# 1. Documento XML. Estructura y sintaxis

## 1.1. Declaración de tipo de documento

* <!DOCTYPE nombre\_ejemplar SYSTEM "URI">
  + En este caso, se especifica un URI donde pueden localizarse las declaraciones.
* <!DOCTYPE nombre\_ejemplar PUBLIC "id\_publico" "URI">
  + En este caso también se especifica un identificador, que puede ser utilizado por el procesador XML para intentar generar un URI alternativo, posiblemente basado en alguna tabla. Como se puede observar también es necesario incluir algún URI.

## 1.2. Definición de la sintaxis de documentos XML

Un elemento es un grupo formado por una etiqueta de apertura, otra de cierre y el contenido que hay entre ambas.

El dato será un elemento si cumple alguna de las siguientes condiciones:

* Contiene subestructuras.
* Es de un tamaño considerable.
* Su valor cambia frecuentemente.
* Su valor va a ser mostrado a un usuario o aplicación.

Los casos en los que el dato será un atributo son:

* El dato es de pequeño tamaño y su valor raramente cambia, aunque hay situaciones en las que este caso puede ser un elemento.
* El dato solo puede tener unos cuantos valores fijos.
* El dato guía el procesamiento XML pero no se va a mostrar.

Los espacios de nombres, o namespaces, ¿qué nos permiten?

* Diferenciar entre los elementos y atributos de distintos vocabularios con diferentes significados que comparten nombre.
* Agrupar todos los elementos y atributos relacionados de una aplicación XML para que el software pueda reconocerlos con facilidad.

¿Cómo se declaran?

* xmnls:"URI\_namespace"

¿Y si se usa un prefijo que nos informe sobre cuál es el vocabulario al que está asociada esa definición?

* xmnls:prefijo="URI\_namespace"

# 2. Definiciones de tipo de documento, DTD

Están formadas por una relación precisa de qué elementos pueden aparecer en un documento y dónde, así como el contenido y los atributos del mismo. Garantizan que los datos del documento XML cumplen las restricciones que se les haya impuesto en el DTD, ya que estas últimas permiten:

* Especificar la estructura del documento.
* Reflejar una restricción de integridad referencial mínima utilizando (ID e IDREF).
* Utilizar unos pequeños mecanismos de abstracción comparables a las macros, que son las entidades.
* Incluir documentos externos.

¿Cuáles son los inconvenientes de los DTD?

Los principales son:

* Su sintaxis no es XML.
* No soportan espacios de nombres.
* No definen tipos para los datos. Solo hay un tipo de elementos terminales, que son los datos textuales.
* No permite las secuencias no ordenadas.
* No es posible formar claves a partir de varios atributos o elementos.
* Una vez que se define un DTD no es posible añadir nuevos vocabularios.

Cuando están definidas dentro del documento XML se ubican entre corchetes después del nombre del ejemplar en el elemento <!DOCTYPE> pero, cuando está definido en un fichero externo ¿a qué tipo de fichero corresponde? Definimos el DTD externo en un fichero de texto plano con extensión dtd.

## 2.1. Declaraciones de tipos de elementos terminales

Los tipos terminales son aquellos elementos que se corresponden con hojas de la estructura de árbol formada por los datos del documento XML asociado al DTD. La declaración de tipos de elementos está formada por la cadena <!ELEMENT> separada por, al menos un espacio del nombre del elemento XML que se declara, y seguido de la declaración del contenido que puede tener dicho elemento.

En el caso de elementos terminales, es decir, aquellos que no contienen más elementos, esta declaración de contenido es dada por uno de los siguientes valores:

* EMPTY: Indica que el elemento no es contenedor. Por ejemplo, la siguiente definición muestra un elemento A que no contiene nada:

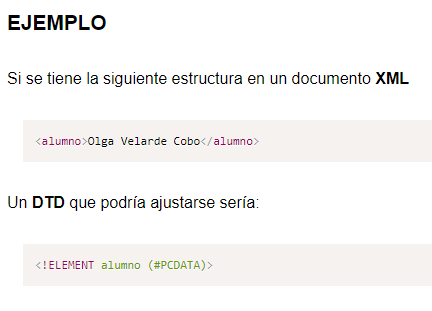
|  |
| --- |
| <!ELEMENT A EMPTY> |

* ANY: Permite que el contenido del elemento sea cualquier cosa. Un ejemplo de definición de un elemento de este tipo es:

|  |
| --- |
| <!ELEMENT A ANY> |

* (#PCDATA): Indica que los datos son analizados en busca de etiquetas, resultando que el elemento no puede contener elementos, es decir solo puede contener datos de tipo carácter exceptuando los siguientes: <, &, ]], >. Si es de este tipo, el elemento A tendrá una definición como:

|  |
| --- |
| <!ELEMENT A (#PCDATA)> |



## 2.2. Declaraciones de tipos de elementos no terminales

Una vez que sabemos el modo de definir las hojas de un árbol de datos veamos cómo definir sus ramas, es decir los elementos que están formados por otros elementos.

Para definirlos utilizamos referencias a los grupos que los componen tal y como muestra el ejemplo:

<!ELEMENT A (B, C)>

**Operador opción, ?.** Indica que el elemento no es obligatorio. En el siguiente ejemplo el subelemento trabajo es opcional.

<!ELEMENT telefono (trabajo?, casa )>

**Operador uno-o-más, +.** Define un componente presente al menos una vez. En el ejemplo definimos un elemento formado por el nombre de una provincia y otro grupo, que puede aparecer una o varias veces.

<!ELEMENT provincia (nombre, (cp, ciudad)+ )>

**Operador cero-o-mas, \*.** Define un componente presente cero, una o varias veces. En el ejemplo el grupo (cp, ciudad) puede no aparecer o hacerlo varias veces.

<!ELEMENT provincia (nombre, (cp, ciudad)\* )>

**Operador de elección, |.** Cuando se utiliza sustituyendo las comas en la declaración de grupos indica que para formar el documento XML hay que elegir entre los elementos separados por este operador. En el ejemplo siguiente, el documento XML tendrá elementos provincia que estarán formados por el elemento nombre y el cp (código postal), o por el elemento nombre y la ciudad.

<!ELEMENT provincia (nombre, (cp | ciudad) )>



## 2.3. Declaraciones de listas de atributos para los tipos de elementos

* Enumeración, es decir, el atributo solo puede tomar uno de los valores determinados dentro de un paréntesis y separados por el operador |.

**<!ATTLIST fecha dia\_semana (lunes|martes|miércoles|jueves|viernes|sábado|domingo) #REQUIRED>**

* **CDATA**, se utiliza cuando el atributo es una cadena de texto.
* **ID**, permite declarar un atributo identificador en un elemento. Hay que recordar que este valor ha de ser único en el documento. Además hay que tener en cuenta que los números no son nombres válidos en XML, por tanto no son un identificador legal de XML. Para resolverlo suele incluirse un prefijo en los valores y separarlo con un guión o una letra.
* **IDREF**, permite hacer referencias a identificadores. En este caso el valor del atributo ha de corresponder con el de un identificador de un elemento existente en el documento.
* **NMTOKEN**, permite determinar que el valor de un atributo ha de ser una sola palabra compuesta por los caracteres permitidos por XML.

¿También hemos de declarar si el valor de un atributo es obligatorio o no? Si, para ello se usan los siguientes modificadores:

* **#IMPLIED**, determina que el atributo sobre el que se aplica es opcional.
* **#REQUIRED**, determina que el atributo tiene carácter obligatorio.
* **#FIXED**, permite definir un valor fijo para un atributo independientemente de que ese atributo se defina explícitamente en una instancia del elemento en el documento XML.
* **Literal**, asigna a un atributo el valor dado por una cadena entre comillas.



## 2.4. Declaraciones de entidades

Para definir una entidad en un DTD se usa el elemento <!ENTITY>

Las entidades pueden ser de cuatro tipos:

* Internas: Existen cinco entidades predefinidas en el lenguaje, son:
  + &lt;: Se corresponde con el signo menor que, <.
  + &gt;: Hace referencia al signo mayor que, >.
  + &quot;: Son las comillas rectas dobles, ''.
  + &apos;: Es el apóstrofe o comilla simple, '.
  + &amp;: Es el et o ampersand, &.

¿Se puede definir una entidad diferente? ¿Cómo? Utilizando la siguiente sintaxis: <!ENTITY nombre\_entidad "valor de la entidad"> Por ejemplo:

<!ENTITY dtd "Definiciones de Tipo de Documento">

* Externas: Permiten establecer una relación entre el documento XML y otro documento a través de la URL de éste último. Un ejemplo de declaración de una entidad externa es:

<!ENTITY nombre\_entidad SYSTEM "http://localhost/docsxml/fichero\_entidad.xml">

* De parámetro: Permite dar nombres a partes de un DTD y hacer referencia a ellas a lo largo del mismo. Son especialmente útiles cuando varios elementos del DTD comparten listas de atributos o especificaciones de contenidos. Se denotan por %entidad;

<!ENTITY %direccion "calle, numero?, ciudad, cp">

<!ENTITY alumno (dni, %direccion;)>

<!ENTITY ies (nombre, %direccion;)>

* De parámetro externas: Permite incluir en un DTD elementos externos, lo que se aplica en dividir la definición DTD en varios documentos.

<!ENTITY persona SYSTEM "persona.dtd">

## 2.5. Declaraciones de notación

<!NOTATION nombre SYSTEM aplicacion>

<!NOTATION gif SYSTEM "gifEditor.exe">

<!ENTITY dibujo SYSTEM "imagen.gif" NDATA gif>

## 2.6. Secciones condicionales

Permiten incluir o ignorar partes de la declaración de un DTD. Para ello se usan dos tokens:

* INCLUDE, permite que se vea esa parte de la declaración del DTD. Su sintaxis es:

<![INCLUDE [Declaraciones visibles] ] >

<![INCLUDE [ <!ELEMENT nombre (#PCDATA)>] ] >

* IGNORE, permite ocultar esa sección de declaraciones dentro del DTD. La forma de uso es:

<![IGNORE [Declaraciones ocultas] ] >

<![IGNORE [<!ELEMENT clave (#PCDATA)>] ] >

# 3. XML Schema

Vamos a crear un esquema correspondiente al siguiente documento XML:

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

<!DOCTYPE alumno>

<alumno edad="22">Olga Velarde Cobo</alumno>

Un XML Schema que podría ajustarse sería:

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

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

<xs:element name="alumno" type="xs:string"/>

</xs:schema>

## 3.1. Tipos de datos

Son los distintos valores que puede tomar el atributo type cuando se declara un elemento o un atributo y representan el tipo de dato que tendrá el elemento o atributo asociado a ese type en el documento XML.

Algunos de estos valores predefinidos son:

* **string**, se corresponde con una cadena de caracteres UNICODE.
* **boolean**, representa valores lógicos, es decir que solo pueden tomar dos valores, true o false.
* **integer**, número entero positivo o negativo.
* **positiveInteger**, número entero positivo.
* **negativeInteger**, número entero negativo.
* **decimal**, número decimal, por ejemplo, 8,97.
* **dateTime**, representa una fecha y hora absolutas.
* **duration**, representa una duración de tiempo expresado en años, meses, días, horas, minutos segundos. El formato utilizado es: **PnYnMnDTnHnMnS**. Por ejemplo para representar una duración de 2 años, 4meses, 3 días, 5 horas, 6 minutos y 10 segundos habría que poner: P2Y4M3DT5H6M7S. Se pueden omitir los valores nulos, luego una duración de 2 años será P2Y. Para indicar una duración negativa se pone un signo – precediendo a la P.
* **time**, hora en el formato **hh:mm:ss**.
* date, fecha en formato CCYY-MM-DD.
* **gYearMonth**, representa un mes de un año determinado mediante el formato CCYY-MM.
* **gYear,** indica un año gregoriano, el formato usado es CCYY.
* **gMothDay**, representa un día de un mes mediante el formato –MM-DD.
* **gDay**, indica el ordinal del día del mes mediante el formato –DD, es decir el 4º día del mes será –04.
* **gMonth**, representa el mes mediante el formato –MM. Por ejemplo, febrero es –02.
* **anyURI**, representa una URI.
* **language**, representa los identificadores de lenguaje, sus valores están definidos en RFC 1766.
* **ID, IDREF, ENTITY, NOTATION, MTOKEN**. Representan lo mismo que en los DTD

### 3.1.1 Facetas de los tipos de datos

* **length, minlength, maxlentgh**: Longitud del tipo de datos.
* **enumeration**: Restringe a un determinado conjunto de valores.
* **whitespace**: Define el tratamiento de espacios (preserve/replace, collapse).
* **(max/min)(In/Ex)clusive**: Límites superiores/inferiores del tipo de datos. Cuando son Inclusive el valor que se determine es parte del conjunto de valores válidos para el dato, mientras que cuando se utiliza Exclusive, el valor dado no pertenece al conjunto de valores válidos.
* **totalDigits, fractionDigits:** número de dígitos totales y decimales de un número decimal.
* **pattern:** Permite construir máscaras que han de cumplir los datos de un elemento. La siguiente tabla muestra algunos de los caracteres que tienen un significado especial para la generación de las máscaras.

|  |  |
| --- | --- |
| **Elementos para hacer patrones.** | |
| **Patrón** | **Significado** |
| **[A-Z a-z]** | Letra. |
| **[A-Z]** | Letra mayúscula. |
| **[a-z]** | Letra minúscula. |
| **[0-9]** | Dígitos decimales. |
| **\D** | Cualquier carácter excepto un dígito decimal. |
| **(A)** | Cadena que coincide con A. |
| **A | B** | Cadena que es igual a la cadena A o a la B. |
| **AB** | Cadena que es la concatenación de las cadenas A y B. |
| **A?** | Cero o una vez la cadena A. |
| **A+** | Una o más veces la cadena A. |
| **A\*** | Cero o más veces la cadena A. |
| **[abcd]** | Alguno de los caracteres que están entre corchetes. |
| **[^abcd]** | Cualquier carácter que no esté entre corchetes. |
| **\t** | Tabulación. |

### 3.1.3. facetas: soluciones a los ejercicios

1.- Creación de una cadena de texto con una longitud máxima de 9 caracteres y dos valores posibles.

<xs:simpleType name="estado">

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

<xs:maxLength value="9"/>

<xs:enumeration value="conectado"/>

<xs:enumeration value="ocupado"/>

</xs:restriction>

</xs:simpleType>

2.- Creación de un elemento en el que se respetan los espacios tal y como se han introducido.0

<xs:simpleType name="nombre">

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

<xs:whitespace value="preserve"/>

</xs:restriction>

</xs:simpleType>

3.- Creación de un elemento calificaciones de dos dígitos cuyo valor es un número entero comprendido entre 1 y 10, ambos inclusive.

<xs:simpleType name="calificaciones">

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

<xs:totalDigits value="2"/>

<xs:minExclusive value="0"/>

<xs:maxInclusive value="10"/>

</xs:restriction>

</xs:simpleType>

4.- Creación de la máscara de un DNI mediante pattern.

<xs:simpleType name="dni">

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

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

</xs:restriction>

</xs:simpleType>

## 3.2. Elementos del lenguaje

A continuación se listan los más usados:

* Esquema, **xs:schema**, contiene la definición del esquema.
* Tipos complejos, **xs:complexType**, define tipos complejos.
* Tipos simples, **xs:simpleType**, permite definir un tipo simple restringiendo sus valores.
* Restricciones, **xs:restriction**, permite establecer una restricción sobre un elemento de tipo base.
* Agrupaciones, **xs:group**, permite nombrar agrupaciones de elementos y de atributos para hacer referencia a ellas.
* Secuencias, **xs:secuence**, permite construir elementos complejos mediante la enumeración de los que les forman.
* Alternativa, **xs:choice**, representa alternativas, hay que tener en cuenta que es una o-exclusiva.
* Contenido mixto, definido dando valor true al atributo mixed del elemento **xs:complexType**, permite mezclar texto con elementos.
* Secuencias no ordenadas, **xs:all**, representa a todos los elementos en cualquier orden.

### 3.2.1. Ejemplo de esquema

Define un esquema correspondiente a un documento XML para estructurar la información personal sobre los alumnos de un centro educativo.

<?xml version="1.0" encoding="ISO-8859-1"?>

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

<!-- elemento raíz -->

<xs:element name="alumnos" type="datosAlum"/>

<!-- Definicion del tipo datosAlum -->

<xs:complexType name="datosAlum">

<xs:sequence>

<xs:element name="alumno" type="datos" minOccurs="1" maxOccurs="unbounded"/>

</xs:sequence>

</xs:complexType>

<!-- Definicion del tipo datos -->

<xs:complexType name="datos">

<xs:sequence>

<xs:element name="nombre" type="xs:string" minOccurs="1" maxOccurs="1"/>

<xs:element name="apellidos" type="xs:string" minOccurs="1" maxOccurs="1"/>

<xs:element name="direccion" type="datosDireccion" minOccurs="1" maxOccurs="1"/>

<xs:element name="contactar" type="datosContactar" minOccurs="1" maxOccurs="1"/>

</xs:sequence>

<!-- Atributos del elemento usuario -->

<xs:attribute name="id" type="xs:string"/>

</xs:complexType>

<xs:complexType name="datosDireccion">

<xs:sequence>

<xs:element name="domicilio" type="xs:string" minOccurs="0" maxOccurs="1"/>

<xs:element name="codigo\_postal" minOccurs="0" maxOccurs="1" >

<xs:complexType>

<xs:attribute name="cp" type="xsd:string"/>

</xs:complexType>

</xs:element>

<xs:element name="localidad" type="xs:string" minOccurs="0" maxOccurs="1"/>

<xs:element name="provincia" type="xs:string" minOccurs="0" maxOccurs="1"/>

</xs:sequence>

</xs:complexType>

<xs:complexType name="datosContactar">

<xs:sequence>

<xs:element name="telf\_casa" type="xs:string" minOccurs="0" maxOccurs="1"/>

<xs:element name="telf\_movil" type="xs:string" minOccurs="0" maxOccurs="1"/>

<xs:element name="telf\_trabajo" type="xs:string" minOccurs="0" maxOccurs="1"/>

<xs:element name="email" minOccurs="0" maxOccurs="unbounded" >

<xs:complexType>

<xs:attribute name="href" type="xs:string"/>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:schema>

## 3.3. Definición de tipos de datos XML Schema

En los DTD se diferencia entre los elementos terminales y los no terminales ¿en este caso también?

Si, este lenguaje permite trabajar tanto con datos simples como con estructuras de datos complejos, es decir, compuestos por el anidamiento de otros datos simples o compuestos.

* **Tipos de datos simples**. Estos datos se suelen definir para hacer una restricción sobre un tipo de datos XSD ya definido y establece el rango de valores que puede tomar. También se pueden crear tipos de datos simples basados en listas de valores utilizando el atributo derivedBy de simpleType.
* **Tipos de datos compuestos**. El elemento xsd:complexType permite definir estructuras complejas de datos. Su contenido son las declaraciones de elementos y atributos, o referencias a elementos y atributos declarados de forma global. Para determinar el orden en que estos elementos aparecen en el documento XML se utiliza el propio elemento.

### 3.3.1. Ejercicios

1.- Creación de un elemento simple de nombre edad que representa la edad de un alumno de la ESO, por tanto su rango está entre los 12 y los 18 años.

<xs:element name="edad">

<xs:simpleType>

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

<xs:minInclusive value="12"/>

<xs:maxInclusive value="18"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

2.- Creación de una lista con los días de la semana en letras.

<xs:simpleType name="dia\_semana" base="xs:string" derivedBy="list"/>

<dia\_semana>Lunes Martes Miercoles Jueves Viernes Sabado Domingo<dia\_semana>

</xs:simpleType>

3. Creación de un elemento compuesto de nombre alumno, formado por los elementos nombre, apellidos, web personal.

<xs:complexType name="alumno">

<xs:secuence>

<xs:element name="nombre" type="xs:string" minOccurs="1" maxOccurs="1"/>

<xs:element name="apellidos" type="xs:string" minOccurs="1" maxOccurs="1"/>

<xs:element name="web" type="xs:string" minOccurs="0" maxOccurs="5">

<xs:complexType>

<xs:attribute name="href" type="xs:string"/>

</xs:complexType>

</xs:element>

<xs:secuence>

</xs:complexType>

## 3.4. Asociación con documentos XML

Dado el ejemplo de esquema de la sección 3.2.1. construye un documento XML que cumpla las especificaciones definidas en el archivo XML Schema.

<?xml version="1.0" encoding="ISO-8859-1"? >

<alumnos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:SchemaLocation="file:/D:/ubicación/del archivo/alumnos.xsd">

<alumno>

<nombre>Jose Ramón</nombre>

<apellidos>García González</apellidos>

<direccion>

<domicilio>El Pez, 12</domicilio>

<codigo\_postal>85620</código\_postal>

<localidad>Suances</localidad>

<provincia>Cantabria</provincia>

</direccion>

<contactar>

<telf.\_casa>985623165</telf.\_casa>

<telf.\_movil>611233544</telf.\_movil>

<telf.\_trabajo>965847536</telf.\_trabajo>

<email>pepito@educadistancia.com</email>

</contactar>

</alumno>

<alumno>

<nombre>Carlos</nombre>

<apellidos>López Pérez</apellidos>

<direccion>

<domicilio>El Cangrejo, 25</domicilio>

<codigo\_postal>86290</código\_postal>

<localidad>Santillana</localidad>

<provincia>Cantabria</provincia>

</direccion>

<contactar>

<telf.\_casa>931132565</telf.\_casa>

<telf.\_movil>623863544</telf.\_movil>

<telf.\_trabajo>984657536</telf.\_trabajo>

<email>carlos@educadistancia.com</email>

</contactar>

</alumno>

</alumnos>

## 3.5. Documentación del esquema

En lugar de usar los comentarios, XML Schema tiene definido un elemento xs:annotation que permite guardar información adicional. Este elemento a su vez puede contener una combinación de otros dos que son:

* **xs:documentation**, además de contener elementos de esquema puede contener elementos XML bien estructurados. También permite determinar el idioma del documento mediante el atributo xml:lang.
* **xs:appinfo**, se diferencia muy poco del elemento anterior, aunque lo que se pretendió inicialmente era que xs:documentation fuese legible para los usuarios y que xs:appinfo guardase información para los programas de software. También es usado para generar una ayuda contextual para cada elemento declarado en el esquema.

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

<xs:annotation>

<xs:documentation xml:lang ="es-es">

Materiales para formación e-Learning

<modulo>Lenguajes de marcas y sistemas de gestión de información.<modulo>

<fecha\_creación> 2011<fecha\_creacion>

<autor> Nuky La Bruji</autor>

</xs:documentation>

</xs:annotation>

<xs:element name="lmsgi" type=xs:string>

<xs:annotation>

<xs:appinfo>

<texto\_de\_ayuda>Se debe de introducir el nombre completo del tema</texto\_de\_ayuda>

<xs:appinfo>

</xs:annotation>

</xs:element>

</xs:schema>

# 4. Herramientas de creación y validación

* Editix XML Editor (Versiones open source y comercial).
* Microsoft Core XML Services (MSXML) (Gratuito).
* XMLFox (freeware).
* Altova XML Spy Edición Estándar (comercial).
* Editor XML xmlBlueprint. (comercial)
* Stylus Studio 2001 (comercial).
* Oxygen XML Editor (comercial).
* Exchanger XML Editor (comercial)
* XML copy editor (open source).

# Introducción a XML Schema

Lenguaje también conocido como XSD (XML Schema Definition

Su objetivo es describir la estructura de un documento XML:

* Elementos y atributos que pueden aparecer.
* Número y orden de los elementos hijo.
* Tipos de datos de los elementos y atributos.
* Valores por defecto y fijos para cada elemento y atributo.

## Ventajas

Soporta tipos de datos por lo tanto es más sencillo:

* Describir los elementos que puede contener el archivo.
* Validar los datos.
* Definir restricciones.
* Definir patrones y formatos de datos.
* Convertir datos entre diferentes tipos.

Usa sintaxis XML por lo tanto:

* Se pueden usar editores y parsers XML para trabajar con el.
* Se puede trabajar sobre el esquema con el DOM XML.
* Se puede transformar el esquema con XSLT.

Los esquemas XML son extensibles puesto que están escritos en XML, por lo tanto:

* Es posible reutilizar esquemas.
* Crear nuestros propios tipos de datos a partir de tipos estándar.
* Hacer referencia a múltiples esquemas en el mismo documento.

Comunicaciones seguras

* XML esquema permite al emisor describir cómo son los datos que va a recibir. Por ejemplo si enviamos a un receptor una fecha como 04 11 2018 puede interpretarse como:
  + 4 de noviembre
  + 11 de abril

Dependiendo del país en el que nos encontremos. Sin embargo si tenemos un documento XML con un elemento como:

*<fecha type =“date”>2018 11 04</fecha>*

Estaremos asegurando el entendimiento ya que el tipo de datos “date” exige ajustarse al formato “YYYY MM DD” (año, mes, día).

## DTD vs XML SCHEMA

Un DTD para definir el archivo anterior podría ser:

*<!ELEMENT email (de, para, asunto, cuerpo)>*

*<!ELEMENT de (#PCDATA)>*

*<!ELEMENT para (#PCDATA)>*

*<!ELEMENT asunto (#PCDATA)>*

*<!ELEMENT cuerpo (#PCDATA)>*

Y se haría referencia desde el archivo XML así:

*<?xml version =”1.0"?>*

*<!DOCTYPE email SYSTEM "email.dtd">*

*<email>*

*<de>Jorge </de>*

*<para>Bea </para>*

*<asunto>Fiesta el viernes </asunto>*

*<cuerpo>Recuerda que el viernes no hay clase! </cuerpo>*

*</email>*

El equivalente XSD al DTD del ejemplo anterior sería:

*<?xml version ="1.0” encoding ="UTF 8"?>*

*<xs:schema xmlns:xs http://www.w3.org/2001/XMLSchema”>*

*<xs:element name =”email”>*

*<xs:complexType>*

*<xs:sequence>*

*<xs:element name ="de” type “xs:string”/>*

*<xs:element name ="para” type “xs:string”/>*

*<xs:element name ="asunto” type “xs:string”/>*

*<xs:element name ="cuerpo” type “xs:string”/>*

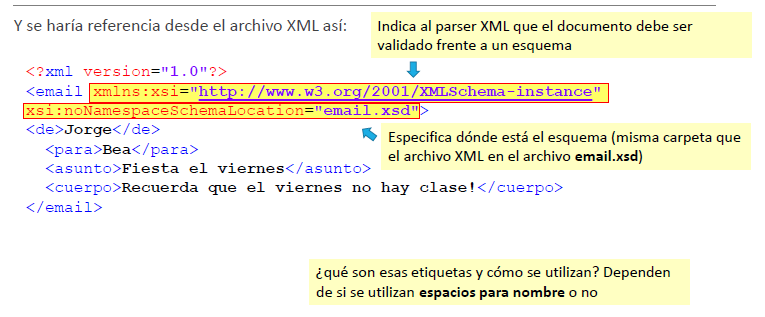
*</ xs:sequence>*

*</ xs:complexType>*

*</ xs:element>*

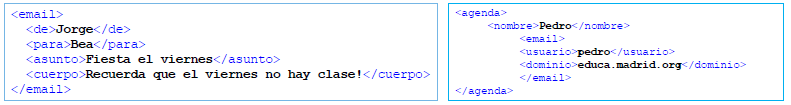
*</xs:schema>*

Se haría referencia así:

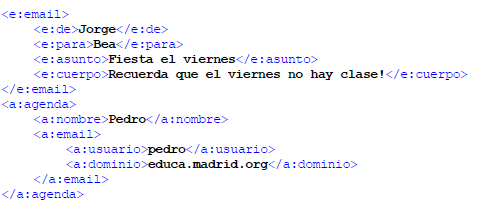


## Espacios para nombres

En XML los nombres de los elementos y atributos son definidos por el desarrollador, esto puede ocasionar un problema cuando se intentan mezclar documentos XML de diferentes orígenes.



Para unir los dos, podríamos resolver el problema usando un prefijo:



En XML cuando usamos prefijos debe definirse un espacio para nombres para el prefijo

Se define mediante el atributo xmlns (XML NameSpace ) en la etiqueta inicial del elemento

* También puede definirse en el elemento raiz

La declaración tiene la siguiente sintaxis: *xmlns:prefijo =“URI”*

El parser en ningún caso busca información en el URI, simplemente se utiliza para dar un nombre único al espacio para nombres.

* Algunas compañías usan el URI para apuntar a una página web en la que habrá información sobre el espacio para nombres.

### 2 posibilidades de definición

#### Opción 1

*<raíz>*

*<e:email xmlns:e http://educa.madrid.org/correos/”>*

*<e:de> Jorge </e:de>*

*<e:para> Bea </e:para>*

*<e:asunto> Fiesta el viernes e:asunto>*

*<e:cuerpo> Recuerda que el viernes no hay clase! </e:cuerpo>*

*</e:email>*

*<a:agenda xmlns:e http://educa.madrid.org/ jorge />*

*<a:nombre> Pedro <a/:nombre>*

*<a:email>*

*<a:usuario> pedro </a:usuario>*

*<a:dominio> educa.madrid.org <a:dominio>*

*</a:email>*

*</a:agenda>*

*</raíz>*

#### Opción 2

*<raíz xmlns:e=”*[*http://educa.madrid.org/correos/*](http://educa.madrid.org/correos/)*”*

*xmlns:e=”*[*http://educa.madrid.org/ jorge/agenda/*](http://educa.madrid.org/%20jorge/agenda/)*”>*

*<e:email>*

*<e:de> Jorge </e:de>*

*<e:para> Bea </e:para>*

*<e:asunto> Fiesta el viernes </e:asunto>*

*<e:cuerpo> Recuerda que el viernes no hay clase! </e:cuerpo>*

*</e:email>*

*<a:agenda> a:nombre Pedro </a:nombre>*

*<a:email>*

*<a:usuario> pedro </a:usuario>*

*<a:dominio> educa.madrid.org </a:dominio>*

*</a:email>*

*</a:agenda>*

*</raíz>*

## Vinculando archivos XML y XSD

### Sin espacio para nombres

#### Xml

*<raiz xmlns:xsi=”http://www.w3.org/2001/XMLSchem- instance”*

*xsi:noNamespaceSchemaLocation ="esquemaAsociado.xsd”>*

*<!-- … --> </ raíz>*

#### XSD

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

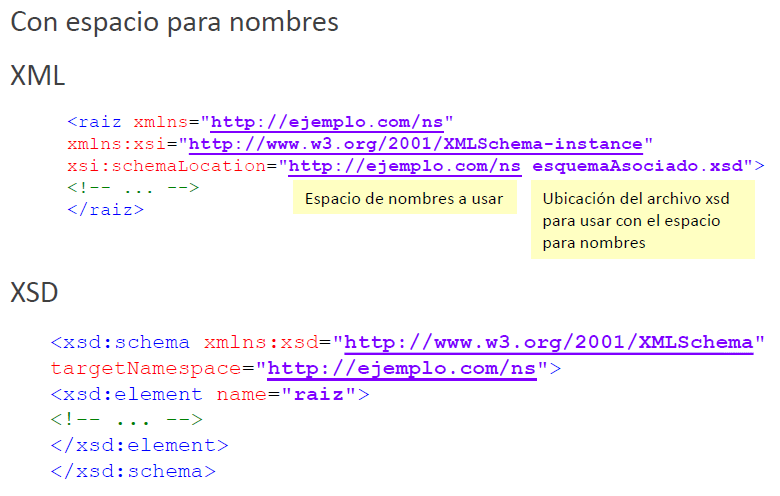
<xsd:element name=”raíz”>

<!-- ... -->

</xsd:element>

</xsd:schema>

### Con espacio para nombres



## Elementos simples XSD

Un elemento simple contiene solo texto, no puede contener otros elementos o atributos.



Name es el nombre del elemento y type el tipo de datos al que se corresponde. Entre los tipos de datos más comunes encontramos:

* xs:string
* xs:decimal
* xs:integer
* xs:boolean
* xs:date
* xs:time

default y fixed son opcionales, default representan el valor por defecto que se asigna al elemento cuando no recibe ningún valor y fixed indica un valor automático que se asigna al elemento y que no permite especificar otro.

## Atributos XSD

Los atributos se declaran siempre como tipos simples.



**Name** es el nombre del atributo y **type** el tipo de datos al que pertenece. Al igual que los elementos simples puede pertenecer a tipos de datos como:

* xs:string
* xs:decimal
* xs:integer
* xs:boolean
* xs:date
* xs:time

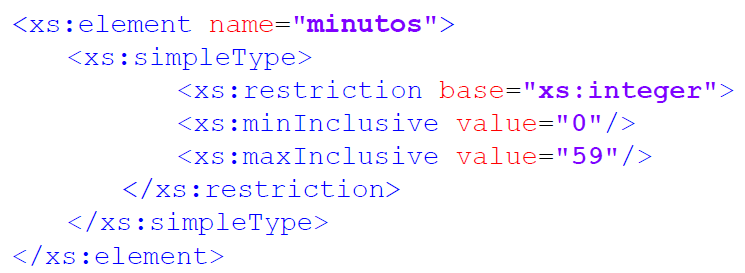
**default** y **fixed** son opcionales, **default** representan el valor por defecto que se asigna al elemento cuando no recibe ningún valor y fixed indica un valor automático que se asigna al elemento y que no permite especificar otro.

Los atributos son opcionales. Para indicar que es obligatorio se debe especificar mediante *use=”required”*

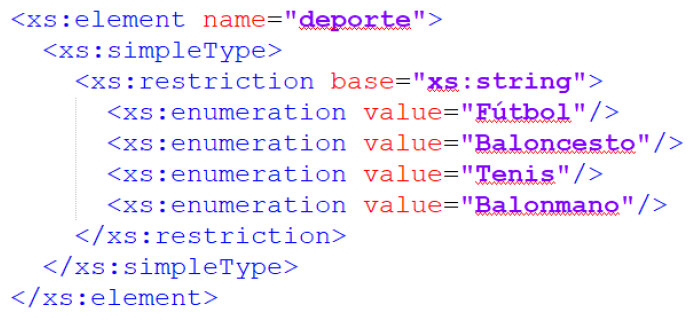
## Restricciones XSD

Lasrestricciones (o facetas) permiten definir el rango de valores aceptable para elementos o atributos XML.

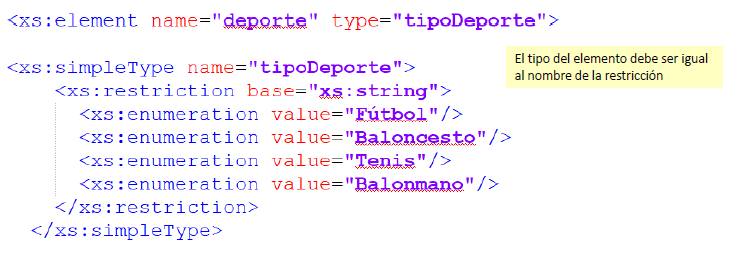
Restricción para los valores del elemento minutos entre 0 y 59.



Restringir los valores de un elemento para que tengan que estar dentro de un conjunto de valores.



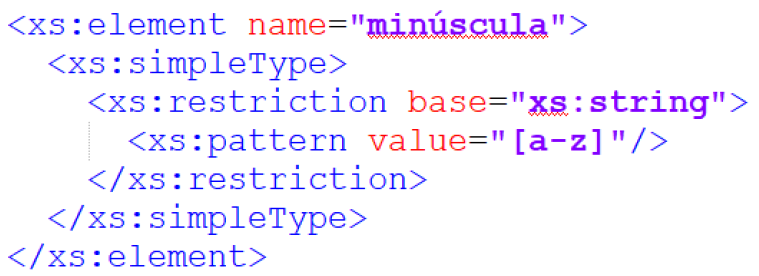
Se pueden definir restricciones de manera independiente al elemento para que puedan ser usados por otros elementos.



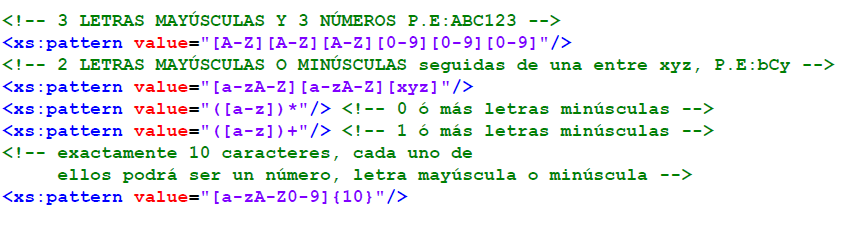
Restricciones de patrones

Es posible definir un patrón al que deben ajustarse los valores introducidos.

* En este caso solo se podrá introducir una letra minúscula

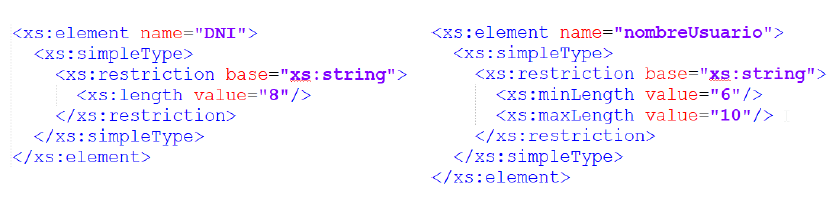


Otras restricciones



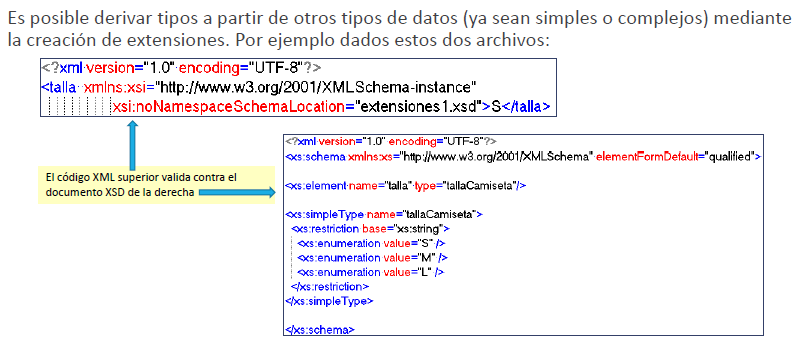
Restricciones de longitud

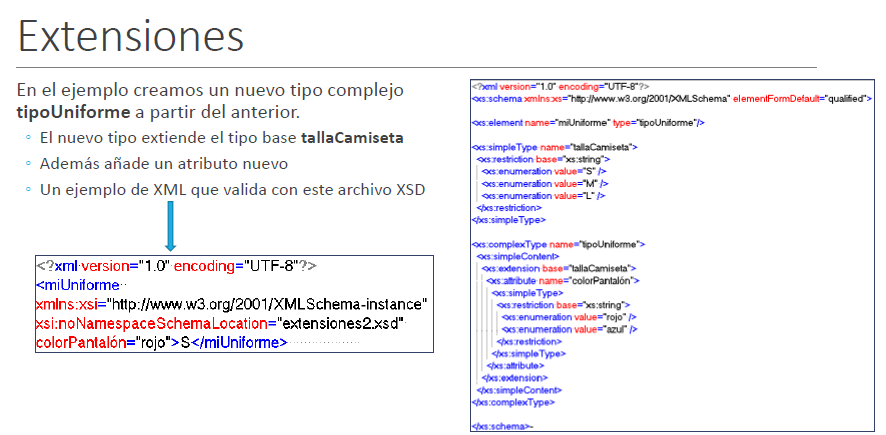
Es posible establecer la longitud exacta de un elemento, también se puede especificar un mínimo y un máximo.



|  |  |
| --- | --- |
| Restricción | Descripción |
| **enumeration** | Define una lista de valores aceptados |
| **fractionDigits** | Máximo número de decimales. |
| **length** | Número exacto de caracteres. |
| **maxExclusive** | Valor máximo para elementos numéricos |
| **maxInclusive** | Valor máximo para elementos numéricos(el propio límite también es aceptado) |
| **maxLength** | Máximo número de caracteres |
| **minExclusive** | Valor mínimo para elementos numéricos |
| **minInclusive** | Valor mínimo para elementos numéricos(el propio límite también es aceptado) |
| **minLength** | Mínimo número de caracteres |
| **pattern** | Patrón al que deben ajustarse los valores introducidos |
| **totalDigits** | Número total de dígitos permitidos. |

## Extensiones





## Elementos complejos

Un elemento complejo contiene otros elementos y/o atributos.

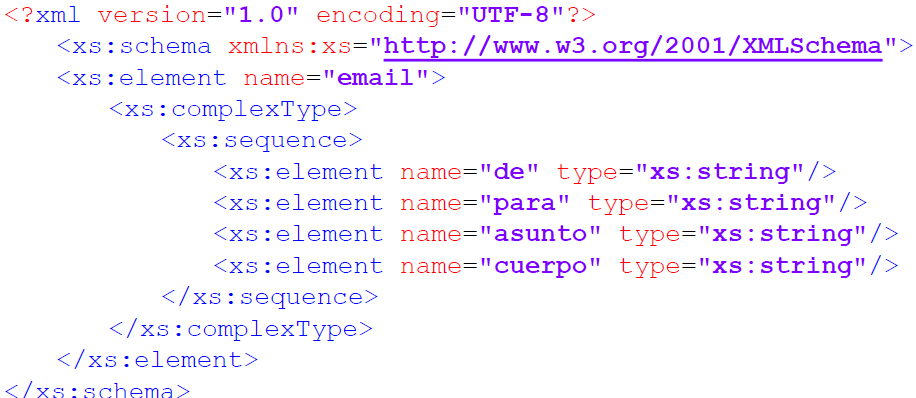
Hay cuatro tipos:

* Elementos vacíos.
* Elementos que contienen otros elementos.
* Elementos que contienen solo texto.
* Elementos que contienen otros elementos y texto.

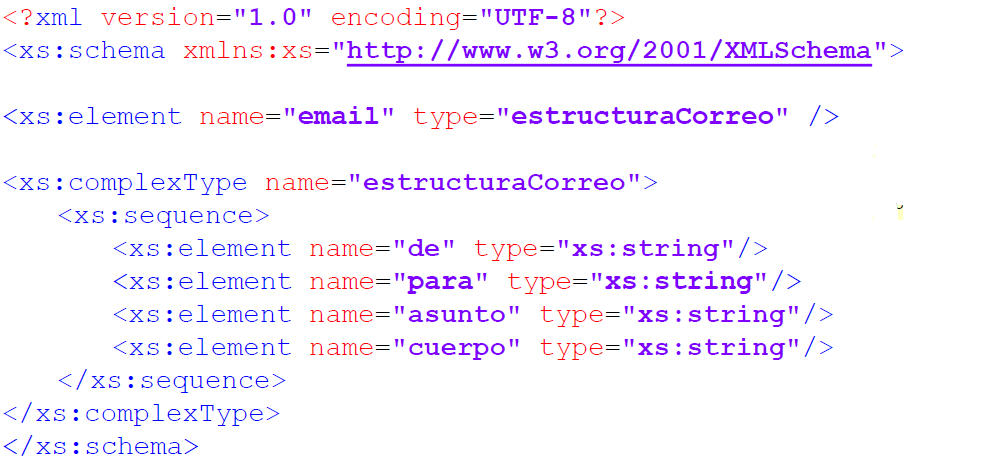
|  |  |
| --- | --- |
| **<cochematrícula="1345HVF"/>** | **<email>**  **<de>Jorge</de>**  **<para>Bea</para>**  **<asunto>Fiesta el viernes</asunto>**  **<cuerpo>Recuerda que el viernes no hayclase!</cuerpo>**  **</email>** |
| **<productotalla=“L"/>**  **Camiseta fitness**  **</producto>** | **<productotalla=“L"/>**  **Camiseta fitness**  **<marca> blanca</marca>**  **</producto>** |

Para definir un elemento complejo podemos usar dos métodos.

* Declarando el elemento raíz y definiendo los elementos que lo forman

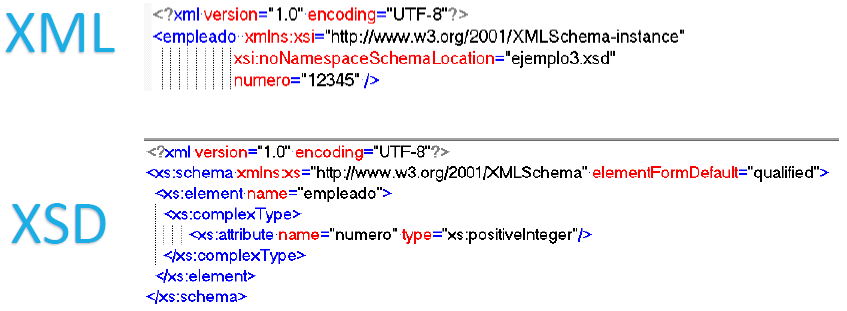


Usando un atributo type que indique el tipo de datos complejo que forma el elemento.

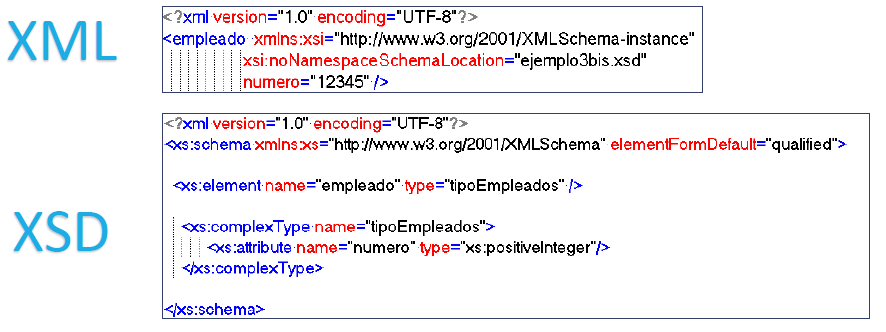


## Elementos vacíos

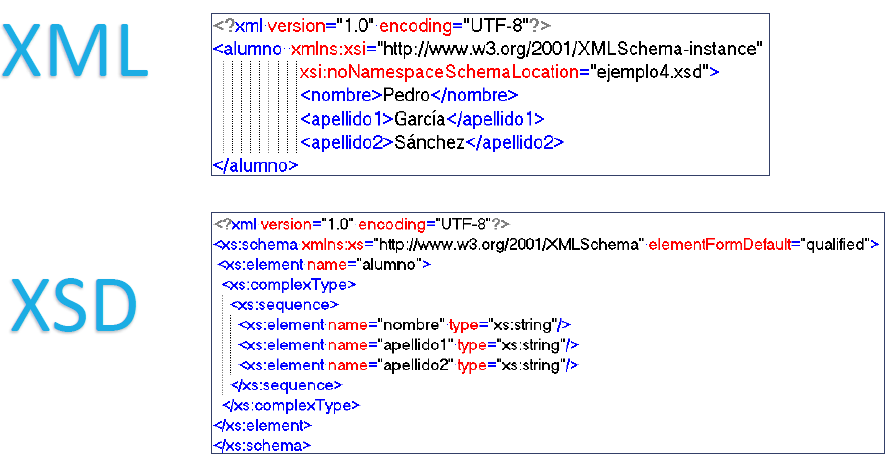
Para definir un tipo sin contenidos es necesario definir un tipo que permite elementos en su contenido pero sin declarar ninguno.



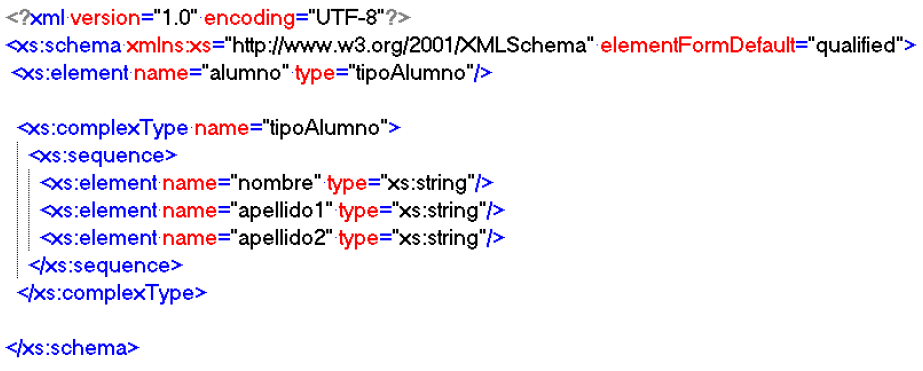
Al igual que ocurría en las facetas es posible independizar la declaración del tipo del propio elemento permitiendo reutilizar el tipo en varios elementos.



La etiqueta < xs:sequence > especifica que el orden en el que aparecen los elementos definidos dentro de ella debe respetarse en el documento XML.



Evidentemente y al igual que en ejemplos anteriores también es correcto declarar el tipo de manera independiente.

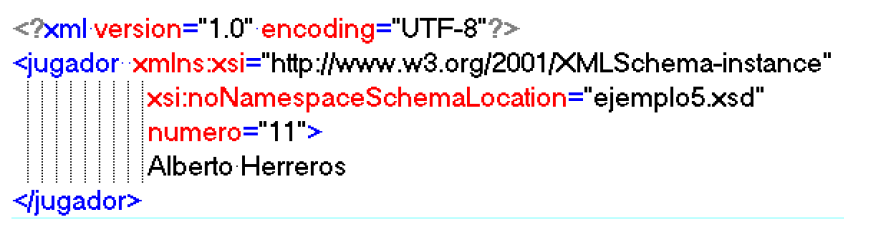


### Elementos complejos que solo contienen texto

En este caso debemos definir una extensión con el tipo de elemento contenido.

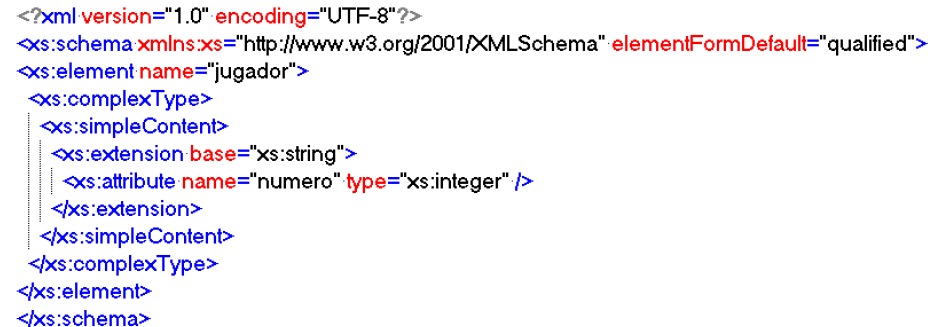
* También es posible definir restricciones, pero este caso solo cuando se desea limitar el tipo base utilizado para el elemento.

Por ejemplo, queremos validar este archivo XML:



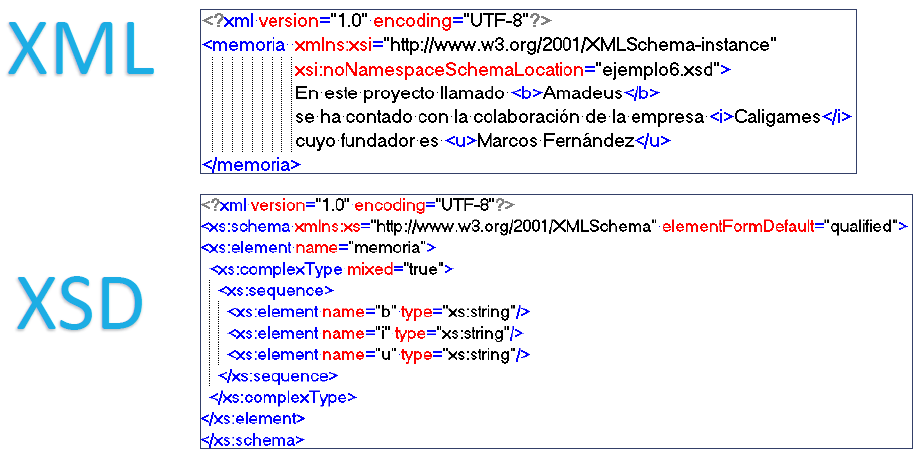
Extensión del elemento contenido.

* El tipo base al que corresponde el elemento contenido es string . Lo definimos como tipo base y lo extendemos.



### Elementos complejos con contenido mixto

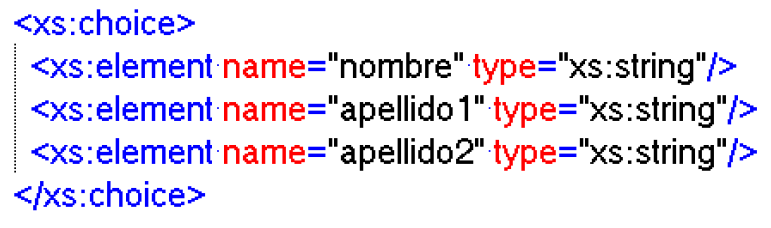
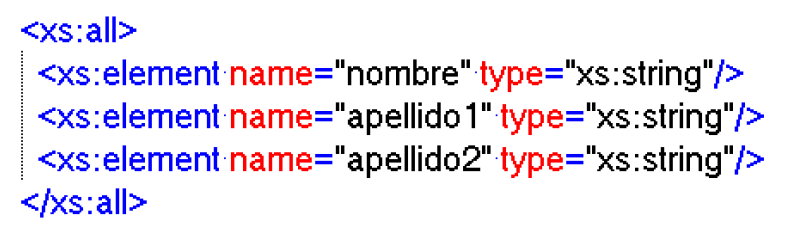
Supongamos contenidos en los que se alternan elementos, atributos y texto. Para permitir que se mezclen datos es necesario permitir que el atributo mixed esté a valor True



## Indicadores XSD

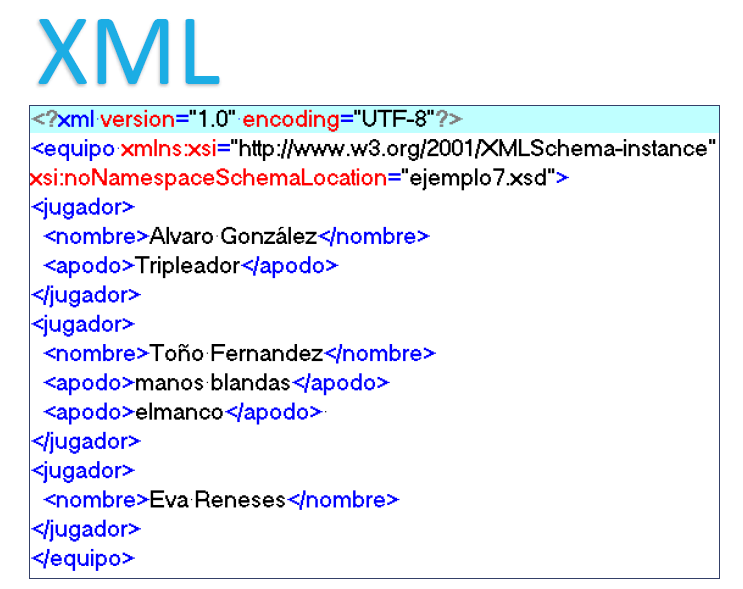
Nos permiten controlar cómo se utilizan los elementos dentro de los documentos XML

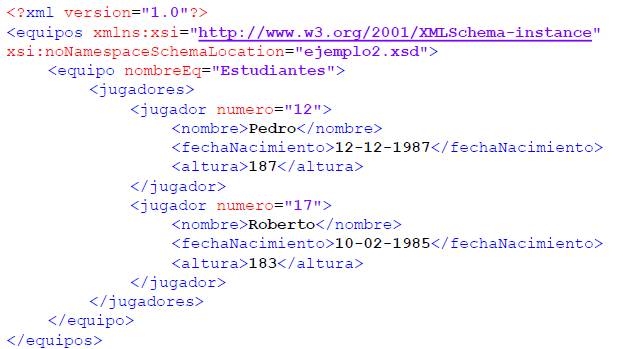
* Elementos de orden:
  + All . Los elementos hijos pueden aparecer en cualquier orden. Cada elemento solo puede aparecer una
  + Choice . Solamente puede aparecer uno de los elementos
  + Sequence . Los elementos hijos deben aparecer en el orden



Indicadores de ocurrencias.

* Nos permiten indicar el número de veces que puede aparecer en elemento.
* El número máximo se especifica con maxOccurs
* El número mínimo se especifica con minOccurs
* Para indicar que no hay límite se usa la palabra unbounded





Se utiliza el elemento any

El elemento a incorporar podría estar definido en otro archivo xsd

|  |  |
| --- | --- |
|  |  |

Se utiliza el elemento anyAttribute

Funcionamiento similar a any pero con atributos

|  |  |
| --- | --- |
|  |  |