Validación de XML

JUAN CARLOS CONDE RAMÍREZ

WEB-TECHNOLOGIES

Objetivos

- Comprender qué es la validación XML y su importancia.
- Identificar la diferencia ente un XML “bien formado” y un XML “válido”.
- Entender qué son las Definiciones de Tipo de Documento (DTD)
- Conocer la sintaxis para poder crear DTD en la validación de XMLs.
- Visualizar las diferencias básicas entre un DTD y un Schema.
- Conocer cuáles son las ventajas de los Schemas vs las DTD.

Definición del tipo de documento I

- *Document Type Definition* (DTD)

- Crear una definición del tipo de documento (DTD) es como crear nuestro propio lenguaje de marcado, para una aplicación específica.

- La DTD define los siguientes componentes y algunas limitaciones para combinarlos:
  - Elementos
  - Atributos
  - Entidades

Definición del tipo de documento II

- Un DTD puede residir en un fichero externo, y ser compartido por varios (puede que miles) de documentos.

- O bien, puede estar contenida en el propio documento XML, como parte de su declaración de tipo de documento.

- Los documentos XML que se ajustan a su DTD se denominan "válidos".

Definición del tipo de documento III

- El concepto de "validez" no tiene nada que ver con el de estar "bien-formado".
- Un documento "bien-fomado" simplemente respeta la estructura y sintaxis definidas por la especificación de XML.
- Un documento "bien-formado" puede además ser "válido" si cumple las reglas de una DTD determinada.

Definición del tipo de documento IV

- También existen documentos XML sin una DTD asociada, en ese caso no son "válidos", pero tampoco "inválidos“, simplemente "bien-formados“ o no.

```
<?xml version="1.0"?>
<!DOCTYPE etiqueta[
    <!ELEMENT etiqueta (nombre, calle, ciudad, pais, codigo)>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT calle (#PCDATA)>
    <!ELEMENT ciudad (#PCDATA)>
    <!ELEMENT pais (#PCDATA)>
    <!ELEMENT codigo (#PCDATA)>
]>
<etiqueta>
    <nombre>Fulgencio Menéndez</nombre>
    <calle>15 Sur, 27</calle>
    <ciudad>Puebla</ciudad>
    <pais>Mexico</pais>
    <codigo>72200</codigo>
</etiqueta>
```

Definición del tipo de documento V

- En el ejemplo anterior, todas las declaraciones DTD que definen "etiqueta" residen dentro del documento.

- Sin embargo, la DTD se puede definir parcial o completamente en otro lugar. Por ejemplo:

```
<?xml version="1.0"?>
<!DOCTYPE etiqueta SYSTEM "http://sitio.com/persona.dtd">
<etiqueta>
    <nombre>Fulgencio Menéndez</nombre>
    <calle>15 Sur, 27</calle>
    <ciudad>Puebla</ciudad>
    <pais>Mexico</pais>
    <codigo>72200</codigo>
</etiqueta>
```

Declaraciones de tipo de Elemento I

- Los *elementos* son la base de las marcas XML, y deben ajustarse a un tipo de documento declarado en una DTD para que el documento XML sea considerado válido.

- Las declaraciones de tipo de elemento deben empezar con `<!ELEMENT` seguidas por el identificador genérico del elemento que se declara.

- A continuación tienen una especificación de contenido:

```
<!ELEMENT receta (titulo, ingredientes, procedimiento)>
```

Declaraciones de tipo de Elemento II

- Siguiendo la definición de elemento anterior, este ejemplo de documento XML sería válido:

```
<receta>
    <titulo>...</titulo>
    <ingredientes>...</ingredientes>
    <procedimiento>...</procedimiento>
</receta>
```

- Pero no este otro:

```
<receta>
    <parrafo>Esto es un párrafo</parrafo>
    <titulo>...</titulo>
    <ingredientes>...</ingredientes>
    <procedimiento>...</procedimiento>
</receta>
```

Declaraciones de tipo de Elemento III

- La especificación de contenido puede ser de cuatro tipos:

1. **EMPTY** Puede no tener contenido. Suele usarse cuando contiene sólo atributos.

```
<!ELEMENT salto-de-pagina EMPTY>
```

2. **ANY** Puede tener cualquier contenido. No se suele utilizar, ya que es conveniente especificar adecuadamente nuestros documentos XML.

```
<!ELEMENT batidillo ANY>
```

Declaraciones de tipo de Elemento IV

3. **Mixed** puede tener caracteres de tipo datos o una mezcla de caracteres y subelementos

```
<!ELEMENT enfasis (#PCDATA)>
<!ELEMENT parrafo (#PCDATA|enfasis)*>
```

4. **Element** sólo puede contener sub-elementos especificados en la especificación de contenido.

```
<!ELEMENT mensaje (remite, destinatario, texto)>
```

- Para declarar que un tipo de elemento tenga contenido de elementos se especifica un modelo de contenido en lugar de una especificación de contenido mixto o una de las claves ya descritas.

Modelos de contenido I

<!ELEMENT aviso (parrafo)>

- Esto indica que <aviso> sólo puede contener un solo <parrafo>

<!ELEMENT aviso (titulo, parrafo)>

- La coma, en este caso, denota una secuencia. Es decir, el elemento <aviso> debe contener un <titulo> seguido de un <parrafo>.

<!ELEMENT aviso (parrafo | grafico)>

- La barra vertical "|" indica una opción. Es decir, <aviso> puede contener o bien un <parrafo> o bien un <grafico>. El número de opciones no está limitado a dos, y se pueden agrupar usando paréntesis.

Modelos de contenido II

```
<!ELEMENT aviso (titulo, (parrafo | grafico))>
```

- En este último caso, el `<aviso>` debe contener un `<titulo>` seguido de un `<parrafo>` o de un `<grafico>`.

- Además, cada partícula de contenido puede llevar un indicador de frecuencia, que siguen directamente a un identificador general, una secuencia o una opción, y no pueden ir precedidos por espacios en blanco.

Indicadores de Frecuencia		
?	0 o 1 vez	Opcional
*	0 o más veces	Opcional
+	1 o más veces	Necesario

Modelos de contenido III

```
<!ELEMENT aviso (titulo?, (parrafo+, grafico)*)>
```

- En este caso, `<aviso>` puede tener `<titulo>`, o no (pero sólo uno), y puede tener cero o más conjuntos `<parrafo><grafico>`, `<parrafo><parrafo><grafico>`, etc.

Declaraciones de Atributos I

- Los atributos permiten añadir información adicional a los elementos de un documento.
- La principal diferencia entre los elementos y los atributos, es que los atributos no pueden contener sub-atributos.
- Se usan para añadir información corta, sencilla y desestructurada.

Declaraciones de Atributos II

- Por ejemplo:

```
<mensaje prioridad="urgente">
    <de>Alfred Hitchock</de>
    <a>Hans van Parijs</a>
    <texto idioma="holandés">
        Hallo Hans, hoe gaat het?
        ...
    </texto>
</mensaje>
```

- Otra diferencia entre los atributos y los elementos, es que cada uno de los atributos sólo se puede especificar una vez, y en cualquier orden.

Declaraciones de Atributos III

- En el ejemplo anterior, para declarar la lista de atributos de los elementos `<mensaje>` y `<texto>` haríamos lo siguiente:

```
<!ELEMENT mensaje (de, a, texto)>
<!ATTLIST mensaje prioridad (normal|urgente) “normal”>
<!ELEMENT texto(#PCDATA)>
<!ATTLIST texto idioma CDATA #REQUIRED>
```

- Las declaraciones de los atributos empiezan con `<!ATTLIST`, y a continuación del espacio en blanco viene el identificador del elemento al que se aplica el atributo.

- Después viene el nombre del atributo, su tipo y su valor por defecto.

Declaraciones de Atributos IV

- En el ejemplo anterior, el atributo `prioridad` puede estar en el elemento `<mensaje>` y puede tener el valor `normal` o `urgente`, siendo `normal` el valor por defecto si no especificamos el atributo.

- El atributo `idioma`, pertenece al elemento texto, y puede contener datos de carácter (`CDATA`).

- Es más, la palabra `#REQUIRED` significa que no tiene valor por defecto, ya que es obligatorio especificar este atributo.

Declaraciones de Atributos V

- A menudo interesa que se pueda omitir un atributo, sin que se adopte automáticamente un valor por defecto.

- Para esto se usa la condición `#IMPLIED`.

- Por ejemplo, en una supuesta DTD que defina la etiqueta `<IMG>` de HTML:

```
<!ATTLIST IMG
URL CDATA #REQUIRED
ALT CDATA #IMPLIED>
```

Tipos de Atributos I

- Atributos `CDATA` y `NMTOKEN`
  - Los atributos `CDATA` (*character data*) son los más sencillos, y pueden contener casi cualquier cosa. Los atributos `NMTOKEN` (*name token*) son parecidos, pero sólo aceptan los caracteres válidos para nombrar cosas (letras, números, puntos, guiones, subrayados y los dos puntos).

```
<!ATTLIST mensaje fecha CDATA #REQUIRED>

<mensaje fecha="15 de Julio de 2019">



<!ATTLIST mensaje fecha NMTOKEN #REQUIRED>

<mensaje fecha="15-7-2019">
```

Tipos de Atributos II

- Atributos Enumerados
  - Los atributos enumerados sólo pueden contener un valor de entre un número reducido de opciones.

```
<!ATTLIST mensaje prioridad (normal|urgente) “normal”>
```

- Atributos ID e IDREF
  - ID permite que un atributo determinado tenga un nombre único que podrá ser referenciado por un atributo de otro elemento que sea de tipo IDREF.
  - Por ejemplo, para implementar un sencillo sistema de hipervínculos en un documento:

```
<!ELEMENT enlace EMPTY>
<!ATTLIST enlace destino IDREF #REQUIRED>
<!ELEMENT capitulo (parrafo)*>
<!ATTLIST capitulo referencia ID #IMPLIED>
```

  - donde destino y referencia deben contener el mismo valor y además debe ser alfa-numérico (p.e. *x001*).

Declaración de entidades I

- XML hace referencia a objetos (archivo, páginas Web, imágenes, cualquier cosa) que no deben q 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 ser sólo una abreviatura utilizada como una forma corta de utilizar algunos textos.

Declaración de entidades II

- 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

Tipo de entidades I

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

```
<!DOCTYPE texto[
<!ENTITY ovni "Objeto Volador No identificado">
]>

<texto>
        <titulo>Un día en la vida de un &ovni;</titulo>
</texto>
```

Tipo de entidades II

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

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

- **Entidades no analizadas**
  - Evidentemente, si el contenido de la entidad es un archivo MPEG o una imagen GIF o un archivo ejecutable EXE, el procesador XML no debería intenta interpretarlo como si fuera texto XML. Este tipo de entidades siempre son generales y externas.

```
<!ENTITY logo SYSTEM “http://server.com/logo.gif”>
```

Tipo de entidades III

- **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.

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

Tipo de entidades IV

- Entidades parámetro externas
  - Igualmente, las entidades parámetro, pueden ser externas.

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

Ejemplo DTD completo (LISTA.dtd)

```
<!ELEMENT lista (persona)+>
<!ELEMENT persona (nombre, email*, relacion?)>
<!ATTLIST persona id #REQUIRED>
<!ATTLIST persona sexo (hombre | mujer) #IMPLIED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT relacion EMPTY>
<!ATTLIST relacion amigo-de IDREFS #IMPLIED
enemigo-de IDREFS #IMPLIED>
```

XML validado con el DTD anterior

```
<?xml version="1.0"?>
<!DOCTYPE lista SYSTEM "LISTA.DTD">
<lista>
    <persona sexo="hombre" id="alvaro">
        <nombre>Álvaro Álvarado</nombre>
        <email>alvaroal@gmail.com</email>
        <relacion amigo-de="Beatriz"/>
    </persona>
    <persona sexo="mujer" id="Beatriz">
        <nombre>Beatriz Barrientos</nombre>
        <email>betyb@yahoo.com</email>
    </persona>
</lista>
```

Introducción a *XML Schemas I*

- Un "*schema XML*" es algo similar a un DTD, es decir, que define qué elementos puede contener un documento XML, cómo están organizados, y que atributos y de qué tipo pueden tener sus elementos.

- La ventaja de los *schemas* con respecto a las DTDs son:

  - Usan sintaxis de XML, al contrario que los DTDs.
  - Permiten especificar los tipos de datos.
  - Son extensibles.

Introducción *XML Schemas II*

- Analicemos el siguiente ejemplo de un documento XML, y su schema correspondiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<documento xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="documento.xsd">
    <persona id="Fulgencio">
        <nombre>Fulgencio Menéndez</nombre>
    </persona>
</documento>
```

Introducción a *XML Schemas III*

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:complexType name="personaType">
        <xsd:all>
            <xsd:element name="nombre" type="xsd:string"/>
        </xsd:all>
        <xsd:attribute name="id" type="xsd:string" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="documentoType" mixed="false">
        <xsd:all>
            <xsd:element name="persona" type="personaType"/>
        </xsd:all>
    </xsd:complexType>

    <xsd:element name="documento" type="documentoType"/>

</xsd:schema>
```

Introducción a *XML Schemas IV*

- `xsd:complexType` Define el tipo y contenido de un elemento, el cual puede incluir texto, atributos o más elementos, así como conjuntos y secuencias* de los mismos.

- `xsd:attribute` Declara que un atributo de un tipo base (de *xsd*) o de un tipo simple previamente definido como `xsd:simpleType` (no en este ejemplo); y que debe aparecer como atributo de un determinado elemento.

- `xsd:element` Declara que un elemento previamente definido por `xsd:complexType` puede aparecer como contenido de otro elemento.

- `xsd:all` Delimita un conjunto de elementos.

Introducción a *XML Schemas IV*

- Tal como hemos visto, es necesario empezar el *schema* definiendo los elementos más profundamente anidados dentro de la estructura jerárquica de elementos del documento XML.

- Es decir, tenemos que trabajar "de dentro hacia fuera". Visto de otra manera, las declaraciones de tipo `xsd:complexType` y `xsd:simpleType` deben preceder a las declaraciones de contenido `xsd:element` y `xsd:attribute` correspondientes.

- Una página útil para checar más detalles de XML Schemas, es:

http://www.w3schools.com/schema/