<?xml version=”1.0”?>

<documento>

<título>Tema 5. XML</título>

<ciclo>DAM</ciclo>

<fecha>

<día>1</día>

<mes>Septiembre</mes>

<año>2012</año>

</fecha>

</documento>

DOCUMENTO XML BIEN FORMADO

Con el validador de w3 – mira si el documento .xml está a tope (formado y validado).

<https://validator.w3.org/>

Los valores de los atributos van entrecomillados (dos dobles o dos simples). Los atributos solo se ponen en la etiqueta de apertura.

<nombre sexo="Hombre">Antonio</nombre>

<nombre sexo='Mujer'>Sara</nombre>

<nombre sexo='Mujer">Eva</nombre> TA feo esto

Todos los atributos tienen que tener un valor asociado. Por ejemplo hidden vale en HTML pero aquí sería hidden=”hidden”.

Para que lo pille bien formado tiene que tener las movidas de un DTD u otro método de validación.

Varias formas de validar

* DTD
* XMLSchema
* Relax NG

**Esquema (con xmlschema):**

Un esquema tiene extensión tipo xsd. Funciona como otro archivo xml.

**Etiquetas:**

Schema: elemento raíz: siguiendo una sintaxis determinada, en este caso:

<schema xmlns="http://www.w3.org/2001/XMLSchema">

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

**Referencia a un esquema:**

Se hace en el documento raíz de esta manera:

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

<documento xmlns=http://www.juanperez.com/doc

xmlns:xs=http://w3.org/2001/XMLSchema-instance

xs:schemaLocation="esquema.xsd">

texto xd…

</documento>

**Partes de un esquema (todo precedido de xs:):**

No aparecen atributos en el esquema si no tiene atributos el xml (atribute).

Los elementos se ponen con la etiqueta element.

Para poner los tipos que se almacenan simpleType.

Para poner elementos que están compuestos por más elementos se pone complexType.

Para poner varios elementos ‘agrupados’ se pone sequence.

Otras más rarunas que iremos viendo son: xs:annotation, xs:documentation y xs:appInfo.

A los elementos se le pueden poner más cosas:

<xs:element

name="nombre del elemento"

type="tipo global de datos"

ref="declaración del elemento global"

id="identificador"

form="cualificación" <!--qualified o unqualified -->

minOccurs="número mínimo de veces"

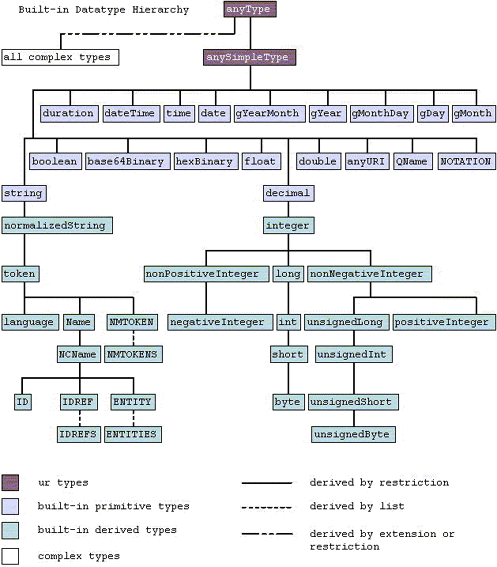
maxOccurs="máximo número de veces"

default="valor por defecto"

fixed="valor fijo"

>

**Diagrama de tipos:**



Podemos hacer nuestros propios tipos de datos por ejemplo fechaType, aunque luego hay que declararlo.

**Datos primitivos:**

Para textos string.

Para lógico boolean.

Para números int, float y double.

Para fechas: duration pero no muy útil xD (pasando de interpretar), datetime (se hace un fechaType y a casa).

**Tipos de datos personales:**

La sintaxis general es:

<xs:simpleType name="nombre">

...definición del tipo....

</xs:simpleType>

Creados por unión:

Se crea un dato formado por la unión de otros dos datos.

Sintaxis ejemplo:

<xs:simpleType name="gMonthC">

<xs:union memberTypes="xs:gMonth xs:gMonthDay"/>

</xs:simpleType>

Creados por restricción:

Se crea con la etiqueta restriction para ver qué datos no poner.

Sintaxis ejemplo:

<xs:simpleType name="nombre">

<xs:restriction base="tipo">

...definición de la restricción....

<xs:restriction>

</xs:simpleType>

Hay bien de tipos de datos por restricción así que achantas.

* **Tamaños de texto.** 
  + minLength. Indica el mínimo número de caracteres. Eso lo hace mediante el atributo value, en el que se indica un número con el tamaño mínimo.
  + maxLength. Indica un tamaño máximo de caracteres o de dígitos numéricos. Usa el mismo atributo value.
  + length. Indica un tamaño fijo de caracteres para el tipo. Es decir si indicados length con value="9" el texto deberá tener exactamente nueve caracteres.

Ejemplo:

<xs:simpleType name="nombresTipo">

<xs:restriction base="xs:normalizedString">

<xs:maxLength value="15" />

<xs:minLength value="4" />

</xs:restriction>

</xs:simpleType>

<xs:element name="título" type="doc:nombresTipo" />

* **Dígitos máximos.**
  + totalDigits. Número máximo de dígitos del número, incluyendo los decimales. El atributo **value** indica el número máximo deseado.
  + fractionDigits. Máximo número de decimales que puede tener el número.

Ejemplo:

<xs:simpleType name="tipo1">

<xs:restriction base="xs:decimal">

<xs:totalDigits value="6" />

<xs:fractionDigits value="2" />

</xs:restriction>

</xs:simpleType>

<xs:element name="partida" type="tipo1"/>

* **Máximos y mínimos numéricos.**
  + minExclusive. Establece un valor mínimo (no incluido) utilizando el atributo **value**.
  + maxExclusive. Establece un valor máximo (no incluido) utilizando el atributo **value**.
  + minInclusive. Establece un valor mínimo (incluido) utilizando el atributo **value**.
  + maxInclusive. Establece un valor máximo (incluido) utilizando el atributo **value**.

Ejemplo:

<xsd:simpleType name="strictlyPositive">

<xsd:restriction base="xsd:double">

<xsd:minExclusive value="0" />

</xsd:restriction>

</xsd:simpleType>

* **Espacios en blanco.**

Sirve para indicar la política de manejo de espacios en blanco en los textos. La etiqueta que lo controla es **whiteSpace** y tiene tres posibles valores para el atributo **value**:

* + **preserve**. No modificar espacios en blanco, ni tabuladores ni saltos de línea. Es decir se les tendrá en cuenta.
  + **replace**. Cada doble espacio o tabulador o salto de línea se cambia por un espacio (al estilo del tipo predefinido normalizedString).
  + **collapse**. Como el anterior, pero además elimina los espacios a izquierda y derecha. Es muy útil para usar en combinación con las propiedades de tamaño de texto vistas anteriormente.

Ejemplo (archivo prueba.xsd):

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:simpleType name="tipo1">

<xs:restriction base="xs:string">

<xs:whiteSpace value="collapse"/>

</xs:restriction>

</xs:simpleType>

<xs:element name="prueba" type="tipo1"/>

</xs:schema>

* **Enumeraciones:**

Limitan el contenido a una lista de valores. Se indican los posibles valores con una sucesión de etiquetas **enumeration**.

Ejemplo:

<xs:simpleType name="sexoTipo">

<xs:restriction base="xs:NCName">

<xs:enumeration value="Hombre"/>

<xs:enumeration value="Mujer"/>

</xs:restriction>

</xs:simpleType>

* **Plantillas (pattern)**

Permite establecer **expresiones regulares**, es decir, un texto con símbolos especiales que permiten establecer expresiones que ha de cumplir el contenido. Las expresiones regulares son bien conocidas por casi todos los programadores y dan una potencia increíble para establecer restricciones de texto avanzadas, lo que las hace muy utilizadas.

Las plantillas se manejan con etiquetas **pattern** a las que, en el atributo **value**, se indica un texto con símbolos especiales que especifica la expresión a cumplir. Los símbolos que se pueden utilizar son:

|  |  |
| --- | --- |
| Símbolo | Significado |
| **texto tal cual** | Hace que sólo se pueda escribir ese texto. |
| **[xyz]** | Permite elegir entre los caracteres x, y o z. |
| **[^xyz]** | Prohíbe usar cualquiera de los caracteres. |
| **[a-z]** | Vale cualquier carácter de la a a la z. |
| **^** | Inicio de línea |
| **$** | Final de línea |
| **+** | Repite acepta el carácter precedente una o más veces |
| **?** | Acepta el carácter precedente 0 o más veces |
| **\*** | Acepta el carácter precedente una o más veces |
| **{n}** | Acepta exactamente n repeticiones del carácter precedente. |
| **{n,}** | Acepta al menos n repeticiones del carácter precedente. |
| **{m,n}** | Acepta entre m y n repeticiones del carácter precedente. |
| **\s** | Permite indicar los caracteres especiales. Por ejemplo \^ representa el carácter circunflejo ^ como texto y no como código especial. |

Por ejemplo la validación para un dato tipo DNI (8 cifras y un número), sería:

<xs:simpleType name="dniTipo">

<xs:restriction base="xs:string">

<xs:whiteSpace value="collapse" />

<xs:pattern value="[0-9]{8}[A-Z]" />

</xs:restriction>

</xs:simpleType>

* **Por lista.**

Las listas permiten que un componente tenga como contenido una determinada lista de valores.

Para construir una lista, hay que seguir dos pasos:

1. Crear un tipo simple de datos cuyo contenido es una etiqueta **list**, la cual posee el atributo **itemType** para indicar el tipo de elementos de la lista.
2. Crear el tipo ya definitivo de datos que contendrá una etiqueta **restriction** a la cual como tipo base se indica el tipo simple de datos relacionado con la lista. En la restricción se puede indicar en el atributo **value** las siguientes etiquetas:

* **length**. Indica que la lista tendrá un número exacto de valores.
* **minLength**. Indica que la lista tendrá un número mínimo de valores.
* **maxLength**. Indica que la lista tendrá un número máximo de valores.
* **enumeration**. Posibles valores que puede tener la lista.
* **whiteSpace**. Gestión de los espacios en blanco en cada elemento de la lista.
* **pattern**. Expresión regular que debe cumplir cada elemento de la lista.

Es decir, son las etiquetas ya conocidas, pero que ahora se refieren a la lista.

Ejemplo:

<xs:simpleType name="listaDecimales">

<xs:list itemType="xs:decimal" />

</xs:simpleType>

<xs:simpleType name="listaNotas">

<xs:restriction base="listaDecimales">

<xs:minLength value="3"/>

<xs:maxLength value="6"/>

</xs:restriction>

</xs:simpleType>

Primero se define un tipo de lista (**listaDecimales**) simplemente indicando que será una lista cuyos valores serán números decimales. Después se concreta la lista (**listaNotas**), de modo que ahora se indica que la lista constará de tres a seis números (en el XML cada número irá separado por espacios).