XML SCHEMA

Es un lenguaje de esquema utilizado para describir las restricciones y la estructura de la información de los documentos de una forma muy exacta. Más allá de la sintaxis se consigue una percepción del tipo de documento con un nivel superior de abstracción, su mayor nivel fue alcanzado en mayo del 2001, tras su creación por parte de World Wide Web Consortium (W3C).

Es importante tener en cuenta las siguientes recomendaciones dentro del contexto de descripción de documentos.

- XML Schema es el nombre oficial otorgado por W3C que elaboro el primer lenguaje de Schema separado de XML.
- Habitualmente escuchamos referirnos a este como "XML Schema" pero se reocmienda utilizar frases para llamarlo como "documento esquema" o "definición de documento" para así dejar el nombre base como reconocimiento del lenguaje como tal.
- El nombre técnico de los lenguajes XML es conocido como "Schema Definition Languaje" para:
    - Definición de tipo de documento.
    - XML Schema
    - Relax NG
    - Schematron
    - Nemespace Routing Languages (NRL)
    - Document Schema Definition Languages (DSDL)
    - Document Definition Markup Language (DDML)
    - Document Structure Description (DSD)
    - SGML
    - Schema for Object-Oriented XML (SOX).

Los primeros trabajos de W3C en XML comenzarón en 1998. La primera versión se convirtió en una recomendación oficial en mayo de 2001. Una segunda edición revisada está disponible desde octubre de 2004. La cual está dividida por tres segmentos:

- **XML Schema Parte 0 Primer:** Nos proporciona una gran cantidad de ejemplos introductorios al mundo XML detallados no normativos.

- **XML Schema Parte 1 Structures:** Nos presente una descripción de cada uno de los componentes de forma muy extensa.
- **XML Schema parte 2 Datatypes:** Nos complemente la primer parte con la definición de cada uno de los tipos de datos incorporados en XML Schema y sus respectivas restricciones.

El propósito general o introductorio del estándar XML Schema es definir la estructura de los documentos XML que estén asignados a tal esquema y los tipos de datos válidos para cada elemento y atributo. En este sentido las posibilidades de control sobre la estructura y los tipos de datos son muy amplias.

Al restringir el contenido, datos o información de los ficheros XML es posible intercambiar información entre aplicaciones con mayor seguridad y eficiencia. Disminuye el trabajo de comprobar la estructura de los ficheros y el tipo de los datos. XML Schema tiene un enfoque modular que recuerda a la programación orientada a objetos y que facilita la reutilización de código.

Los tipos de datos tienen en XML Schema la función de las clases en la POO. El usuario puede construir tipos de datos a partir de tipos predefinidos, agrupando elementos y atributos de una determinada forma y con mecanismos de extensión parecidos a la herencia. Los tipos de datos se clasifican en función de los elementos y atributos que contienen.

TIPOS DE DATOS:

Podemos definir dos tipos de datos en XML Schema, Simples o Complejos.

Tipos Simples: se definen como simples aquellos no tienen elementos hijos ni atributos:

- Tipos predefinidos de XML: string, double, boolean, etc.
- List (lista de datos separados por espacios).
- Unión (tipo de dato derivado de la unión de tipos predefinidos).

Tipos Complejos: se definen como complejos aquellos que tienen elementos hijos y/o atributos.

Pueden tener nombre o ser anónimos. Si tienen nombre pueden ser reutilizados dentro del mismo XML Schema o por otros XML Schemas. Es posible "mezclar" o combinar elementos y texto.

EJEMPLO XML SHEMA:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="Libro">
      <xsd:complexType>
        <xsd:sequence>
           <xsd:element name="Título" type="xsd:string"/>
           <xsd:element name="Autores" type="xsd:string" maxOccurs="10"/>
           <xsd:element name="Editorial" type="xsd:string"/>
        </xsd:sequence>
        <xsd:attribute name="precio" type="xsd:double"/>
     </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

El elemento raíz se llama libro y tiene tres hijos y un atributo. Los hijos son Título, Editorial que deben aparecer una vez y Autores que pueden aparecer de una a diez veces. El hecho de que estén agrupados en una secuencia indica que los elementos deben aparecer en orden, es decir, primero el Título, luego los Autores y por último la Editorial. Los tres elementos son de tipo string. El atributo de libro se llama precio y es de tipo double.

DTD (DOCUMENT TYPE DEFINITION)

Es conocida mundialmente como una descripción de estructura dela sintaxis previamente definida XML. Su función general consiste en la descripción de la estructura de la información o datos, para usar la estructura común y mantener la consistencia de los datos entre cada uno de los documentos que utilicen la misma DTD, para así poder validarlos, conocen la estructura de los elementos y la descripción de los datos que trae consigo cada documento, y pueden además compartir la misma descripción y forma de validación dentro de un grupo de trabajo que usa el mismo tipo de información.

La DTD es una definición de un documento XML, que especifica restricciones en la estructura y sintaxis del mismo. La DTD se puede incluir dentro del archivo del documento, pero normalmente se almacena en un fichero ASCII de texto separado. La

sintaxis de las DTD para SGML y XML es similar pero no idéntica. Una DTD es un documento que define la estructura de un documento XML: los elementos, atributos, entidades, notaciones, etc, que pueden aparecer, el orden y el número de veces que pueden aparecer, cuáles pueden ser hijos de cuáles, etc. El procesador XML utiliza la DTD para verificar si un documento es válido, es decir, si el documento cumple las reglas del DTD.

La definición de una DTD especifica la sintaxis de una aplicación de SGML o XML, que puede ser un estándar ampliamente utilizado como XHTML o una aplicación local.

Las DTD se emplean generalmente para determinar la estructura de un documento mediante etiquetas (en inglés tags) XML o SGML. Una DTD describe:

- Elementos: indican qué etiquetas son permitidas y el contenido de dichas etiquetas.
- Estructura: indica el orden en que van las etiquetas en el documento.
- Anidamiento: indica qué etiquetas van dentro de otras.

La DTD que debe utilizar el procesador XML para validar el documento XML se indica mediante la etiqueta DOCTYPE. La DTD puede estar incluida en el propio documento, ser un documento externo o combinarse ambas.

La DTD puede incluirse en el propio documento, con la siguiente sintaxis:

```
<!DOCTYPE nombre [
 ... declaraciones ...
]>
```

La DTD puede estar en un documento externo y, si sólo va a ser utilizada por una única aplicación, la sintaxis es la siguiente:

```
<!DOCTYPE nombre SYSTEM "uri">
```

Se puede combinar una DTD externa con una DTD interna, con la siguiente sintaxis:

```
<!DOCTYPE nombre SYSTEM "uri" [
 ... declaraciones ...
]>
```

La DTD puede estar en un documento externo y, si va a ser utilizada por varias aplicaciones, la sintaxis es la siguiente:

```
<!DOCTYPE nombre PUBLIC "fpi" "uri">
```

Para cada uno de los casos previamente definidos se debe tener en cuenta:

- "nombre" es el nombre del tipo de documento XML, que debe coincidir con el nombre del elemento raíz del documento XML.
- "uri" es el camino (absoluto o relativo) hasta la DTD.
- "fpi" es un indentificador público formal (Formal Public Identifier).

Los DTD describen la estructura de los documentos XML mediante declaraciones, las cuales son:

- **Declaraciones de entidades**
- **Declaraciones de notaciones**
- **Declaraciones de elementos**, que indican los elementos permitidos en un documento y su contenido (que puede ser simplemente texto u otros elementos).
- **Declaraciones de atributos**, que indican los atributos permitidos en cada elemento y el tipo o valores permitidos de cada elemento.

EJEMPLO DTD SIMPLE:

```
<!ELEMENT lista_de_personas (persona*)>
<!ELEMENT persona (nombre, fechanacimiento?, sexo?, numeroseguridadsocial?)>
<!ELEMENT nombre (#PCDATA) >
<!ELEMENT fechanacimiento (#PCDATA) >
<!ELEMENT sexo (#PCDATA) >
<!ELEMENT numeroseguridadsocial (#PCDATA)>
```

Observándolo línea a línea nos dice:

- <lista_de_personas> es un nombre de elemento válido. El * indica que puede haber 0 o más elementos de persona.

- <persona> es un nombre de elemento válido. Éste contiene obligatoriamente el elemento nombre mientras que el resto son opcionales. Y lo son porque nos lo indica el símbolo "?".
- <nombre> es un nombre de elemento válido. Contiene caracteres.
- <fechanacimiento> es un nombre de elemento válido.
- <sexo> es un nombre de elemento válido. Contiene caracteres.
- <numeroseguridadsocial> es un nombre de elemento válido.

EJEMPLO FICHERO CON DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE lista_de_personas SYSTEM "ejemplo.dtd">
<lista_de_personas>
 <persona>
  <nombre>José García</nombre>
  <fechanacimiento>25/04/1984</fechanacimiento>
  <sexo>Varón</sexo>
 </persona>
</lista_de_personas>
```

DIFERENCIAS ENTRE DTD Y XML SCHEMA

DTD:

- No Usan sintaxis XML.
- No permiten especificar tipos de datos.
- No son extensibles (No permite crear nuevos elementos).
- Posee un lenguaje propio de escritura lo que ocasiona problemas a la hora del aprendizaje, pues no sólo hay que aprender XML, sino también conocer el lenguaje de las DTDs.
- Para el procesado del documento, las herramientas y analizadores (parsers) empleados para tratar los documentos XML deben ser capaces de procesar también las DTDs .

- No permite el uso de namespaces y estos son muy útiles ya que permiten definir elementos con igual nombre dentro del mismo contexto, siempre y cuando se anteponga un prefijo al nombre del elemento.
- Tiene una tipología para los datos del documento extremadamente limitada, pues no permite definir el que un elemento pueda ser de un tipo número, fecha, etc. sino que sólo presenta variaciones limitadas sobre cadenas.
- El mecanismo de extensión es complejo y frágil ya que está basado en sustituciones sobre cadenas y no hace explicitas las relaciones, es decir, que dos elemento que tienen definido el mismo modelo de contenido no presentan ninguna relación.

XML SCHEMA:

- Usan Sintaxis de XML.
- Permiten especificar tipos de datos.
- Son extensibles. (Permite crear nuevos elementos).
- XML Schema presenta una estructura de tipos mucho más rica. En la segunda parte de la especificación de XML Schema (XML Schema Part 2: Datatypes) se definen los tipos base que se pueden emplear dentro de esquema de XML, como ejemplo podemos destacar: byte, integer, bolean, string, date, sequence, etc. Este sistema de tipos es muy adecuado para importar y exportar sistemas de bases de datos y, sobre todo, distingue los requerimientos relacionados con la representación léxica de los datos y el conjunto de información dominante y subyacente.
- Permite tipos definidos por el usuario, llamados Arquetipos. Dando un nombre a estos arquetipos, se pueden usar en distintas partes dentro del Schema.
- Es posible agrupar atributos, haciendo más comprensible el uso de un grupo de aspectos de varios elementos distintos, pero con un denominador común, que deben ir juntos en cada uno de estos elementos.

- Es posible trabajar con espacios de nombre, según la Especificación XML Schema Part 0: Primer, permitiendo validar documentos con varios namespaces.
- Con XML Schema es posible extender Arquetipos de un modo específico, es decir permite lo que en términos de orientación a objetos se llama herencia. Considérese un esquema que extiende otro previamente hecho, se permite refinar la especificación de algún tipo de elemento para, por ejemplo, indicar que puede contener algún nuevo elemento del tipo que sea; pero dejando el resto del esquema antiguo completamente intacto.

COMPARACIÓN SINTAXIS

DTD	XML Schema
ELEMENT	<element>
#PCDATA	Soportado como parte de un tipo simple
ANY	<any>
EMPTY	Soportado
Modelo de Contenido	<complexType>
, (Conector de secuencia)	<sequence>
| (Conector de alternativas)	<disjunction>
? (Opcional)	Soportado
+(Requerido y Repetible)	Soportado
*(Opcional y Repetible)	Soportado
ATTLIST	<attributeGroup>
Tipo de atributo CDATA, ID, IDREF, NOTATION...	Tipos <simpleType>predefinidos
ENTITY	NO Soportado
ENTITY%Parameter	NO Soportada