Lenguaje de Marcas y Sistemas de Gestión de Información.

UT 04.03- Validación de Documentos XML.
XML Schema.

UT 04.03- Validación de Documentos XML. XML Schema.
1.- Introducción.
1.1.- Definición Ventajas de usar XSD.

- En este tema veremos otro mecanismo para validar las especificaciones de un documento XML: el **XML Schema**.
- Los XML Schemas se diferencian de los DTD en los siguientes aspectos:
  - Emplean la sintaxis propia de XML, y por tanto pueden ser analizados sintácticamente para comprobar si estaán bien formados.
  - Pueden ser manipulados por otro documento XML, y pueden utilizarse herramientas XML para elaborarlos y tratarlos.
  - Son más potentes que las DTD, es posible describir con mucho más detalle un documento: definir tipos de datos, especificar exactamente el número de ocurrencias de un elemento, etc.
- El lenguaje usado para definir los XML Schemas es **XSD (XML Schema Definition)**.

UT 04.02- Validación de Documentos XML. DTD.
1.- Introducción.
1.2.- Estructura de un documento XSD (I).

- Un documento XSD puede tener la forma siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="libro">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="titulo" type="xsd:string"/>
        <xsd:element name="autor" type="xsd:string" maxOccurs="5"/>
        <xsd:element name="editorial" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="precio" type="xsd:double"/>
     </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

- En su estructura podemos distinguir una serie de partes:
  - Declaración XSD.
  - Declaración de elementos.

UT 04.02- Validación de Documentos XML. DTD.
1.- Introducción.
1.2.- Estructura de un documento XSD (II).

Declaración XSD.

- En XSD el elemento raiz es siempre el elemento **schema**. Por tanto, la última línea del esquema es la etiqueta de cierre de dicho elemento.
- El elemento schema incluye el atributo **xmlns** (XML NameSpace), que especifica el espacio de nombres para el esquema. El valor de este atributo es la URI que identifica el vocabulario (XML vocabulary) al que se ajusta el documento, en este caso el vocabulario XML Schema.
- En el ejemplo, el formato **xmlns:xsd** indica que todos los elementos o atributos que lleven el prefijo “xsd:” pertenecen al espacio de nombres especificado en la URI:

  http://www.w3.org/2001/XMLSchema

  Se puede utilizar cualquier prefijo, siempre que se especifique el espacio de nombres XML asociado.
- Si el esquema referencia un único espacio de nombres, por ejemplo, si sólo utiliza elementos y atributos definidos en la especificación XML Schema, no es obligatorio usar el prefijo.

UT 04.02- Validación de Documentos XML. DTD.
1.- Introducción.
1.2.- Estructura de un documento XSD (III).

- Un posible fichero que puede validar este ejemplo es el siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<libro
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
  xsd:noNamespaceSchemaLocation="libro.xsd" precio="20">

  <Título>Fundamentos de XML Schema</Título>
  <Autores>Allen Wyke</Autores>
  <Autores>Andrew Watt</Autores>
  <Editorial>Wiley</Editorial>
</Libro>
```

- Como se puede ver en el elemento raíz se incluye una serie de definiciones:
  - Se especifica que se va a usar la especificar de XML Schema del año 2.001.
  - Se especifica que el espacio de nombres viene indicado en el fichero **libro.xsd**.

UT 04.02- Validación de Documentos XML. DTD.
2.- Definición de Elementos.
2.1.- La etiqueta <xsd:element>. Atributos.

- Como vimos en los DTD, todos los elementos que se vayan a usar en el ejemplar XML tienen que declararse en el esquema.
- La declaraciones de un elemento tiene la siguiente sintaxis:

```
<xsd:element
  name="nombre" type="tipo"
  minOccurs="valor" maxOccurs="valor"
  fixed=”valor" default=”valor”>
</xsd:element>
```

- Cada uno de los atributos tiene la siguiente función:
  - **name**. Nombre del elemento
  - **type**. Tipo de elemento.
  - **minOccurs** y **maxOccurs** (Opcional): Mínimo y máximo número de ocurrencias del elemento. Por defecto para ambos atributos es uno.

    Además, Si se quiere indicar que el elemento puede aparecer un número ilimitado de veces, el atributo maxOccurs tomará el valor **unbounded**.
  - **fixed** (Opcional). Especifica un **valor fijo** para el elemento.
  - **default** (Opcional). Especifica un **valor por defecto** para el elemento.

UT 04.02- Validación de Documentos XML. DTD.
2.- Definición de Elementos.
2.2.- La etiqueta <xsd:element>. Tipos de datos.

Tipos simples.

- Son elementos que **sólo pueden contener datos carácter; no pueden incluir otros elementos ni tampoco atributos**.
- Ejemplo:

```
<xsd:element name="fecha" type="xsd:date"/>
```

  En este caso se declara un elemento llamado fecha, de tipo de datos estandar de XSD **xsd:date**. Este tipo de datos es parte del vocabulario de XML Schema.

Tipos complejos.

- Estos elementos pueden incluir otros elementos además de atributos.
- El contenido de estos elementos se define incluyendo en su interior la etiqueta de inicio **<xsd:complexType>**.
- Ejemplo:

```
<xsd:element name="libro">
  <xsd:complexType>
    <xsd:attribute name="formato" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

  El elemento “libro” incluye un atributo “formato”, que es de tipo string.

UT 04.02- Validación de Documentos XML. DTD.
3.- Tipos primitivos de datos (I).

- El elemento **sólo puede contener datos carácter**. El tipo de dato será un tipo de dato **primitivo** o un tipo de dato **definido por el usuario**.
- Ejemplo.

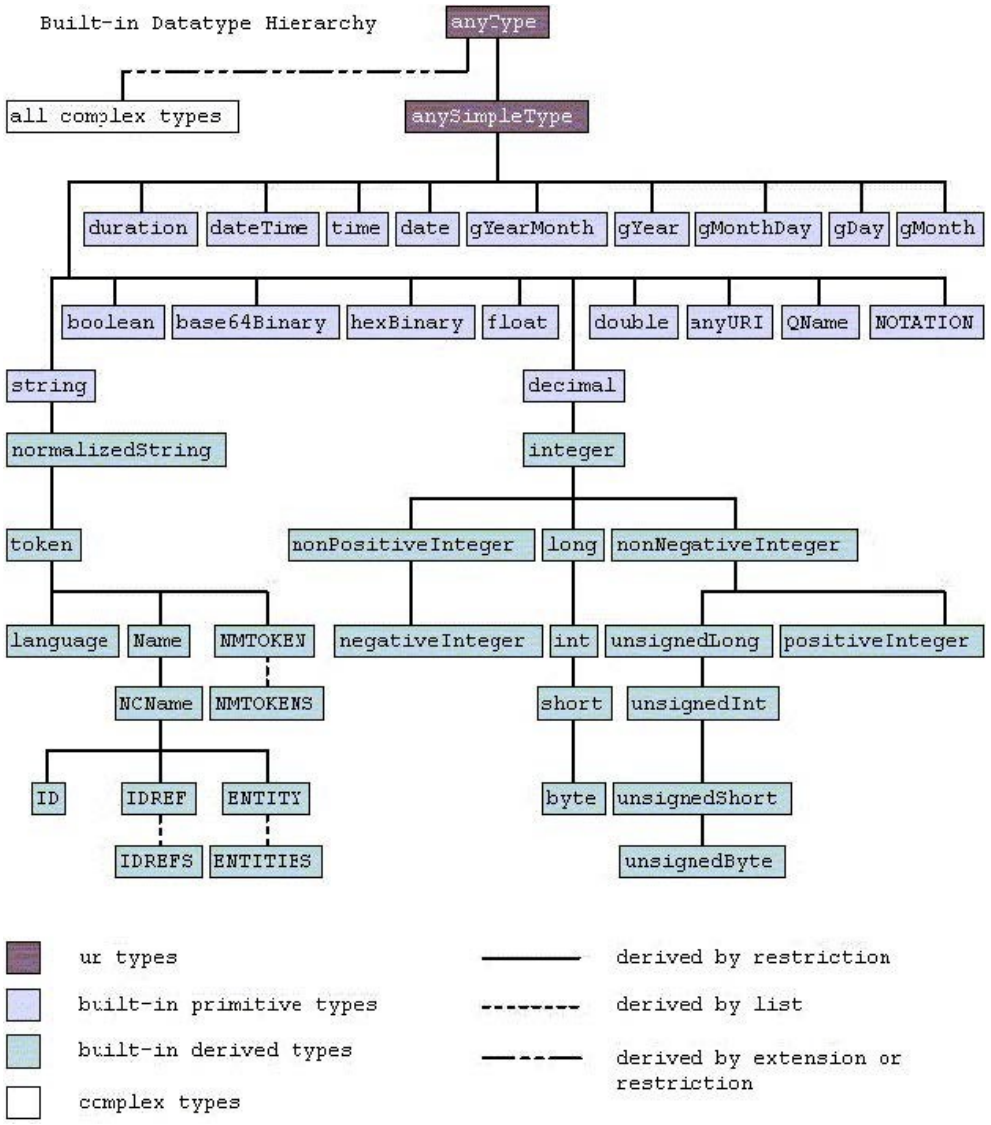
```
<xsd:element name="autor" type="xsd:string"/>
```

  En esta definición sólo contiene una cadena texto (el tipo string representa una secuencia de caracteres).
- Existen 19 tipos de datos simples predefinidos primitivos. En la sección W3Schools XSD Data podemos ver ejemplos de los distintos tipos de datos definidos. Un ejemplo de los mas usados son los siguientes:

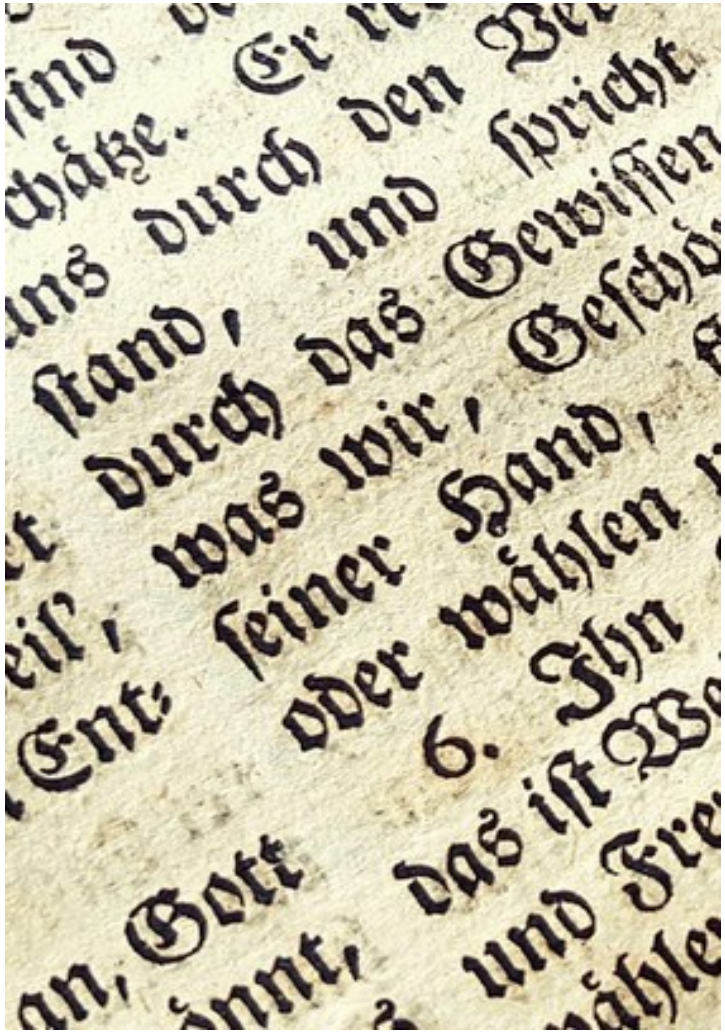
Tipo de dato	Ámbito
string	Texto con cualquier contenido excepto los símbolos no incluidos por XML.
boolean	true o false. También admite 1 (verdadero) ó 0 (falso).
integer	Números enteros.
decimal	Números decimales en coma fija.
time	Horas en el formato **hh:mm:ss**.
date	Fecha en formato **yyyy-mm-dd**.

UT 04.02- Validación de Documentos XML. DTD.
3.- Tipos primitivos de datos (II).

- En este gráfico podemos ver los tipos de datos primitivos definidos en el lenguaje.
- Podemos obtener tipos de datos derivados mediante el uso de las llamadas **facetas**.



UT 04.02- Validación de Documentos XML. DTD.
4.- Modelos de contenido para elementos.
4.1.- Modelo de texto.



- El elemento sólo puede contener datos carácter. Es decir, tiene **contenido de texto entre las marcas** que componen el elemento.
- **En este modelo no se permite que el elemento tenga atributos**.
- Ejemplo:

```
<xsd:element name="autor" type="xsd:string"/>
```

UT 04.02- Validación de Documentos XML. DTD.
4.- Modelos de contenido para elementos.
4.2.- Modelo de elemento vacío.

- **El elemento no puede contener datos carácter ni otros subelementos, pero sí puede incluir atributos**.
- **En este caso hay que declararlos como tipos complejos**.
- **Ejemplo**: el elemento antigüedad no tiene contenido. La información la alamacena mediante el atributo anyos que viene definido como un entero positivo.

```
<xsd:element name="antiguedad">
  <xsd:complexType>
    <xsd:attribute name="anyos" type="xsd:positiveInteger"/>
  </xsd:complexType>
</xsd:element>
```

UT 04.02- Validación de Documentos XML. DTD.
4.- Modelos de contenido para elementos.
4.3.- Elementos compuestos (I). Secuencia <xsd:sequence>.

- En este caso el elemento es el nodo raiz o es padre de otros elementos.
- En XML Schema existen tres formas de especificar las relaciones entre el elemento padre y sus hijos: **sequence**, **choice** y **all**.

<xsd:sequence>.

- Secuencia de elementos que tienen que aparecer en el documento XML.
- Deben aparecer todos, y en el mismo orden en que se especifican.
- Ejemplo:

```
<xsd:element name="vehiculo">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="propietario" type="xsd:string"/>
      <xsd:element name="matricula" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

UT 04.02- Validación de Documentos XML. DTD.
4.- Modelos de contenido para elementos.
4.3.- Elementos compuestos (II). Secuencia <xsd:choice>.

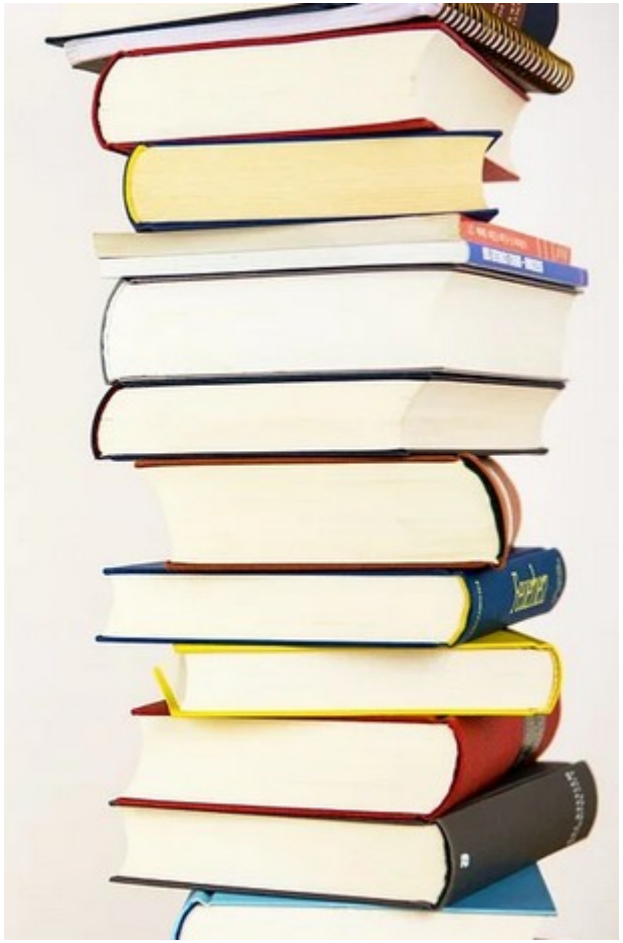
- Especifica una lista de elementos de los cuales **sólo puede aparecer uno de ellos en el elemento padre**.
- Ejemplo:

```
<xsd:element name="vehiculo">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="coche" type="xsd:string"/>
      <xsd:element name="moto" type="xsd:string"/>
      <xsd:element name="furgoneta" type="xsd:string"/>
      <xsd:element name="camion" type="xsd:string"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

En este caso se indica que un vehículo puede tener un elemento hijo de tipo coche, moto, furgoneta o camión.

- El elemento choice puede incluir opcionalmente los atributos **minOccurs** y **maxOccurs**, para especificar **el mínimo y máximo número de elementos hijos que pueden incluirse en el documento**.

UT 04.02- Validación de Documentos XML. DTD.
4.- Modelos de contenido para elementos.
4.3.- Elementos compuestos (III). Secuencia <xsd:all>.



- Se comporta igual que el elemento sequence, pero **es obligado que en el documento XML aparezcan todos los elementos especificados, pero no en el mismo orden**.
- Ejemplo:

```
<xsd:element name="camiseta">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="color" type="xsd:string"/>
      <xsd:element name="talla" type="xsd:string"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

En este caso el elemento camiseta puede tener

- un elemento color y un elemento talla.
- un elemento talla y un elemento color.

UT 04.02- Validación de Documentos XML. DTD.
4.- Modelos de contenido para elementos.
4.4.- Mixto.

- El elemento puede contener tanto datos carácter como elementos hijo.
- Los elementos hijo se definen igual que en el modelo anterior, mediante los elementos.
- Para indicar que el elemento puede además incluir datos carácter se usa el atributo “mixed” con valor igual al “true” en el elemento “complexType”.
- Ejemplo:

```
<xsd:element name="confirmacionPedido">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="intro" type="xsd:string"/>
      <xsd:element name="nombre" type="xsd:string"/>
      <xsd:element name="fecha" type="xsd:string"/>
      <xsd:element name="titulo" type="xsd:string"/>
    <xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<confimacion>
  <intro>Para:</intro>
  <nombre>Antonio Lara</nombre>
  Con fecha<fecha>24-01-2005</fecha> hemos recibido
  su pedido de <titulo>Raices</titulo>. Su título
  será enviado en 2 días hábiles desde la recepción
  del pedido.Gracias,
</confimacion>
```

UT 04.02- Validación de Documentos XML. DTD.
5.- Declaraciones de atributos (I).

- La sintaxis genérica de declaración de atributos es la siguiente:

```
<xsd:attribute name="nombreAtributo"
  type="tipoSimple"
  use="valor"
  default="valor"
  fixed="valor"/>
```

- **name**. Nombre del atributo.
- **type**. Tipo del atributo. Los atributos sólo pueden contener tipos simples.
- **use (opcional)**. Puede tomar uno de los siguientes valores:
  - **required** . Debe aparecer en el documento XML.
  - **optional**. Puede aparecer o no hacerlo. Es el valor por defecto.
  - **prohibited. El atributo no puede aparecer en el documento XML.**
- **default (opcional)**. Si no aparece en el documento XML, se le asigna el valor especificado.

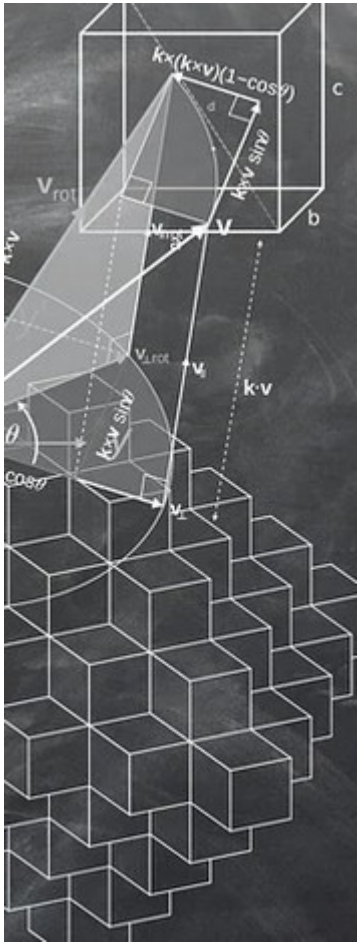
  Los valores por defecto sólo tienen sentido si el atributo es opcional, de lo contrario tendremos un error.
- **fixed (opcional)**. Define un valor fijo para el atributo. Si el valor del atributo está presente en la instancia del documento XML, el valor debe ser el mismo que el que indica el atributo “fixed”. En el caso de que el atributo no está presente en el documento XML, se le asigna el valor contenido en el atributo.

Los valores de los atributos “default” y “fixed” son mutuamente excluyentes.

UT 04.02- Validación de Documentos XML. DTD.
5.- Declaraciones de atributos (II).

- Sólo los elementos de tipo compuesto pueden contener atributos.
- Las declaraciones de atributos para un elemento **deben aparecer siempre al final del bloque delimitado por la etiqueta de inicio complexType**, después de las especificaciones de todos los demás componentes.

```
<xsd:element name="cliente">
  <xsd:complexType>
    <xsd:sequence>
      <xsd element name="apellido" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="numCliente"type="xsd:positiveInteger"/>
  </xsd:complexType>
</xsd:element>
```



UT 04.02- Validación de Documentos XML. DTD.
6.- Definiciones de datos por parte del usuario.
6.1.- Referencias a otros elementos.

- Es posible declarar elementos de forma global y luego hacer referencias a ellos desde otros elementos.
- Ventajas:
  - **Permite reutilizar una misma definición de un elemento en varios sitios del Schema.**
  - **Puede ayudar a que los schema estén mejor estructurados y sean más fácilmente legibles.**
- Para referenciar a un elemento se utiliza el atributo **ref** cuyo valor es el nombre del elemento referenciado, en lugar del atributo name seguido de la definición del elemento.
- En el siguiente ejemplo se utiliza el elemento nombre referenciado en dos tipos de datos compuestos definidos por el usuario.

```
<xsd:complexType name="tipoAlumno">
  <xsd:sequence>
    <xsd:element ref="nombre"/>
    <xsd:element name="edad" type="xsd:positiveInteger"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="tipoTutor">
  <xsd:sequence>
    <xsd:element ref="nombre"/>
    <xsd:element name="especialidad" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="nombre" type="xsd:string"/>
```

UT 04.02- Validación de Documentos XML. DTD.
6.- Definiciones de datos por parte del usuario.
6.2.- Tipos de datos con nombre.

- La forma más sencilla de crear un nuevo tipo es crear un elemento complexType al que se le asigna un nombre mediante el atributo **name**.
- Ejemplo:

```
<xsd:complexType name="tipoAlumno">
  <xsd:sequence>
    <xsd:element ref="nombre"/>
    <xsd:element ref="edad"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="nombre" type="xsd:string"/>
<xsd:element name="edad" type="xsd:positiveInteger"/>
```

Con esto creamos un tipo nuevo llamado **tipoAlumno** que está formado por una secuencia de dos elementos: nombre y edad.

- Podremos utilizar este tipo que hemos creado para definir elementos, por ejemplo:

```
<xsd:element name="precio" type="precioInfo"/>
```

- En la definición de tipoAlumno el nombre del tipo no lleva el prefijo “xsd”, porque no pertenece al espacio de nombres del estándar XML Schema.
- La principal ventaja de definir tipos de datos propios es:por un lado, que estos tipos se
  - Pueden utilizar donde se quiera.
  - Se pueden utilizar como tipos base para definir otros tipos.

UT 04.02- Validación de Documentos XML. DTD.
6.- Definiciones de datos por parte del usuario.
6.2.- Restricciones <xsd:restriction> (I). Concepto.

- Muchas veces es necesario **restringuir los valores de un campo**. Para ello, a partir de un **tipo simple de datos** (cadenas, números, etc) se delimitan que valores puede tener. Ese mecanismo se conoce como **restricciones**.
- Con una restricción se añaden condiciones a alguno de los tipos predefinidos en el XML Schema. Esto se hace con el elemento **<xsd:restriction>**.
- Ejemplo:

```
<xsd:simpleType name="tipoEuro">
    <xsd:restriction base="xsd:decimal">
        <xsd:fractionDigits value="2"/>
    </xsd:restriction>
</xsd:simpleType>
```

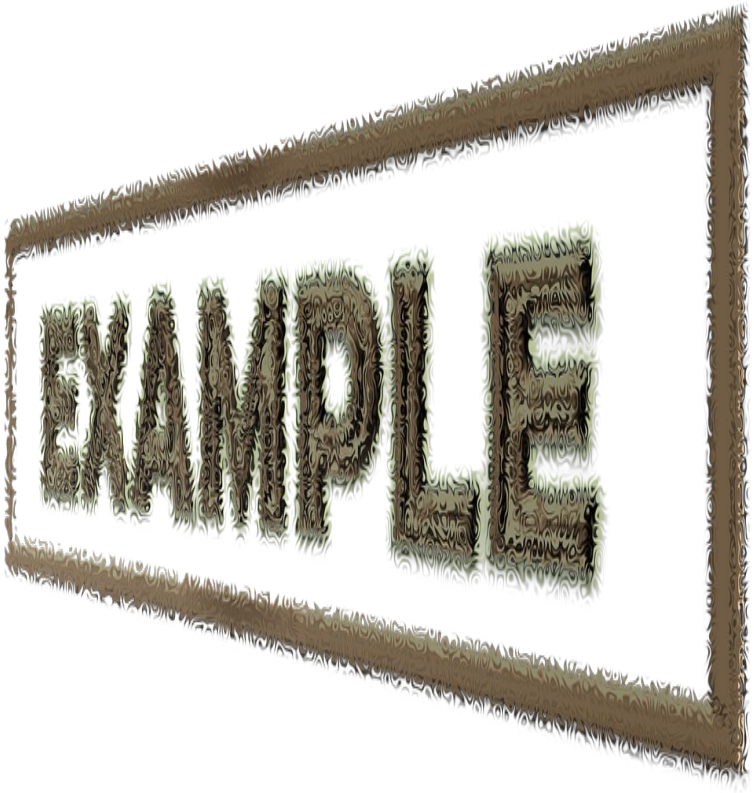
En este ejemplo, partiendo del tipo de datos xsd:decimal, que es primitivo de XML Schema, se obtiene el tipo **tipoEuro** que limita el **número de decimales a dos**.

UT 04.02- Validación de Documentos XML. DTD.
6.- Definiciones de datos por parte del usuario.
6.2.- Restricciones <xsd:restriction> (II). Facetas.

- XML Schema tiene definidas un conjunto de reglas de aplicación a la hora de hacer restricciones en los tipos de datos. Estas se conocen como **facetas**. En la siguiente tabla se muestra un listado de facetas disponibles en XML Schema.

Tipo de dato	Ámbito
minInclusive	mínimo valor que puede tomar el número.
minExclusive	el número debe ser mayor que este valor.
maxInclusive	máximo valor que puede tomar el número.
maxInclusive	el número debe ser menor que este valor.
totalDigits	total de cifras en el número, incluyendo las enteras y las decimales.
fractionDigits	número de cifras decimales.
length	número de unidades del valor literal. Para cada
minLength **maxLength**	valor mínimo y máximo respectivamente para la faceta length.tipo de dato se define de una forma.
pattern	formato que debe tener el valor, especificado mediante una expresión regular tradicional.
enumeration	conjunto de posibles valores que puede tomar el dato.
whiteSpace	controla la forma que tendrá el contenido de este dato una vez haya sido procesado.

UT 04.02- Validación de Documentos XML. DTD.
6.- Definiciones de datos por parte del usuario.
6.2.- Restricciones <xsd:restriction> (III). Ejemplos.



- En este ejemplo se crea el tipo **tipoEdad** como una restricción de valores enteros que van desde cero a ciento veinte.

```
<xs:simpleType name=”tipoEdad”>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
     <xs:maxInclusive value="120"/>
  </xs:restriction>
</xs:simpleType>
```

- En este ejemplo se crea el tipo **tipoNombre** como una restricción de valores a una cadena que tendrá una longitud máxima de treinta caracteres.

```
<xs:simpleType name=”tipoNombre”>
  <xs:restriction base="xs:string">
    <xs:maxlength value="30"/>
  </xs:restriction>
</xs:simpleType>
```

UT 04.02- Validación de Documentos XML. DTD.
6.- Definiciones de datos por parte del usuario.
6.3.- Tipos enumerados <xsd:enumeration>.

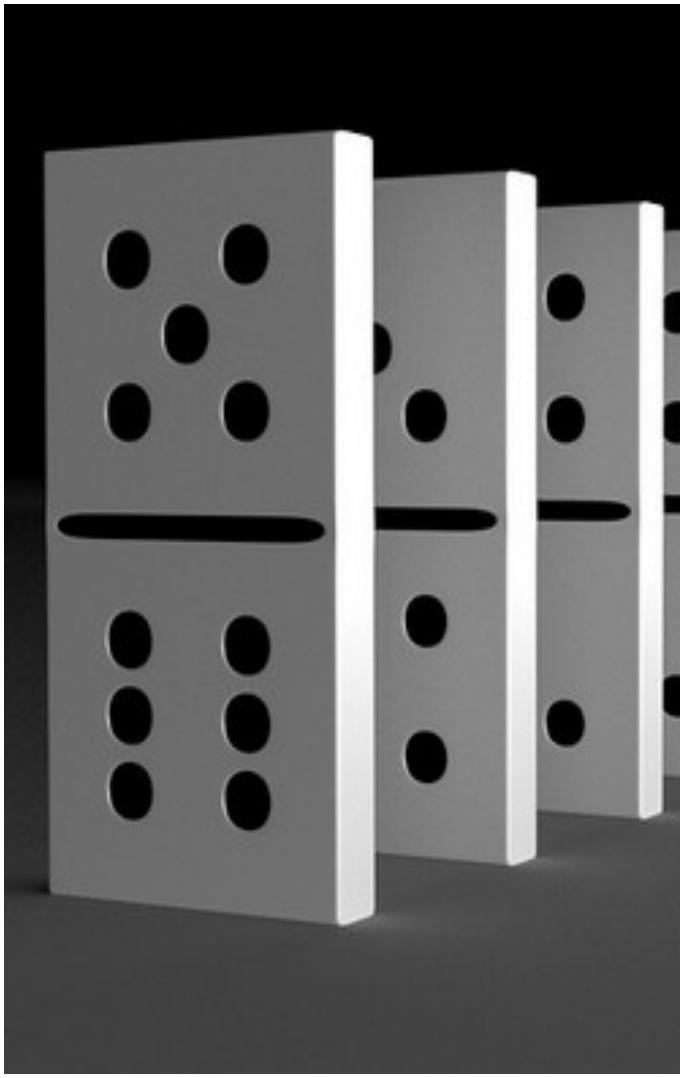
- Los tipos enumerados para limitar los valores que puede tomar un elemento o atributo.
- Ejemplo:

```
<xsd:simpleType name="tipoRiesgo">
  <xs:restriction base="xsd:string">
    <xs:enumeration value="bajo"/>
    <xs:enumeration value="medio"/>
    <xs:enumeration value="alto"/>
    <xs:enumeration value="extremo"/>
  </xs:restriction>
</xs:simpleType>
```

tipoRiesgo solo puede tener cuatro valores diferentes, dando un error de validación el valor no está entre lso posibles.

- Como siempre, a partir de el podemos definir otros tipos de datos donde se use esta definición

```
<xsd:complexType name="tipoAmenaza">
  <xsd:sequence>
    <xsd:element name="codigo" type="xs.string"/>
    <xsd:element name="nombre" type="xsd:string"/>
    <xsd:element name="riesgo" type="tipoRiesgo"/>
  </xsd:sequence>
</xsd:complexType>
```



UT 04.02- Validación de Documentos XML. DTD.
6.- Definiciones de datos por parte del usuario.
6.4.- Tipos listas <xsd:list>.

- Las listas **permiten incluir valores múltiples, separados mediante espacios**.
- Las listas permiten elegir el tipo de datos de los valores en la lista, mediante el atributo itemType. Además, se pueden aplicar otras facetas, como length, minlength, etc.
- Ejemplo:

```
<!-- Tipo el enumerado tipoSoporte -->
<xsd:simpleType name="tipoSoporte">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="libro"/>
    <xsd:enumeration value="eBook"/>
    <xsd:enumeration value="audioLibro"/>
  </xsd:restriction>
</xsd:simpleType>

<!-- Tipo lista tipoListadoSoporte -->
<xsd:simpleType name="tipoListaSoportes">
  <xsd:list itemType="tipoSoporte"/>
</xsd:simpleType>

<!-- Restringimos la lista a tres elementos →
<xsd:simpleType name="tipoListaTresSoportes">
  <xsd:restriction base="tipoListaSoportes">
    <xsd:maxLength value="3"/>
  </xsd:restriction>
</xsd:simpleType>
```

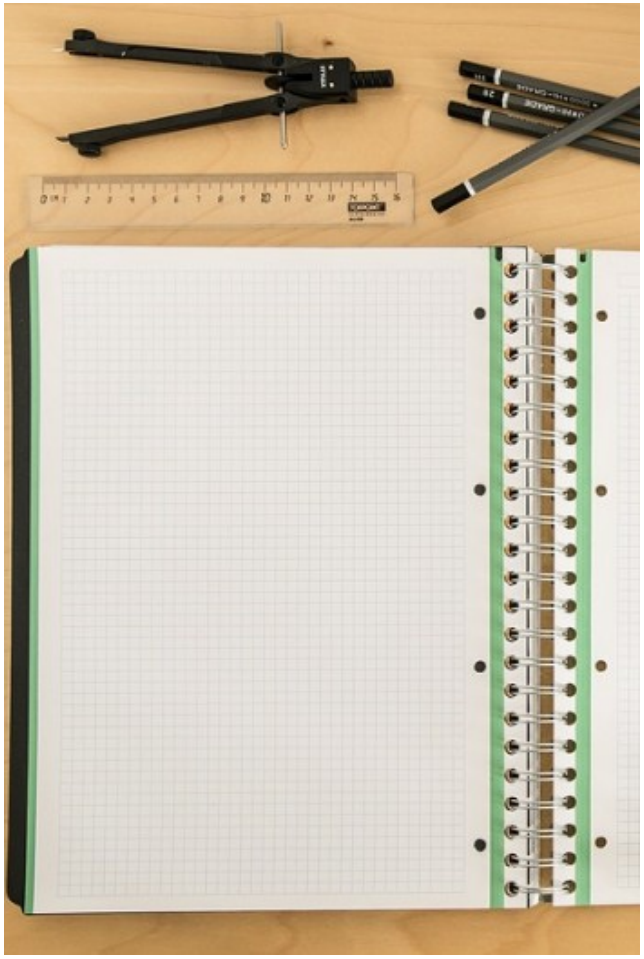
UT 04.02- Validación de Documentos XML. DTD.
6.- Definiciones de datos por parte del usuario.
6.5.- Definición de atributos por extensión.

- En el caso de que **un elemento simple pueda tener atributos, pero no elementos hijo** se puede definir mediante una **extensión**.
- Esto se hace utilizando el concepto **xsd:extension** dentro de un xsd:simpleContent.
- Ejemplo:

```
<xsd:element name="tipoReserva">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="confirmado" default="no"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

- En este ejemplo definimos el elemento **tipoReserva** de tipo complejo. Sin embargo, su contenido solo tiene datos carácter y un atributo.

  Esto se especifica definiendo el elemento xsd:simpleContent como una extensión del tipo base xsd:string a la que se le añade el atributo confirmado cuyo valor por defecto es **no**.



UT 04.02- Validación de Documentos XML. DTD.
7.- Espacios de nombres y XML Schemas.

- Al crear un XML Schema hacemos uso de los elementos y atributos especificados en el **estándar de XML Schema**. Por ello, debemos incluir en el elemento raíz del esquema una referencia al espacio de nombres

```
xmlns:xsd = “http://www.w3.org/2001/XMLSchema”
```

  Esto indica que los elementos y tipos de datos utilizados en el esquema provienen del espacio de nombres indicado y se le asigna el prefijo xsd.

- En el documento XML que se va a validar, se debe especificar donde la ruta del fichero donde está definido el XML Schema.

  Esto se hace mediante el comando **noNamespaceSchemaLocation**. Identifica un documento de XML Schema que no tiene un espacio de nombres de destino y lo asocia al documento XML instancia. Para ello, se introduce la linea

```
<raiz
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
  xsd:noNamespaceSchemaLocation="fichero.xsd">
```

UT 04.02- Validación de Documentos XML. DTD.
8.- Métodos de diseño de XML schema.

- Hay tres formas básicas de crear el diseño de un XML Schema. Estos son las siguientes:
  - Diseño plano.
  - Usar referencias a elementos y atributos.
  - Crear tipos con nombre.
- **Diseño plano**. Partiendo de un documento XML a validar se sigue la estructura del mismo definiendo los elementos que aparecen en el mismo de forma secuencial la definición completa de cada elemento en el mismo orden en el que aparecen en el documento instancia.

  Es un método de diseño sencillo, pero puede dar lugar a esquemas XML difíciles de leer y mantener cuando los documentos son complejos.
- **Usar referencias a elementos y atributos**. Primero se definen los diferentes elementos y atributos, para después referenciarlos utilizando el atributo **ref**.
- **Crear tipos con nombre**. Se definen clases o tipos utilizando los elementos de XML Schema con un nombre. Estos tipos definidos se pueden utilizar mediante el atributo **type** de los elementos.

  Este mecanismo permite reutilizar definiciones de tipos en diferentes puntos del documento.