Tutorial DTD

(rev.3)

¿Qué es un DTD?

Un DTD es una definición de tipo de documento.

Un DTD define la estructura y los elementos y atributos legales de un documento XML.

¿Por qué utilizar un DTD?

Con un DTD, grupos independientes de personas pueden acordar un DTD estándar para intercambiar datos.

Una aplicación puede usar una DTD para verificar que los datos XML sean válidos.

Una declaración interna de DTD

Si la DTD se declara dentro del archivo XML, debe incluirse dentro de la definición de <!DOCTYPE>:

Documento XML con DTD interna

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

<!DOCTYPE note [

<!ELEMENT note (to,from,heading,body)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT from (#PCDATA)>

<!ELEMENT heading (#PCDATA)>

<!ELEMENT body (#PCDATA)>

]>

<note>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend</body>

</note>
```

El DTD anterior se interpreta así:

- !DOCTYPE note define que el elemento raíz de este documento es note
- !ELEMENT note define que el elemento de nota debe contener cuatro elementos: "a, desde, encabezado, cuerpo"
- !ELEMENT to define que el elemento to sea del tipo "#PCDATA"
- !ELEMENT from define el elemento from como de tipo "#PCDATA"
- !ELEMENT heading define que el elemento del encabezado sea del tipo "#PCDATA"
- !ELEMENT body define el elemento body como de tipo "#PCDATA"

Una declaración de DTD externa

Si la DTD se declara en un archivo externo, la definición de <!DOCTYPE> debe contener una referencia al archivo DTD:

Documento XML con referencia a un DTD externo

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

<!DOCTYPE note SYSTEM "note.dtd">

<note>

  <to>Tove</to>

  <from>Jani</from>

  <heading>Reminder</heading>

  <body>Don't forget me this weekend!</body>

</note>
```

Y aquí está el archivo "note.dtd", que contiene el DTD:

```
<!ELEMENT note (to,from,heading,body)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT from (#PCDATA)>

<!ELEMENT heading (#PCDATA)>

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

Los componentes básicos de los documentos XML

Visto desde el punto de vista de DTD, todos los documentos XML están compuestos por los siguientes bloques de construcción:

- Elementos
- Atributos
- Entidades
- PCDATA
- CDATA

Elementos

Los elementos son los componentes principales de los documentos XML y HTML.

Algunos ejemplos de elementos HTML son "cuerpo" y "tabla". Ejemplos de elementos XML podrían ser "nota" y "mensaje". Los elementos pueden contener texto, otros elementos o estar vacíos. Ejemplos de elementos HTML vacíos son "hr", "br" e "img".

Ejemplos:

```
<body>some text</body>

<message>some text</message>
```

Atributos

Los atributos proporcionan información adicional sobre los elementos .

Los atributos siempre se colocan dentro de la etiqueta de apertura de un elemento. Los atributos siempre vienen en pares de nombre / valor. El siguiente elemento "img" tiene información adicional sobre un archivo fuente:

```
<img src="computer.gif" />
```

El nombre del elemento es "img". El nombre del atributo es "src". El valor del atributo es "computer.gif". Dado que el elemento en sí está vacío, se cierra con una "/".

Entidades

Algunos caracteres tienen un significado especial en XML, como el signo menor que (<) que define el inicio de una etiqueta XML.

La mayoría de ustedes conocen la entidad HTML: " ". Esta entidad "sin espacio de ruptura" se utiliza en HTML para insertar un espacio adicional en un documento. Las entidades se expanden cuando un analizador XML analiza un documento.

Las siguientes entidades están predefinidas en XML:

Entity References	**Character**
&lt;	<
&gt;	>
&amp;	&
&quot;	"
&apos;	'

PCDATA

PCDATA significa datos de caracteres analizados.

Piense en los datos de caracteres como el texto que se encuentra entre la etiqueta inicial y la etiqueta final de un elemento XML.

PCDATA es texto que SERÁ analizado por un analizador . El analizador examinará el texto en busca de entidades y de marcas .

Las etiquetas dentro del texto se tratarán como marcado y las entidades se expandirán.

Sin embargo, los datos de caracteres analizados no deben contener caracteres &, <o>; estos deben estar representados por & < y > entidades, respectivamente.

CDATA

CDATA significa datos de caracteres.

CDATA es texto que NO será analizado por un analizador. Las etiquetas dentro del texto NO se tratarán como marcado y las entidades no se expandirán.

Declarar elementos

En una DTD, los elementos XML se declaran con la siguiente sintaxis:

```
<!ELEMENT element-name category>
```

```
or
```

```
<!ELEMENT element-name (element-content)>
```

Elementos vacíos:

Los elementos vacíos se declaran con la palabra clave de categoría VACÍO:

```
<!ELEMENT element-name EMPTY>
```

Example:

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

XML example:

```
<br/>
```

Elementos con datos de caracteres analizados:

Los elementos con solo datos de caracteres analizados se declaran con #PCDATA entre paréntesis:

```
<!ELEMENT element-name (#PCDATA)>
```

Example:

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

Elementos con cualquier contenido:

Los elementos declarados con la palabra clave de categoría ANY pueden contener cualquier combinación de datos analizables:

```
<!ELEMENT element-name ANY>
```

Example:

```
<!ELEMENT note ANY>
```

Elementos hijo (secuencias):

Los elementos con uno o más elementos secundarios se declaran con el nombre de los elementos secundarios entre paréntesis:

```
<!ELEMENT element-name (child1)>
```

ó

```
<!ELEMENT element-name (child1,child2,...)>
```

Example:

```
<!ELEMENT note (to,from,heading,body)>
```

Cuando los elementos hijo se declaran en una secuencia separada por comas, los hijo deben aparecer en la misma secuencia en el documento. En una declaración completa, los hijos también deben declararse, y los hijos también pueden tener hijos. La declaración completa del elemento "nota" es:

```
<!ELEMENT note (to,from,heading,body)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT from (#PCDATA)>

<!ELEMENT heading (#PCDATA)>

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

Declarar solo una aparición de un elemento:

```
<!ELEMENT element-name (child-name)>
```

Example:

```
<!ELEMENT note (message)>
```

El ejemplo anterior declara que el elemento secundario "mensaje" debe aparecer una vez, y solo una vez dentro del elemento "nota".

Declarar como mínimo una ocurrencia de un elemento

```
<!ELEMENT element-name (child-name+)>
```

Example:

```
<!ELEMENT note (message+)>
```

El signo + en el ejemplo anterior declara que el elemento secundario "mensaje" debe aparecer una o más veces dentro del elemento "nota".

Declarar cero o más ocurrencias de un elemento

```
<!ELEMENT element-name (child-name*)>
```

Example:

```
<!ELEMENT note (message*)>
```

El signo * en el ejemplo anterior declara que el elemento secundario "mensaje" puede aparecer cero o más veces dentro del elemento "nota".

Declarar cero o una apariciones de un elemento

```
<!ELEMENT element-name (child-name?)>
```

```
Example:
```

```
<!ELEMENT note (message?)>
```

El ? firmar en el ejemplo anterior declara que el elemento secundario "mensaje" puede aparecer cero o una vez dentro del elemento "nota".

Declarar contenido

```
<!ELEMENT note (to,from,header,(message|body))>
```

El ejemplo anterior declara que el elemento "nota" debe contener un elemento "para", un elemento "desde", un elemento "encabezado" y un elemento "mensaje" o "cuerpo".

Declaración de contenido mixto

```
<!ELEMENT note (#PCDATA|to|from|header|message)*>
```

El ejemplo anterior declara que el elemento "nota" puede contener cero o más ocurrencias de datos de caracteres analizados, elementos "a", "desde", "encabezado" o "mensaje".

DTD - Atributos

En una DTD, los atributos se declaran con una declaración ATTLIST.

Declaración de atributos:

Una declaración de atributo tiene la siguiente sintaxis:

```
<!ATTLIST element-name attribute-name attribute-type attribute-value>
```

```
DTD example:

<!ATTLIST payment type CDATA "check">

XML example:

<payment type="check" />
```

El tipo de atributo puede ser uno de los siguientes:

Type	Description
CDATA	The value is character data
(*en1*|*en2*|..)	The value must be one from an enumerated list
ID	The value is a unique id

IDREF	The value is the id of another element
IDREFS	The value is a list of other ids
NMTOKEN	The value is a valid XML name
NMTOKENS	The value is a list of valid XML names
ENTITY	The value is an entity
ENTITIES	The value is a list of entities
NOTATION	The value is a name of a notation
xml:	The value is a predefined xml value

El valor-atributo puede ser uno de los siguientes:

Value	**Explanation**
value	The default value of the attribute
#REQUIRED	El atributo es obligatorio

#IMPLIED	El atributo es opcional
#FIXED *value*	El valor de atributo es fijo

Un valor de atributo predeterminado

DTD:

```
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">
```

Valid XML:

```
<square width="100" />
```

En el ejemplo anterior, el elemento "cuadrado" se define como un elemento vacío con un atributo de "ancho" de tipo CDATA. Si no se especifica ningún ancho, tiene un valor predeterminado de 0.

#REQUERIDO

Sintaxis

```
<!ATTLIST element-name attribute-name attribute-type #REQUIRED>
```

Ejemplo

DTD:

```
<!ATTLIST person number CDATA #REQUIRED>
```

Valid XML:

```
<person number="5677" />
```

Invalid XML:

```
<person />
```

Utilice la palabra clave #REQUIRED si no tiene una opción para un valor predeterminado, pero aún desea forzar la presencia del atributo.

#IMPLIED (opcional)

Sintaxis

```
<!ATTLIST element-name attribute-name attribute-type #IMPLIED>
```

Ejemplo

DTD:

```
<!ATTLIST contact fax CDATA #IMPLIED>
```

Valid XML:

```
<contact fax="555-667788" />
```

Valid XML:

```
<contact />
```

Utilice la palabra clave #IMPLIED si no desea forzar al autor a incluir un atributo y no tiene una opción para un valor predeterminado.

#FIXED (fijo)

Sintaxis

```
<!ATTLIST element-name attribute-name attribute-type #FIXED "value">
```

Ejemplo

DTD:

```
<!ATTLIST sender company CDATA #FIXED "Microsoft">
```

Valid XML:

```
<sender company="Microsoft" />
```

Invalid XML:

```
<sender company="W3Schools" />
```

Utilice la palabra clave #FIXED cuando desee que un atributo tenga un valor fijo sin permitir que el autor lo cambie. Si un autor incluye otro valor, el analizador XML devolverá un error.

Valores de atributo enumerados:

Sintaxis

```
<!ATTLIST element-name attribute-name (en1|en2|..) default-value>
```

Ejemplo

DTD:

```
<!ATTLIST payment type (check|cash) "cash">
```

XML example:

```
<payment type="check" />
```

or

```
<payment type="cash" />
```

Utilice valores de atributo enumerados cuando desee que el valor de atributo sea uno de un conjunto fijo de valores legales.

¿Evitar el uso de atributos?

¿Debería evitar el uso de atributos?

Algunos de los problemas con los atributos son:

- los atributos no pueden contener varios valores (los elementos secundarios pueden)
- los atributos no se pueden expandir fácilmente (para cambios futuros)
- los atributos no pueden describir estructuras (los elementos secundarios pueden)
- Los atributos son más difíciles de manipular mediante el código del programa.
- los valores de los atributos no son fáciles de probar contra una DTD

Si usa atributos como contenedores de datos, terminará con documentos que son difíciles de leer y mantener. Intente utilizar elementos para describir datos. Utilice atributos solo para proporcionar información que no sea relevante para los datos.

No termine así (no es así como se debe usar XML):

```
<note day="12" month="11" year="2002"

to="Tove" from="Jani" heading="Reminder"

body="Don't forget me this weekend!">

</note>
```

Una excepción a la regla de atributos

Las reglas siempre tienen excepciones.

La regla sobre atributos tiene una excepción:

A veces asigno referencias de identificación a elementos. Estas referencias de ID se pueden utilizar para acceder a elementos XML de la misma forma que los atributos NAME o ID en HTML. Este ejemplo demuestra esto:

```
<messages>

<note id="p501">

  <to>Tove</to>

  <from>Jani</from>

  <heading>Reminder</heading>

  <body>Don't forget me this weekend!</body>

</note>



<note id="p502">

  <to>Jani</to>

  <from>Tove</from>

  <heading>Re: Reminder</heading>

  <body>I will not!</body>

</note>

</messages>
```

El ID en estos ejemplos es solo un contador, o un identificador único, para identificar las diferentes notas en el archivo XML, y no una parte de los datos de la nota.

Lo que estoy tratando de decir aquí es que los metadatos (datos sobre datos) deben almacenarse como atributos y los datos en sí deben almacenarse como elementos.

DTD - Entidades

Las entidades se utilizan para definir accesos directos a caracteres especiales.

Las entidades pueden declararse internas o externas.

Una declaración de entidad interna

Sintaxis

```
<!ENTITY entity-name "entity-value">
```

Ejemplo

```
DTD Example:



<!ENTITY writer "Donald Duck.">

<!ENTITY copyright "Copyright W3Schools.">



XML example:



<author>&writer;&copyright;</author>
```

Nota: Una entidad tiene tres partes: un ampersand (&), un nombre de entidad y un punto y coma (;).

Una declaración de entidad externa

Sintaxis

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

Ejemplo

DTD Example:

```
<!ENTITY writer SYSTEM "https://www.w3schools.com/entities.dtd">
<!ENTITY copyright SYSTEM "https://www.w3schools.com/entities.dtd">
```

XML example:

```
<author>&writer;&copyright;</author>
```