Lenguajes de Marcado: SGML, HTML, XML

Lenguaje SGML

El lenguaje de marcado generalizado estándar o SGML (por sus siglas en inglés de Standard Generalized Markup Language) (SGML; ISO 8879: 1986) es un estándar para definir lenguajes de marcado generalizados para documentos. De forma más coloquial, se trata de un lenguaje para marcar y describir documentos con independencia total del hardware y software utilizados. Los lenguajes de marcado no son equivalentes a los lenguajes de programación, pero este nombre genera confusión por la palabra lenguaje

- El marcado debe ser declarativo, es decir, debe describir el contenido y atributos del documento, en lugar de decir cómo debe ser procesado.
- El marcado debe ser riguroso.

HTML era un ejemplo de este lenguaje hasta la versión 5, en la cual se permite que un documento codifique directamente lo que se debe hacer. Docbook SGML y LinuxDoc son ejemplos de documentos SGML

Un documento SGML se compone de 3 partes:

- Declaración SGML
- Declaración de tipo de documento (DTD)
- Instancia de documento.

La declaración SGML le dice al usuario que puede y que no puede hacer. Se trata de un diagrama formal y normalizado que le dice al receptor el conjunto de caracteres, los delimitadores y las características opcionales que se van a utilizar en el documento. La declaración SGML es necesaria para cada documento que se transmite, sin embargo suele omitirse cuando tanto el sistema emisor como el receptor utilizan sintaxis por defecto o una sintaxis de referencia concreta. La DTD identifica la estrucutra del documento. Contiene las reglas, el nombre y su significado. Puede escribirse dentro de un documento, o en un fichero separado, y ser referenciado dentro del archivo SGML. Ejemplo de un DTD interno

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE address [
   <!ELEMENT address (name,company,phone)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT company (#PCDATA)>
   <!ELEMENT phone (#PCDATA)>
]>
<address>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</address>
```

Justo despues de la declaración, nos encontramos con el *document type declaration*, llamado más comunmente DOCTYPE. Después de esto, viene el cuerpo del DTD, donde se declaran los elementos, atributos, entidades y anotaciones.

```
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
```

Normas:

- La declaración de tipo de documento debe aparecer al principio (solamente precedido de un encabezado, en este caso XML).
- Las declaraciones de elemento deben empezar con un signo de exclamación.
- El nombre en la declaración del tipo de documento debe coincidir con el tipo de elemento del elemento raíz.

Veamos ahora un ejemplo de DTD externo:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE address SYSTEM "address.dtd">
<address>
  <name>Tanmay Patil</name>
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</address>
```

donde address.dtd es un archivo de la siguiente forma:

```
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
```

Comandos que se pueden utilizar en un DTD:

ELEMENT: Sirve para definir un elemento. Su forma básica es:

```
<!ELEMENT nombreelemento(contenido)>
```

El contenido de un elemento puede ser:

- EMPTY: El elemento está vacío (ojo, puede contener atributos). Ej
- ANY: El elemento puede almacenar cualquier tipo de contenido
- Otros elementos: Contiene otros elementos en si interior: Ej:
- #PCDATA: Esto es texto a procesar.
- Mixto. El elemento puede incluír secuencias de caracteres opcionalmente mezcladas con elementos hijos. Ej

Cardinalidad: Si un elemento se puede repetir 1 o más veces, utilizamos el símbolo + Si un elemento se puede repetir 0 o más veces, utilizamos * Si un elemento se puede repetir 0 o 1 vez: ? Si tenemos que elegir uno entre una lista: |

ATTLIST: Listas de atributos.

Los atributos definen una propiedad de un elemento. Su declaración en muy parecida a la de los elementos: Sintaxis.

```
<!ATTLIST nombre_elemento nombre_atributo tipo_atributo valor_atributo>
```

Veamos esto en una declaración completa:

```
<?xml version = "1.0"?>
<!DOCTYPE address [
<!ELEMENT address ( name )>
<!ELEMENT name ( #PCDATA )>
<!ATTLIST name id CDATA #REQUIRED>
 ]>
<address>
   <name id="123">Tanmay Patil</name>
</address>
```

En esta declaración indicamos que el elemento name tiene un atributo id de tipo CDATA (datos formados por caracteres), y su valor es obligatorio. Los tipos posibles son:

Tipo	Descripción
CDATA	Texto
ID	Identificador. No debe aparecer más de una vez. Es un tipo de atributo de caso
IDREF	Se usa para hacer referencia a la identidad de otro elemento
IDREFS	Se usa para referenciar múltiples identidades
ENTIDAD	Representa una entidad externa en el documento
ENTIDADES	Representa una lista de entidades externas en el documento
NMTOKEN	Similar a CDATA y el valor del atributo consisten en un nombre xml válido
NMTOKENS	Similar a CDATA y el valor del atributo consiste en una lista de nombres xml válidos
ANOTACIÓN	Un elemento será referenciado a una anotación declarada en el documento
ENUMERACIÓN	Permite definir un aspecto de la lsita de valores donde uno de los valores debe coincidir

Lenguaje XML

XMl es un subconjunto de SGML, lo que significa que toma todos los pros de SGML, sin embargo ha sido diseñado para que su análisis sea mucho más fácil. Es como una versión más simplificada. Por ejemplo:

- Las definiciones SHORTREF y USEMAP no se pueden dar en XML.
- Los nombres diferencia mayúsculas de minúsculas (en SGML se puede definir esto).
- En los nombres se permite _
- Los nombres pueden utilizar caracteres UNICODE, y no están restringidos a ASCII.
- En XML no se permite dejar etiquetas abiertas.
- Tampoco están permitidas las etiquetas vacías.

Podemos ver un listado de todas las diferencias en [Link](https://www.w3.org/TR/NOTE-sgml-xml-971215/Comparison of SGML and XML).