el DTD.
 - o Valor por defecto si no se especifica.
- o Si no se define ninguna de estas alternativas el atributo será por defecto opcional
- o Siguiendo y completando nuestros ejemplos de los coches, podríamos tener:
 - o <!ATTLIST coche color CDATA #IMPLIED>
 - o <!ATTLIST coche matricula ID #REQUIRED>
 - o <!ATTLIST coche color CDATA "rojo">
 - o <!ATTLIST coche marca CDATA FIXED "Seat">

5.- Declaración de entidades

- Una entidad es una referencia a un objeto (texto, ficheros, paginas web, etc.) que serán sustituidas por el contenido al que se refieren
- Permite guardar contenido que puede ser utilizado muchas veces y poder descomponer un documento grande en subconjuntos mas manejables
- En ocasiones se emplean para descomponer un documento grande en otros mas pequeños, y en otros casos se usan para representar caracteres que no pueden incluirse como texto, como el caso de caracteres especiales
- Su sintaxis general seria:

<!ENTITY identificador "valor">

- Puede ser una **entidad interna**
 - Es la mas sencilla.
 - Consiste en abreviaturas definidas en el DTD.
 - Ejemplo:

<!ENTITY tema "Introducción a XML">

5.- Declaración de entidades

- Una vez definida en el DTD, en el documento XML correspondiente podemos utilizarla insertando *&tema*;
- Es decir, con el identificador precedido de & y acabado en “;”
- El parser cambiará la entidad por el valor asignado
- Existen también las **entidades externas**.
 - En ellas no tenemos el contenido dentro del DTD sino en cualquier otro sitio del sistema
 - Se hace referencia a su contenido mediante una URL precedida de la palabra SYSTEM o PUBLIC según proceda, y de esa forma podemos incluir parte de archivos para poder descomponerlos en pequeñas partes
- La sintaxis es

<!ENTITY nombre SYSTEM “URL”>

- Por ejemplo:

<!ENTITY tema SYSTEM http://www.misapuntes.com/tema3.xml>

- o

<!ENTITY intro SYSTEM http://ww.miservidor.com/intro.xml>

5.- Declaración de entidades

- XML hace referencia a objetos (ficheros, páginas web, imágenes, cualquier cosa) que no deben ser analizados sintácticamente según las reglas de XML, mediante el uso de entidades
- Se declaran en la DTD mediante el uso de "<!ENTITY"
- Una entidad puede no ser más que una abreviatura que se utiliza como una forma corta de algunos textos
- Al usar una referencia a esta entidad, el analizador sintáctico reemplaza la referencia con su contenido
- En otras ocasiones es una referencia a un objeto externo o local
- Las entidades pueden ser:
 - Internas o Externas
 - Analizadas o No analizadas
 - Generales o Parámetro

5.- Declaración de entidades

● Entidades generales internas

- Son básicamente abreviaturas definidas en la sección de la DTD del documento XML
- Son siempre entidades analizadas, es decir, una vez reemplazada la referencia a la entidad por su contenido, pasa a ser parte del documento XML y como tal, es analizada por el procesador XML.

● Ejemplo:

```
<!DOCTYPE texto[  
  <!ENTITY ovni "Objeto Volante No identificado">  
>  
<texto><titulo>Un día en la vida de un &ovni;  
</titulo></texto>
```

● Entidades generales externas analizadas

- Las entidades externas obtienen su contenido en cualquier otro sitio del sistema, ya sea otro archivo del disco duro, una página web o un objeto de una base de datos

5.- Declaración de entidades

Entidades generales externas analizadas

- Se hace referencia al contenido de una entidad así mediante la palabra SYSTEM seguida de un URI (*Universal Resource Identifier*)
- Ejemplo:

```
<!ENTITY intro SYSTEM "http://server.com/intro.xml">
```

Entidades no analizadas

- Referencian cualquier archivo que no sea XML
- Se declaran utilizando el calificador SYSTEM o PUBLIC, y van acompañadas de una notación

```
<!ENTITY logonscreen SYSTEM "c:\fm1.jpg" NDATA JPG>
```

- La notación se escribe al comienzo de la DTD
- <!NOTATION JPG SYSTEM "/programas/viewer.exe">
- o
- <!NOTATION gif SYSTEM "image/jpeg"

5.- Declaración de entidades

○ Notaciones

- Las notaciones pueden cumplir distintos propósitos:
 - Indicar el path del programa encargado de procesar la entidad (por ejemplo un visor especial)
 - Apuntar a un lugar en el que existe documentación sobre el formato, etc.
- La norma es abierta en este aspecto

○ Entidades parámetro internas

- Se denominan entidades parámetro a aquellas que sólo pueden usarse en la DTD, y no en el documento XML
- Para hacer referencia a ellas, se usa el símbolo "%" en lugar de "&" tanto como para declararlas como para usarlas
- Ejemplo

```
<!DOCTYPE texto[
```

```
<!ENTITY % elemento-alf "<!ELEMENT ALF (#PCDATA)>"
```

```
%elemento-alf; ]>
```

5.- Declaración de entidades

- **Entidades parámetro externas**

- Igualmente, las entidades parámetro, pueden ser externas

- Ejemplo:

```
<!DOCTYPE texto[
```

```
<!ENTITY % elemento-alf SYSTEM "alf.ent">
```

```
...
```

```
%elemento-alf;
```