VALIDACIÓN DE DOCUMENTOS XML

Los documentos bien formados, aseguran que las reglas de XML se cumplen y que **no hay ninguna incoherencia** al usar el lenguaje. Sin embargo, no es suficiente porque podríamos definir documentos que utilizaran los elementos que quisiéramos sin restricción. En la realidad los elementos y los atributos que se pueden utilizar y la manera de disponerlos en el documento son fundamentales para mantener una mayor **homogeneidad**.

Una empresa puede decidir que los documentos internos para describir el software que utiliza la empresa deben poseer como elemento raíz un elemento llamado ***software*** (y no otro), y que este elemento obligatoriamente debe contener los elementos ***nombre***, ***fabricante*** y ***precio***. Esas reglas no se refieren a que el documento esté bien formado; son reglas más complejas y que permitirán al documento que sea válido.

Para ello se crea un documento que contendrá las reglas que deben de cumplir los XML que se basen en él. De modo que **un documento deberá indicar qué plantilla de reglas utiliza y deberá cumplirlas a rajatabla para considerarse válido**.

Con la validación tenemos la seguridad de que los documentos cumplen unas reglas más concretas y de esa forma es fácil establecer un protocolo en las empresas para sus documentos. De hecho, cuando un documento XML cumple estrictamente las reglas generales de creación XML, se dice que **está bien formado**; cuando además sigue las reglas de un documento de validación entonces se dice que es **válido**.

Las técnicas más populares para validar documentos son:

- **DTD**, ***Document Type Definition***. Validación por documentos de definición de tipos. Se utilizaba en el lenguaje SGML y de ahí debe su popularidad. Es la más utilizada, pero tiene numerosas voces críticas porque su sintaxis no es XML.
- **XML Schema** o esquemas XML. Mucho más coherente con el lenguaje XML, es la aconsejada actualmente.
- **Relax NG**. Es una notación sencilla y fácil de aprender que está haciéndose muy popular. No tiene tantas posibilidades con el XML Schema, pero tiene una sintaxis muy sencilla. Además admite añadir instrucciones de tipo XML Schema por lo que se convierte en una de las formas de validación más completas.
- **Schematron**. Permite establecer reglas que facilitan establecer las relaciones que han de cumplir los datos de un documento XML. No es

tan bueno para establecer el resto de reglas de validación (orden de elementos, tipos de datos,…).

DTD

1. POSIBILIDADES DE USO DE UN DTD

- **DTD 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 esta definición sólo vale para dicho documento. Sintaxis:

```
<!DOCTYPE raíz [....códigoDTD...]>
```

Dentro de los símbolos **[** y **]** se especifican las instrucciones DTD. Ejemplo:

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

- **DTD en un documento externo privado**

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

```
<!DOCTYPE raíz SYSTEM "rutaURLalDTD">
```

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">
```

En ambos casos se puede **añadir código DTD** para en ese documento concreto, añadir instrucciones de validación. Ejemplo:

```
<!DOCTYPE raíz SYSTEM "http://www.empresa.com/docs.dtd" [
<!ELEMENT nombre (#PCDATA)>
] >
```

- **DTD externo de tipo public**

Si se trata de un documento de uso público, entonces se usa PUBLIC. La sintaxis sería:

```
<!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. Si disponemos de un repositorio de DTDs públicos (como ocurre en entornos de trabajo como **Oxygene** por ejemplo) le cargaría sin ir a Internet. Si el **nombreDTD** no es reconocido se usa la dirección URL para descargarlo y utilizarlo. Ejemplo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Este es el DOCTYPE para una página web escrita en XHTML 1.0 estricto, utilizada para validar miles de páginas web.

- **Atributo *standalone* de la etiqueta de cabecera**

La etiqueta **<?xml** de cabecera de todo documento XML, posee un atributo llamado **standalone** que puede tomar dos valores:

- **yes**. En caso de que el documento XML no utilice DTD externa.
- **no**. Cuando el documento obligatoriamente hace uso de DTD externa.

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. Además incluso indicando sus posibles valores válidos.
- **Entidades** que puede utilizar el documento XML.

2. ELEMENTOS

Mediante un DTD podemos especificar elemento que se puede utilizar en un XML se define en su DTD mediante una etiqueta !ELEMENT. La sintaxis es:

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

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. A continuación se indican las posibilidades de este parámetro:

- **EMPTY**

Significa que el elemento no podrá tener contenido alguno (aunque sí atributos). Es un elemento vacío (como la etiqueta <br/> de las páginas web). Ejemplo de definición de elemento vacío:

```
<!ELEMENT línea EMPTY >
```

- **ANY**

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

Aquí, la palabra clave 'ANY' indica que el texto (PCDATA) y/o cualquier elemento declarado en la DTD puede usarse en el contenido del <elementname> elemento. Se pueden usar en cualquier orden, las veces que se quiera. Sin embargo, la palabra clave 'ANY' no permite incluir elementos que no están declarados en la DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
<!ELEMENT persona (nombre, apellidos)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos ANY>
]>
<persona>
<nombre>Pepe</nombre>
<apellidos>Sanz
        <nombre>Pepa</nombre>
</apellidos>
</persona>
```

- **ELEMENTOS CONCRETOS**

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

Se indican entre paréntesis:

- En el ejemplo dentro de una etiqueta ***persona*** **obligatoriamente** debe de existir una etiqueta ***nombre*** (**una y sólo una**).
- La indicación **#PCDATA** significa que el elemento podrá contener texto literal (tan largo como se desee).

- **SECUENCIAS**

Lista de elementos separados por comas:

```
<?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>
```

"persona", deberá tener la lista indicada, en el mismo orden que el indicado.

- **ELECCIONES**

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

El elemento "articulo" puede contener una u otra opción, pero no ambas a la vez.
Si dentro de la lista de opciones aparece #PCDATA, ésta debe ser la primera opción.

Nos podemos encontrar con combinaciones de los elementos anteriores, p.e.:

Fichero: coordenada.dtd

```
<!ELEMENT coordenada ((longitud, latitud) | coordUniversal)>
<!ELEMENT longitud (#PCDATA)>
<!ELEMENT latitud (#PCDATA)>
<!ELEMENT coordUniversal (#PCDATA)>
```

Serían válidos los siguientes documentos XML:

Fichero: coordenada1.xml

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

Fichero: coordenada2.dtd

```
<?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 (0 ó 1)

+ Contenido obligatorio y repetible. Tiene que aparecer al menos una vez, e incluso puede aparecer varias veces (1 ó muchos).

***** Contenido opcional y repetible. Es decir puede no aparecer y puede aparecer varias veces (0 ó muchos).

Por ejemplo:

Fichero: película.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pelicula [
<!ELEMENT pelicula (titulo, direccion+, argumento?, actor*)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT direccion (#PCDATA)>
<!ELEMENT argumento (#PCDATA)>
<!ELEMENT actor (#PCDATA)>
]>

<pelicula>
   <titulo> La guerra de las galaxias </titulo>
   <direccion> Galaxia </direccion>
</pelicula>
```

3. ATRIBUTOS

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. Sintaxis:

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

- ***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 dar un valor que el atributo tomará en el documento XML en caso de que no se le dé en el mismo ningún valor al atributo. También indica si es necesario rellenar o no el atributo o bien si es opcional.

Ejemplo de declaración:

Declaración en un DTD: <!ATTLIST persona nacionalidad CDATA>
Nota: el tipo CDATA quiere decir que es de texto normal.
Uso en un fichero XML: <persona nacionalidad=”española”>

- **TIPOS DE ATRIBUTOS**

  - ***CDATA***

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 (PCDATA no admite caracteres inválidos para la sintaxis de XML, como el signo < por ejemplo, CDATA sí los admite).

  - ***ID***

El valor del atributo servirá para identificar al elemento que le contiene. Los IDs deben cumplir:

- El valor tiene que cumplir las mismas reglas que para especificar nombres XML.
- No puede haber dos etiquetas con el mismo ID en un mismo documento XML
- En el DTD, para cada elemento sólo puede indicarse un atributo como ID.
- Los atributos ID sólo pueden indicar **#IMPLIED** o **#REQUIRED** en el apartado del valor por defecto.

  - ***IDREF***

- El atributo contendrá el nombre de un ID de otro elemento. Es decir será una referencia a otra etiqueta. Las reglas de los IDREFs son:
- El valor de un IDREF debe cumplir las reglas para especificar nombres XML
- Debe existir un atributo ID en el documento XML cuyo valor coincida con él (de otro modo se haría referencia a una etiqueta inexistente y esto no está permitido)

Fichero: directorio.dtd

```
<!ELEMENT directorio (persona)+ >
 <!ELEMENT persona (#PCDATA) >
<!ATTLIST persona id ID #REQUIRED madre IDREF #IMPLIED padre IDREF #IMPLIED>
```

Fichero: directorio.xml

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

  - ***IDREFS***

Igual que el anterior salvo que permite indicar varias referencias a otros IDs, separados por espacios en blanco. Ejemplo de uso:

Fichero: directorio.dtd

```
<!ELEMENT directorio (persona)+ >
```

```
 <!ELEMENT persona (#PCDATA) >
<!ATTLIST persona id ID #REQUIRED padres IDREFS #IMPLIED>
 Fichero: directorio.xml
<?xml version="1.0" encoding="UTF-8"?>
<!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***

Texto que sólo podrá tener letras, dígitos, guion, subrayado, punto y dos puntos. Es lo que se llama nombres válidos XML.

- ***NMTOKENS***

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

- ***ENTITY***

El tipo del atributo es una entidad que se ha declarado anteriormente.

- ***ENTITIES***

Lista de entidades.

- ***Enumeración***

En este caso el valor del atributo debe de ser uno de una lista de valores posibles cada uno de los cuales se separa del siguiente mediante el símbolo **|**. Ejemplo de uso:

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

- ***Agrupación de atributos en la misma etiqueta***

Podríamos, en la misma etiqueta ATTLIST, declarar varios tipos de atributos. De esta forma, sería correcta la siguiente declaración:

```
<!ATTLIST persona nacionalidad CDATA "Española" sexo (Hombre | Mujer) #IMPLIED id
ID #REQUIRED>
```

```
<persona nacionalidad="Francesa" id="A1234">Vivian Maret</persona>
```

- **VALORES POR DEFECTO**

  - ***Valor por defecto concreto***

Se indica al final de la declaración de un atributo. Si no se indica el atributo en el elemento, toma éste valor.

```
<!ELEMENT directorio (persona)+>
 <!ELEMENT persona (#PCDATA)>
<!ATTLIST persona nacionalidad CDATA "Española">
```

  - ***Valores fijos (#FIXED)***

En este caso, en ningún documento XML se podrá modificar éste atributo.
Ejemplo de uso:

```
<!ELEMENT directorio (persona)+>
<!ELEMENT persona (#PCDATA)>
<!ATTLIST persona nacionalidad CDATA #FIXED "Española">
```

- ***Valores requeridos (#REQUIRED)***

Se usa para indicar que al atributo hay que especificarle en el documento XML algún valor. Ejemplo de uso:

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

- ***Valores opcionales (#IMPLIED)***

Se usa para indicar que el atributo puede quedarse sin valor. Ejemplo de uso:

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

4. ENTIDADES

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.

ENTIDADES QUE SE USAN EN DOCUMENTOS XML

- ***Entidades ya existentes***

No hay que declararlas, ya que todos los analizadores XML las conocen. Son:

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

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

- ***Entidades para referencia 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;
```

Ejemplo de uso:

```
<calle>Kantstra&#225;e, Berlín</calle>
```

En el navegador este elemento aparecería como:

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

Nota: si sobre cualquier editor de textos pulsamos la tecla ALT+225 (haciendo uso del teclado numérico), se representa dicho carácter (ß).

- ***Entidades generales***

Se usan como abreviaturas que aparecerán en el documento XML. La razón de su uso es facilitar la escritura de nombres repetitivos. Sintaxis:

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

Ejemplo de declaración y uso de una entidad:

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

 <!DOCTYPE texto[
        <!ELEMENT texto (titulo?)>
        <!ELEMENT titulo (#PCDATA)>
        <!ENTITY alf "Alien Life Form">
      ]>
<texto><titulo>Un día en la vida de un &alf;</titulo></texto>
```

Nota: si ejecutamos dicho fichero desde un navegador WEB, se sustituirá **&alf;** por la cadena correspondiente.

También se pueden usar las entidades en este sentido:

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

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.***

ENTIDADES QUE SE USAN EN DOCUMENTOS DTD

- ***Entidades de parámetros internas y externas***

Sólo 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. 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.

Un ejemplo de declaración de entidad **interna** podría ser:

```
<!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, directorio) >
<!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.**

Otro ejemplo de uso de ambas entidades:

Uso entidad interna

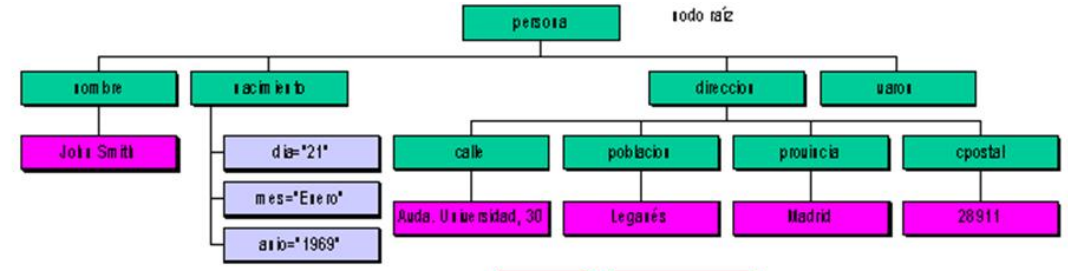
```
<!DOCTYPE texto[
<!ENTITY %elemento-alf "<!ELEMENT ALF (#PCDATA)>">
...
%elemento-alf;
]>
```

Uso entidad externa

```
<!DOCTYPE texto[
<!ENTITY %elemento-alf SYSTEM "alf.ent">
...
%elemento-alf;
]>
```

Nota: es importante terminar el uso de la entidad con ';'.

Ejemplo de jerarquía de datos y el DTD que la representa:



Nodos de elementos

Nodos hoja (contenido)

Nodos de atributos

Secuencia | 0 o 1 elementos | 1o más elementos | Disyunción

```
<!ELEMENT persona (nombre, nacimiento?, direccion+, (varon|hembra))>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT nacimiento EMPTY>
<!ATTLIST nacimiento
          dia      CDATA    #REQUIRED
          mes      CDATA    #REQUIRED
          anio     CDATA    #REQUIRED>
<!ELEMENT direccion (calle,poblacion,provincia, cpostal)>
<!ELEMENT calle (#PCDATA)>
<!ELEMENT poblacion (#PCDATA)>
<!ELEMENT provincia (#PCDATA)>
<!ELEMENT cpostal (#PCDATA)>
<!ELEMENT varon EMPTY>
<!ELEMENT hembra EMPTY>
```

DTD (fragmento)