

Definición de esquemas y vocabularios en xml.

Caso práctico

María había salido a dar un paseo por el puerto, como todos los domingos por la mañana iba acompañada de su marido José Ramón y su hijo.

Mientras les veía jugar, estaba pensando que al día siguiente tenía que hablar con Juan, el técnico que se encargaba de resolver los problemas de **informática de la empresa de la cual es socia**. Había estado pensando si existiría algún modo de poder garantizar que la **estructura de datos** de cada uno de los **documentos XML que comparte con Félix**, su socio, es la que tiene que ser y no otra, además de asegurar que todos los documentos del mismo tipo mantienen la misma estructura.



Nate Steiner María y su hijo (CC BY)

1.- Documento XML. Estructura y sintaxis.

Caso práctico

Al día siguiente, **cuando habla con Juan**, sus dudas quedan disipadas. Resulta que hay **varias posibilidades para asegurar una normalización en el formato de los documentos XML**.

Juan comienza por describir la estructura de un documento XML y Félix y María descubren que puede ser un poco más compleja que la que habían estado usando hasta entonces para generar sus documentos.

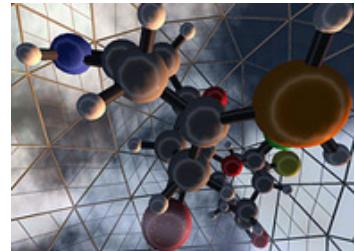


Nate Steiner María hablando (CC BY)

Hasta ahora hemos trabajado con documentos básicos de XML. Esto significa que dichos documentos están incompletos ya que solo hemos declarado el tipo de documento que va a ser, es decir que ejemplar vamos a definir, pero no hemos definido qué cualidades tiene ese tipo.

En la primera unidad vimos que un documento XML básico estaba formado por un prólogo y un ejemplar.

Recordamos que cada una de esas partes tiene el siguiente cometido:



Hiro Sheridan Estructura molecular gigante y móvil del museo de ACS Island. (CC BY)

- ✓ **Prólogo:** Informa al intérprete encargado de procesar el documento de todos aquellos datos que necesita para realizar su trabajo. Consta de dos partes:
 - ➡ **Definición de XML:** Donde se indica la versión de XML que se utiliza, el código de los datos a procesar y la autonomía del documento. Este último dato hasta ahora siempre ha sido "yes" ya que los documentos generados eran independientes.
 - ➡ **Declaración del tipo de documento:** Hasta el momento solo hemos dicho que es el nombre del ejemplar precedido de la cadena "".

```
<?xml version=1.0" encoding=UTF-8" standalone"yes"?><br /><!DOCTYPE ejemplo>
```

- ✓ **Ejemplar:** Contiene los datos del documento que se quiere procesar. **Es el elemento raíz del documento y ha de ser único**. Está compuesto de elementos estructurados según una estructura de árbol en la que el elemento raíz es el ejemplar y las hojas los elementos terminales, es decir, aquellos que no contienen más elementos. Los elementos pueden estar a su vez formados por atributos.

Autoevaluación

Marcar los componentes de un documento XML:

- Prólogo.

Ejemplar.

Definición de codificación del documento.

Cabecera.

[Mostrar retroalimentación](#)

Solución

1. Correcto
2. Correcto
3. Correcto
4. Incorrecto

1.1.- Declaración de tipo de documento.

Ya habíamos visto que permite al autor definir restricciones y características en el documento, aunque no habíamos profundizado en las partes que la forman:

- ✓ **La declaración del tipo de documento propiamente dicha.** Comienza con el texto que indica el nombre del tipo, precedido por la cadena " separado del nombre del tipo por, al menos, un espacio. El nombre del tipo ha de ser idéntico al del ejemplar del documento XML en el que se está trabajando.
- ✓ **La definición del tipo de documento.** Permite asociar al documento una definición de tipo DTD, la cual se encarga de definir las cualidades del tipo. Es decir, define los tipos de los elementos, atributos y notaciones que se pueden utilizar en el documento así como las restricciones del documento, valores por defecto, etc. Para formalizar todo esto, XML está provisto de ciertas estructuras llamadas declaraciones de marcado, las cuales pueden ser internas o externas. Normalmente un documento XML se compone de una mezcla de declaraciones de marcado internas y externas. En este último caso debe expresarse en el documento dónde encontrar las declaraciones, así como indicar en la declaración de XML que el documento no es autónomo. Las diferencias entre estos tipos de declaraciones de marcado dan lugar a dos subconjuntos, el interno y el externo. Conviene saber que primero se procesa el subconjunto interno y después el externo, lo que permite sobrescribir declaraciones externas compartidas entre varios documentos y ajustar el DTD a un documento específico.
 - ↳ Subconjunto interno: Contiene las declaraciones que pertenecen exclusivamente a un documento y no es posible compartir las. Se localizan dentro de unos corchetes que siguen a la declaración de tipo del documento.
 - ↳ Subconjunto externo: Están localizadas en un documento con extensión dtd que puede situarse en el mismo directorio que el documento XML. Habitualmente son declaraciones que pueden ser compartidas entre múltiples documentos XML que pertenecen al mismo tipo. En este caso la declaración de documento autónomo ha de ser negativa, ya que es necesario el fichero del subconjunto externo para la correcta interpretación del documento. Con ello el procesado del documento será más lento, ya que antes de procesar el documento el procesador ha de obtener todas las entidades. Ahora los corchetes pierden sentido, para localizar las declaraciones del tipo de documento externo mediante una declaración explícita de subconjunto externo se utiliza:
 - URI">
En este caso, se especifica un URI donde pueden localizarse las declaraciones.
 - 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.



Petra Gregorová
Definición de tipo de documento (CC BY-ND)

Autoevaluación

La definición de tipo de documento ha de ser:

- Interna o externa al documento XML al que está referida.
- Interna al documento XML que le refiere.
- Externa al documento XML que le refiere.
- Única.

Muy bien. Has captado la idea. Correcta.

Incorrecta, porque puede ser externa.

No es la respuesta correcta porque puede definirse en el documento.

No es correcta ya que un documento puede tener una definición DTD interna u otra externa definidas simultáneamente.

Solución

- 1. Opción correcta**
- 2. Incorrecto**
- 3. Incorrecto**
- 4. Incorrecto**

1.2.- Definición de la sintaxis de documentos XML.

Recordamos que en estos documentos las etiquetas de marcado describen la estructura del documento.

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

En los documentos de lenguajes de marcas, la distribución de los elementos está jerarquizada según una estructura de árbol, lo que implica que es posible anidarlos pero no entrelazarlos.

Hemos visto que en los elementos el orden es importante, ¿lo es también para los atributos? En este caso el orden no es significativo. Lo que hay que tener presente es que no puede haber dos atributos con el mismo nombre en el mismo elemento.

Sabemos que los atributos no pueden tener nodos que dependan de ellos, por tanto solo pueden corresponder con hojas de la estructura de árbol que jerarquiza los datos. ¿Significa esto que todas las hojas van a ser atributos? Pues no, es cierto que los atributos son hojas, pero las hojas pueden ser atributos o elementos.

En ese caso, ¿qué criterios podemos utilizar para decidir si un dato del documento que se pretende estructurar ha de representarse mediante un elemento o un atributo? Aunque no siempre se respetan, podemos usar los siguientes criterios:

- ✓ 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.

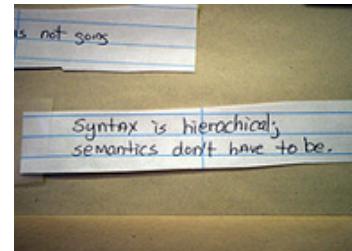
¿Cómo se declaran?

`xmlns:"URI_namespace"`

`<nombre xmlns="https://educacionadistancia.juntadeandalucia.es/EspacioNombres">Ejemplo</nombre>`

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

`xmlns:prefijo="URI_namespace"`



En ambos casos URI_namespace es la localización del conjunto del vocabulario del espacio de nombres al que se hace referencia.

Autoevaluación

Marcar las afirmaciones válidas que hacen referencia a un espacio de nombres.

- Facilitan que el software localice el vocabulario de la aplicación.

- Define los atributos que forman parte de un documento.

- Permiten tener varios elementos homónimos con diferentes significados.

- Determinan los elementos que forman parte de un documento XML.

[Mostrar retroalimentación](#)

Solución

1. Correcto
2. Incorrecto
3. Correcto
4. Incorrecto

2.- Definiciones de tipo de documento, DTD.

Caso práctico



[Jonny Goldstein Juan \(CC BY\)](#)

Según Juan, el método más sencillo para intentar normalizar los documentos con los que trabajan, consiste en definir unos vocabularios que han de cumplir los documentos que generan, estos se llaman Definición de Tipo de Documento. Además, aunque no es un lenguaje XML, tiene una sintaxis sencilla y fácil para que ella y Félix puedan comprenderla y utilizarla.

Una definición de tipo de documento o DTD es una descripción de estructura y sintaxis de un documento XML o SGML. Su función básica es la descripción de la estructura de datos, para usar un diseño común y mantener la consistencia entre todos los documentos que utilicen la misma DTD. De esta forma, dichos documentos pueden ser validados, conocen la estructura de los elementos y la descripción de los datos que trae consigo cada documento.

Dos o más documentos XML que tengan la misma DTD, se construyen de forma similar, tienen el mismo tipo de etiquetas y en el lugar, orden y cantidad especificados en la DTD.

En temas anteriores se explicó cuándo un documento XML estaba bien formado o era correcto. Pero, igual que para construir una frase en castellano de forma correcta no solo hace falta escribirla sin faltas de ortografía, un documento XML bien formado no es necesariamente válido. Para que un documento XML sea válido debe primero estar bien formado y después seguir las especificaciones dictadas por la DTD.

Las 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.



[Leo Reynolds. DTD \(CC BY-NC-SA\)](#)

- ✓ 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.

Ejercicio Resuelto

Al siguiente documento XML tenemos que agregarle un DTD que lo valide.

Con '". El nombre que aparece a continuación debe ser la raíz del documento, es decir, pelicula, a su vez se indica que la etiqueta título está dentro de la etiqueta película.

```
<?xml version="1.0"?>
<pelicula>
  <titulo>Titanic</titulo>
</pelicula>
```

[Mostrar retroalimentación](#)

```
<?xml version="1.0"?>
<!DOCTYPE pelicula [
  <!ELEMENT pelicula (titulo)>
  <!ELEMENT titulo (#PCDATA)>
]>
<pelicula>
  <titulo>Titanic</titulo>
</pelicula>
```

Autoevaluación

Marcar las afirmaciones referidas a un DTD:

Permite la formación de identificadores compuestos por varios atributos.

Para construirlas no se usa un lenguaje basado en XML.

No definen distintos tipos para los datos.

[Mostrar retroalimentación](#)

Solución

1. Incorrecto
2. Correcto
3. Correcto

2.1.- Declaración de la DTD

La DTD está compuesta por lo que aparece entre la etiqueta "`"` y la etiqueta ""]>". El nombre corresponde con la etiqueta raíz que debe tener el documento XML.

Existen dos formas de definir la DTD que describirá la estructura de un documento XML. Se puede incluir dentro del mismo documento, o incluir la información sobre su ubicación (enlace a un documento DTD).

- ✓ **DTD incrustada:** Es posible incluir la DTD en el mismo documento como se ha visto anteriormente. En el siguiente ejemplo está resaltada en negrita:

```
<?xml version="1.0" encoding="UTF-8"?>
<strong><!DOCTYPE pelicula [
    <!ELEMENT pelicula (titulo)>
    <!ELEMENT titulo (#PCDATA)>
]></strong>
<pelicula>
    <titulo>Titanic</titulo>
</pelicula>
```



jcomp / Freepik. Declaración (CC BY)

Se puede proporcionar una ayuda extra al analizador de XML, si a través de las instrucciones de proceso presentes en el prólogo, se indica que el documento es independiente y que todo lo que necesita está contenido en el mismo. Para ello basta con añadir el atributo `standalone="yes"`, como puede verse a continuación:

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

El valor por defecto del atributo `standalone` es `no`, es decir, en caso de omitir el atributo es como si se pusiera `standalone="no"`.

- ✓ **DTD externa:** En el ejemplo anterior, la DTD estaba incrustada en el documento XML. Es posible separar ambos elementos, guardándolos en archivos diferentes. Así, se puede situar en un archivo la DTD, en este caso se llama `cine.dtd`

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?> <!-- Enlace a la DTD -->
<!DOCTYPE pelicula SYSTEM "cine.dtd">
<pelicula>
    <titulo>Titanic</titulo>
</pelicula>
```

No hace falta poner `standalone="no"` ya que por defecto toma ese valor.

Debes conocer

Las ventajas de incluir una DTD externa son las siguientes:

- ✓ Si la DTD que se va a incluir es compartida por muchos documentos XML, es preferible que se encuentre en un archivo independiente ya que si hay que hacer algún cambio en la DTD, solo tendrá que hacerse en un archivo y no en cada XML en el caso de tener una DTD incrustada en el mismo documento XML.
- ✓ La DTD puede ubicarse en un servidor web, de forma que cualquier persona con acceso a Internet puede validar el documento XML que está creando, lo que garantiza que todos los documentos XML usan la última versión de la DTD. Para declarar que nuestra DTD está en un servidor de Internet basta con especificarlo en el DOCTYPE de la siguiente forma:

```
<!DOCTYPE cine SYSTEM "http://cine.com/filmoteca.dtd">
```

Aunque es más correcto si se pone de forma pública. Se hace así:

```
<!DOCTYPE cine PUBLIC "filmoteca" "http://cine.com/filmoteca.dtd">
```

Siendo filmoteca el nombre de la DTD.

2.2.- Declaraciones de tipos de elementos terminales.

Los tipos de elementos 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 " 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:



Karl-Ludwig Poggemann
Elemento terminal (CC BY)

- ✓ **EMPTY:** Indica que el elemento no es contenedor, es vacío, es decir, que no puede tener contenido. Por ejemplo, la siguiente definición muestra un elemento "ejemplo" que no contiene nada: .
 - ⇒ XML asociado correcto:

```
<ejemplo></ejemplo> ó <ejemplo />
```

- ⇒ XML asociado incorrecto:

```
<ejemplo>Esto es un ejemplo</ejemplo>
ó
<ejemplo><a></a></ejemplo>
```

- ✓ **(#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 "ejemplo" tendrá una definición como:

```
<!ELEMENT ejemplo (#PCDATA)>.
```

- ⇒ XML asociado correcto:

```
<ejemplo />
ó
<ejemplo>Esto es un ejemplo</ejemplo>
```

- ⇒ XML asociado incorrecto:

```
<ejemplo><a></a></ejemplo>
```

- ✓ **ANY:** Permite que el contenido del elemento sea cualquier cosa (texto y otros elementos). No es recomendable usar este tipo de elemento.

- ⇒ Ejemplos de XML correcto con elemento ANY:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mascota[
    <!ELEMENT mascota ANY>
```

```
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT tipo(#PCDATA)><br />  <!ELEMENT raza(#PCDATA)>
]>
```

Para este vocabulario, estos documentos XML son válidos:

```
<mascota>
  <nombre>Coco</nombre> es mi mascota. <br />Es una <tipo>chinchilla</tipo> <raza>blanca</raza>.
</mascota>

<mascota>
  Coco es mi mascota. <br />Es una <tipo>chinchilla</tipo> <raza>blanca</raza>.
</mascota>

<mascota>
  <nombre>Coco</nombre>.
</mascota>

<mascota/>
```

Ejercicio resuelto

Creación de una DTD correspondiente a la siguiente estructura de datos de un documento XML:

```
<alumno>Olga Velarde Cobo</alumno>
```

[Mostrar retroalimentación](#)

```
<!ELEMENT alumno (#PCDATA)>
```

Autoevaluación

Los elementos terminales de tipo ANY son aquellos que están:

- Formados por cadenas de texto exclusivamente.
- Vacíos.
- Formados por cualquier cosa.

- Estará formada por otros elementos.**

No es correcta porque pueden estar formados por cualquier cosa.

Incorrecta, porque también pueden ser otra cosa.

Ánimo, sigue así. ¡Correcta!

No es correcta, los elementos terminales son aquellos que no están formados por otros elementos.

Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta
4. Incorrecto

2.3.- 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:

En este caso se ha definido un elemento A que está formado por un elemento B seguido de un elemento C.

¿Y qué sucede cuando un elemento puede aparecer en el documento varias veces, hay que indicarlo de algún modo? Pues sí, también hay que indicar cuando un elemento puede no aparecer. Para ello usamos los siguientes operadores, que nos permiten definir la cardinalidad de un elemento:



Bill Barber. Árbol (CC BY-NC)

- ✓ 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) )>
```

Ejercicio resuelto

Creación de un DTD correspondiente a la siguiente estructura de datos de un documento XML:

```
<alumno>
  <nombre>Olga</nombre>
  <dirección>El Percebe 13</dirección>
</alumno>
```

Mostrar retroalimentación

```
<!ELEMENT alumno (nombre, dirección)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT dirección (#PCDATA)>
```

Autoevaluación

La diferencia entre el operador ? y el + es que el primero permite que el elemento sobre el que se aplica esté presente una vez, como máximo, mientras que el operador + no limita el número máximo de veces que está presente el elemento en el documento XML, sólo el mínimo.

- Verdadero.**
- Falso.**

Muy bien, correcto.

Incorrecto, el operador ? es el que determina que un elemento es opcional mientras que + determina que un elemento aparecerá una o más veces.

Solución

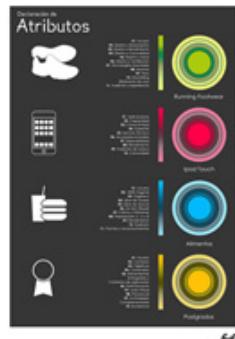
1. Opción correcta
2. Incorrecto

2.4.- Declaraciones de listas de atributos para los tipos de elementos.

Ya sabemos cómo declarar elementos, ahora veamos el modo de declarar los atributos asociados a un elemento.

Para ello utilizamos la cadena seguida del nombre del elemento asociado al atributo que se declara, luego el nombre de éste último seguido del tipo de atributo y del modificador.

```
<! ATTLIST <strong>nombre_elemento_del_atributo</strong> nombre_atribut
```



[frvelizadic](#). Lista de atributos
(CC BY-NC-ND)

Este elemento puede usarse para declarar una lista de atributos asociada a un elemento, o repetirse el número de veces necesario para asociar a dicho elemento esa lista de atributos, pero individualmente. Si un elemento tiene más de un atributo se puede expresar:

```
<! ATTLIST <strong>nombre_elemento_del_atributo1</strong> nombre_atributo1 tipo_del_atributo1
```

Al igual que los elementos no todos los atributos son del mismo tipo, los más destacados son:

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

↳ DTD:

```
<!ATTLIST fecha dia_semana (lunes|martes|miércoles|jueves|viernes|sábado|domingo) #RE
```

- ✓ CDATA, se utiliza cuando el atributo es una cadena de texto.

↳ DTD:

```
<!ATTLIST ejemplo color CDATA #REQUIRED>
```

↳ XML asociado:

```
<ejemplo color="verde" />
```

- ✓ ID, permite declarar el valor del atributo (no el nombre) debe ser único y no se puede repetir en otros elementos o atributos. 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.

↳ DTD:

```
<!ATTLIST libro <b>codigo ID #REQUIRED</b>>
```

↳ XML asociado:

```
<libro codigo="Q1">El Quijote</libro>
```

- ✓ 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.
- ✓ NMOKEN, permite determinar que el valor de un atributo ha de ser una sola palabra compuesta por los caracteres permitidos por XML, es decir letras, números y los caracteres ":" , "_" , "-" o ".".
- ✓ ¿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.
- ◆ DTD:

```
<!ATTLIST ejemplo color CDATA #IMPLIED>
```

- ◆ XML asociado:

```
<ejemplo color="verde" /> o <ejemplo color="" />
```

- ◆ #REQUIRED, determina que el atributo tiene carácter obligatorio.
- ◆ DTD:

```
<!ATTLIST ejemplo color CDATA #REQUIRED>
```

- ◆ XML asociado NO válido:

```
<ejemplo color="" />
```

- ◆ XML asociado válido:

```
<ejemplo color="verde" />
```

- ◆ #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.
- ◆ DTD:

```
<!ATTLIST ejemplo color CDATA #FIXED "verde">
```

- ◆ XML asociado NO válido:

```
<ejemplo color="rojo" />
```

- ◆ XML asociado válido:

```
<ejemplo color="verde" />
```

- ◆ Literal, asigna por defecto a un atributo el valor dado por una cadena entre comillas, pero puede tomar otro valor.

- ◆ DTD:

```
<!ATTLIST ejemplo color CDATA (rojo|verde|amarillo) "verde">
```

- ◆ XML asociado:

```
<ejemplo color="verde" />
```

Ejercicio resuelto

Creación de un DTD correspondiente a la siguiente estructura de datos de un documento XML:

```
<alumno edad=15>
  <nombre>Olga</nombre>
  <apellidos>Velarde Cobo</apellidos>
  <dirección>El Percebe 13</dirección>
</alumno>
```

Mostrar retroalimentación

```
<!ELEMENT alumno (nombre, apellidos, dirección)>
<!ATTLIST alumno edad CDATA #REQUIRED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT dirección (#PCDATA)>
```

Autoevaluación

¿Cuál de los siguientes atributos permite añadir a los datos una restricción de integridad referencial?

- NMTOKEN.**
- ID.**
- IDREF.**
- CDATA.**

Incorrecto, los datos de este tipo garantizan que tienen un nombre que cumple las normas de XML.

No es correcto. Este atributo impone la restricción que ese valor debe ser único para ese atributo dentro de un documento XML.

Genial, es correcto.

No está bien. Este tipo de datos se usa para cadenas de texto.

Solución

- 1. Incorrecto**
- 2. Incorrecto**
- 3. Opción correcta**
- 4. Incorrecto**

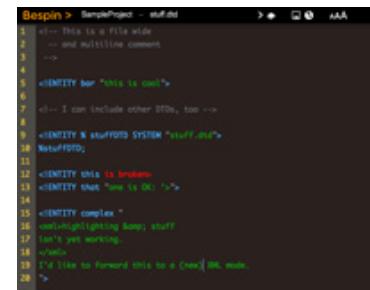
2.5.- Declaraciones de entidades.

¿Qué sucede si queremos declarar valores constantes dentro de los documentos? ¿Podemos?

Las entidades nos permiten definir constantes en un documento XML. Cuando se usan dentro del documento XML se limitan por "&" y ";", por ejemplo & entidad;

¿Cómo trabaja el intérprete con ellos? Al procesar el documento XML, el intérprete sustituye la entidad por el valor que se le ha asociado en el DTD.

No admiten recursividad, es decir, una entidad no puede hacer referencia a ella misma.



```
<!DOCTYPE libro [<!ELEMENT libro (autor,editorial)><!ELEMENT autor (#PCDATA)><!ELEMENT editorial (#PCDATA)><!ENTITY autor "Cervantes"><!ENTITY editorial "Alfaguara">]><libro>Don Quijote de la Mancha fue escrito por &autor; </libro><libro>SIDI fue escrito por Arturo Pérez-Reverte y publicado por &editorial; </libro><libro>Tiempos recios fue escrito por Mario Vargas Llosa y publicado por &editorial; </libro></libros>
```

Axel Hecht. Entidades (CC BY-ND)

Las entidades básicamente pueden ser de dos tipos, generales y de parámetro.

✓ Generales:

↳ Internas: Son las entidades declaradas en el DTD.

● Existen cinco entidades predefinidas en el lenguaje, son:

- ✖ <: Se corresponde con el signo menor que, <.
- ✖ >: Hace referencia al signo mayor que, >.
- ✖ "": Son las comillas rectas dobles, " ".
- ✖ ': Es el apóstrofe o comilla simple, ' '.
- ✖ &: Es el et o ampersand, &.

● Podemos definir nuestras propias entidades usando la estructura:, donde:

- ✖ nombre_entidad es el nombre que recibe la entidad
- ✖ "valor_entidad" es el valor que toma dicha entidad
- ✖ se hace referencia a la entidad usando &nombre_entidad;
- ✖ Veamos un ejemplo. Dado el DTD:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE libros [
3      <!ELEMENT libros (libro)+>
4      <!ELEMENT libro (#PCDATA)>
5
6          <!ENTITY autor "Cervantes">
7          <!ENTITY editorial "Alfaguara">
8      ]>
9
10 <libros>
11     <libro>Don Quijote de la Mancha fue escrito por &autor; </libro>
12     <libro>SIDI fue escrito por Arturo Pérez-Reverte y publicado por &editorial; </libro>
13     <libro>Tiempos recios fue escrito por Mario Vargas Llosa y publicado por &editorial; </libro>
14 </libros>
```

Silvia Thomas. Definición entidades internas (CC0)

Al abrir el fichero en un navegador, se vería:

Este fichero XML no parece tener ninguna información de estilo asociada. Se muestra debajo el árbol del documento.

```
--<libros>
  <libro>Don Quijote de la Mancha fue escrito por Cervantes</libro>
  <libro>
    SIDI fue escrito por Arturo Pérez-Reverte y publicado por Alfaguara
  </libro>
  <libro>
    Tiempos recios fue escrito por Mario Vargas LLosa y publicado por Alfaguara
  </libro>
</libros>
```

Silvia Thomas. XML en el navegador (CC0)

- ➡ Externas: Permiten establecer una relación entre el documento XML y otro documento a través de la URL de éste último.
 - La declaración de una entidad externa cuando sólo va a ser utilizada por una única aplicación es:

```
<!ENTITY nombre_entidad SYSTEM "http://localhost/docsxml/fichero_entidad.xml">
```

En este caso el contenido de los ficheros es analizado, por lo que deben seguir la sintaxis XML.

Cuando es necesario incluir ficheros con formatos binarios, es decir ficheros que no se analicen, se utiliza la palabra reservada NDATA en la definición de la entidad y habrá que asociar a dicha entidad una declaración de notación, tal y como muestra el ejemplo del apartado siguiente.

- ✓ La declaración de una entidad externa cuando va a ser utilizada por varias aplicaciones es:

```
<!ENTITY nombre_entidad PUBLIC "identificador público formal" "camino hasta la DTD (uri)">
```

- ✓ De parámetro:

- ➡ Internas: Permiten 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;

```
<strong><!ENTITY % direccion "calle, numero?, ciudad, cp"></strong> y se puede usar c
```

- ➡ Externas: Permite incluir en un DTD elementos externos, lo que se aplica en dividir la definición DTD en varios documentos.

```
<strong><!ENTITY %persona SYSTEM "persona.dtd"></strong>
```

Autoevaluación

Las entidades permiten definir elementos cuyo valor es constante dentro de un DTD.

- Verdadero.**
- Falso.**

Entre otras cosas permiten definir valores constantes.

Genial, es correcto.

Solución

1. Incorrecto
2. Opción correcta

2.6.- Declaraciones de notación.

Cuando se incluyen ficheros binarios en un fichero, ¿cómo le decimos qué aplicación ha de hacerse cargo de ellos? La respuesta es utilizando notaciones. La sintaxis para declarar notaciones es:

```
<!NOTATION nombre SYSTEM aplicación>
```

Por ejemplo, una notación llamada gif donde se indica que se hace referencia a un editor de formatos gif para visualizar imágenes será:

```
<!NOTATION gif SYSTEM "gifEditor.exe">
```

Para asociar una entidad externa no analizada, a esta notación basta declarar dicha entidad del siguiente modo:

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



MattieTK. Notas musicales (CC BY-NC-SA)

Autoevaluación

Las notaciones permiten:

- Establecer el modo en el que se ha de estructurar un fichero XML.
- Determinar cuál es la aplicación que ha de procesar un fichero binario asociado a un fichero XML.
- Asociar un fichero binario a un fichero XML.
- Definir las etiquetas válidas en un fichero XML.

No es correcta pues sirve para determinar la aplicación que ha de tratar un fichero binario.

Estupendo, continúa como hasta ahora.

No es la respuesta correcta porque eso es lo que hace una entidad.

Incorrecto, esa es la labor del DTD.

Solución

1. Incorrecto
2. Opción correcta
3. Incorrecto
4. Incorrecto

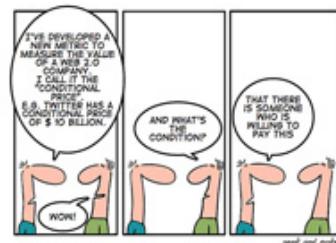


2.7.- Secciones condicionales.

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] ] >
```



Por ejemplo:

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

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

```
<!IGNORE [Declaraciones ocultas] ] >
```

Por ejemplo:

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

El uso de las secciones condicionales suele estar ligado a entidades paramétricas.

Autoevaluación

Las sentencias condicionales permiten definir unos elementos u otros dentro del fichero XML en función de una determinada condición.

- Verdadero.
- Falso.

Será mejor que leas de nuevo el apartado.

Muy bien. Has captado la idea...

Solución

1. Incorrecto
2. Opción correcta

Ejercicio resuelto

Vamos a crear un dtd para guardar la información sobre un alumno y queremos tener la posibilidad de obtener el registro de dos maneras diferentes, una corta que incluya únicamente el nombre y la edad del alumno y la otra que incluya además los apellidos, la dirección donde vive y su DNI.

[Mostrar retroalimentación](#)

Creamos un archivo llamado "alumno.dtd" y escribimos:

```
%datos_basicos; [  
]  
]  
  
%datos_ampliados; [  
]  
]
```

El siguiente documento XML sería válido:

```
INCLUDE">  
IGNORE">  
]>
```

```
Silvia  
13
```

También sería válido el documento:

```
IGNORE">  
INCLUDE">  
]>
```

```
Ana  
Gil Lechuga  
14  
C/ Granada sn. Almería 22222222-S/>
```

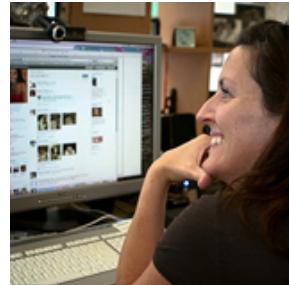
Obsérvese que, en este ejemplo, en los dos documentos XML asociados a la DTD externa escrita en el archivo "alumno.dtd", se ha indicado –por medio de IGNORE e INCLUDE si el elemento "alumno" tiene que contener 2 ó 5 hijos, es decir, ("nombre" y "edad") o ("nombre", "apellidos", "edad", "direccion" y "DNI").

3.- XML Schema.

Caso práctico

Félix, quien considera que la normalización de los documentos XML que manejan en la empresa va a ser un duro trabajo para María, él y otros trabajadores inexpertos, plantea la posibilidad de que se encargue de ello algún trabajador de la consultoría informática que dirige Juan.

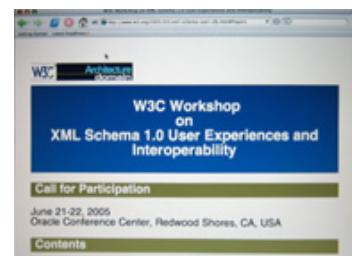
Al final se va a encargar de ello Marina. Les explica que, en lugar de trabajar con DTD's le parece mejor hacerlo con un lenguaje XML llamado XML Schema, el cual tiene, entre otras, la ventaja de permitir definir el tipo de datos de cada uno de los componentes de cada documento.



See-ming Lee 李思明 SML, Marina
(CC BY-SA)

Los DTD permiten diseñar un vocabulario para ficheros XML, pero, ¿qué sucede cuando los valores de los elementos y atributos de esos ficheros han de corresponder a datos de un tipo determinado, o cumplir determinadas restricciones que no pueden reflejarse en los DTD? Para ello se definen XML Schemas, que se componen de elementos y atributos, al igual que los DTD.

¿También se definen en ficheros planos? Si, ya que son documentos XML, pero en este caso la extensión de los archivos es xsd, motivo por el cual también se les denomina documentos XSD.



psd. Schema (CC BY)

Los elementos XML que se utilizan para generar un esquema han de pertenecer al espacio de nombres XML Schema, que es: <http://www.w3.org/2001/XMLSchema>. En esta especificación se usa el prefijo , pero para abreviar se usa , aunque en la práctica cualquier prefijo puede ser usado, siempre que se use el mismo prefijo en todo el documento.

Las estructuras que se definen en XML Schema definen a su vez numerosos atributos para uso directo en cualquier documento XML y están en un espacio de nombres diferente, <http://www.w3.org/2001/XMLSchema-instance>. En esta especificación se usa el prefijo , pero para abreviar se usa , aunque en la práctica cualquier prefijo puede ser usado, siempre que se use el mismo prefijo en todo el documento.

De esta manera, los prefijos , y se pueden utilizar indistintamente al definir el esquema siempre que en el mismo se utilice únicamente uno de ellos.

El ejemplar de estos ficheros es , contiene declaraciones para todos los elementos y atributos que puedan aparecer en un documento XML asociado válido. Los elementos hijos inmediatos de este ejemplar son que nos permiten crear globalmente un elemento. Esto significa que el elemento creado puede ser el ejemplar del documento XML asociado.

El elemento puede tener algunos atributos. La declaración de un esquema suele tener el siguiente aspecto:

```
<?xml version="1.0" encoding="UTF-8"?><br />
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema">
  targetNamespace="https://www.w3schools.com"
  xmlns="https://www.w3schools.com"
```

```
elementFormDefault="qualified">
...<br />
...<br />
</xs:schema>
```

El siguiente fragmento:

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

indica que los elementos y tipos de datos usados en el esquema vienen del espacio de nombres "http://www.w3.org/2001/XMLSchema". También especifica que los elementos y los tipos de datos que provengan de dicho espacio de nombres deben tener el prefijo xs:. Este fragmento es el único obligatorio para que la definición sea correcta.

El siguiente fragmento:

targetNamespace="https://www.w3schools.com"

indica que los elementos definidos en el esquema pertenecen al espacio de nombres de "https://www.w3schools.com". Por defecto toma este valor.

El siguiente fragmento:

xmlns="https://www.w3schools.com"

indica que el espacio de nombres por defecto es "https://www.w3schools.com".

El siguiente fragmento:

elementFormDefault="qualified"

indica que cualquier elemento usado en una instancia xml que ha sido declarada con este esquema debe ser identificado con el espacio de nombres. Por defecto toma este valor.

Ejercicio resuelto

Creación de un esquema correspondiente al siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<alumno>Olga Velarde Cobo</alumno>
```

Mostrar retroalimentación

```
< ?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>
```

Para saber más

En este primer enlace encontrarás los fundamentos del estándar XML Schema.

[Fundamentos XML Schema](#)

3.1.- Tipos de elementos en XML Schema

Un elemento se utiliza para describir datos. Recordemos que su etiqueta más habitual suele ser xsd:element, xsi:element o xs:element.

Los elementos se utilizan para especificar las etiquetas válidas en un documento XML. Todos los elementos que se vayan a utilizar en el ejemplar XML tienen que estar declarados en el esquema. Las declaraciones de elementos en XML Schema tienen una estructura diferente dependiendo de si son simples o complejos:

- ✓ **Tipo simple:** No pueden contener otros elementos o atributos.

Su estructura es la siguiente:

```
<xsd:element name="nombreElemento"
ref="elementoReferenciado"
type="tipoDato"
minOccurs="valor"
maxOccurs="valor"
fixed="valor"
default="valor"/>
```

Donde:

- ✓ **name:** es el nombre del elemento.
- ✓ **ref:** el elemento al que hace referencia está declarado en otro lugar del esquema. No puede aparecer junto con "name" y ni si el elemento padre es
- ✓ **type:** el tipo de dato del elemento. No puede aparecer si se usa "ref".
- ✓ **minOccurs** y **maxOccurs** (Opcionales): estos dos atributos indican el mínimo (minOccurs) y máximo (maxOccurs) número de ocurrencias del elemento. El valor por defecto en ambos casos es 1. Para indicar que el elemento puede aparecer un número ilimitado de veces, el atributo maxOccurs toma el valor "unbounded". Para indicar que un elemento puede no aparecer, el atributo minOccurs toma el valor 0. Ninguno de los atributos se puede usar si el elemento padre es .
- ✓ **fixed** (Opcional): especifica un valor fijo para el elemento.
- ✓ **default** (Opcional): especifica un valor por defecto para el elemento.
- ✓ Se pueden crear nuevos elementos "simpleType" a partir de uno ya existente, añadiendo condiciones a alguno de los tipos predefinidos en el XML Schema. Para ello se usa el elemento

- Ejemplo:

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



[O_sonha_dor](#). Elementos de linea y de bloque (CC BY-NC)

- ✓ **Tipo complejo:** Pueden estar compuestos por otros elementos y/o atributos. Su contenido está definido entre las etiquetas de inicio y de cierre del elemento .

- ◆ Esquema, xs:schema, contiene la definición del esquema, es el elemento tipo complejo básico.
- ◆ xs:complexType, entre sus etiquetas de inicio y cierre se definen los elementos de tipo complejo. Pueden estar formados por subelementos predefinidos en XML Schema como:
 - Secuencias, xs:sequence, permite construir elementos complejos mediante la enumeración de los elementos que los forman en un orden concreto. Si se altera dicho orden en el documento xml, dicho documento no será válido.
 - Alternativa, xs:choice, representa alternativas, hay que tener en cuenta que es una o-exclusiva. Especifica una lista concreta de elementos de los que sólo puede aparecer uno.
 - Secuencias no ordenadas, xs:all, representa a todos los elementos que componen el elemento de tipo compuesto en cualquier orden. Pueden aparecer en el documento xml en cualquier orden y dicho documento es válido.
 - Contenido mixto, definido dando valor true al atributo mixed del elemento <xs:complexType mixed="true">, permite mezclar texto con elementos hijo. Los elementos hijo se definen con las opciones anteriores; xs:sequence, xs:choice o xs:all.
 - Elemento vacío, el elemento no puede contener texto ni otros subelementos; únicamente atributos.

❖ XSD:

```
<xsd:element name="asignatura">
  <xsd:complexType>
    <xsd:attribute name='codigo' type='xs:integer' />
  </xsd:complexType>
</xsd:element>
```

❖ XML válido:

```
<asignatura codigo='MAT-1920'><b>(vacío)</b></asignatura>

<strong><i>0 lo que es lo mismo y más correcto:</i></strong>

<asignatura codigo='MAT-1920' />
```

- ✓ Referencias a otros elementos: Tal y como ocurre en otros lenguajes, en un documento xsd podemos definir elementos de forma global y luego hacer referencias a ellos desde otros elementos. Esto es muy útil si a lo largo de un documento se repiten determinados elementos.

◆ Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Direccion">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="via"/>
        <xsd:element ref="numero"/>
        <xsd:element ref="poblacion"/>
        <xsd:element ref="provincia"/>
        <xsd:element ref="cp"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="via" type="xsd:string"/>
  <xsd:element name="numero" type="xsd:integer"/>
  <xsd:element name="poblacion" type="xsd:string"/>
  <xsd:element name="provincia" type="xsd:string"/>
  <xsd:element name="cp" type="xsd:string"/>
</xsd:schema>
```

Ejercicio resuelto

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

Mostrar retroalimentación

```
<?xml version="1.0" encoding="UTF-8"?>
<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="un
        </xs:sequence>
    </xs:complexType>

    <!-- Definicion del tipo datos -->

    <xs:complexType name="datos">
        <xs:sequence>
            <xs:element name="nombre" type="xs:string" minOccurs="1" maxOccur
            <xs:element name="apellidos" type="xs:string" minOccurs="1" maxOcc
            <xs:element name="direccion" type="datosDireccion" minOccurs="1" m
            <xs:element name="contactar" type="datosContactar" minOccurs="1" m
        </xs:sequence>

        <!-- Atributos del elemento alumno-->

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

    <xs:complexType name="datosDireccion">
        <xs:sequence>
            <xs:element name="domicilio" type="xs:string" minOccurs="0" maxOcc
            <xs:element name="codigo_postal" minOccurs="0" maxOccurs="1" >
                <xs:complexType>
                    <xs:attribute name="cp" type="xs:string"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="localidad" type="xs:string" minOccurs="0" maxOcc
            <xs:element name="provincia" type="xs:string" minOccurs="0" maxOcc
        </xs:sequence>
    </xs:complexType>
```

```
<xs:complexType name="datosContactar">
  <xs:sequence>
    <xs:element name="telf_casa" type="xs:string" minOccurs="0" maxOcc
    <xs:element name="telf_movil" type="xs:string" minOccurs="0" maxOc
    <xs:element name="telf_trabajo" type="xs:string" minOccurs="0" max
    <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>
```

Autoevaluación

Para hacer un elemento complejo formado por un listado de elementos en los que importa el orden hay que usar el elemento:

- <xs:choice>
- <xs:group>
- <xs:all>
- <xs:sequence>

No es correcta porque este elemento permite elegir entre elementos.

Incorrecta, porque este elemento permite hacer agrupaciones de elementos y atributos para hacer referencia a ellos dentro del esquema.

No es la respuesta correcta porque no respeta el orden de los elementos.

¡Genial! Ánimo, que está muy bien.

Solución

- 1. Incorrecto
- 2. Incorrecto
- 3. Incorrecto
- 4. Opción correcta

Para saber más

En el siguiente enlace puedes ver ejemplos de declaración de elementos en un esquema.

[Elementos en un esquema](#)

Este enlace te permitirá consultar las estructuras del estándar XML Schema.

[XML Schema Estructuras](#)

3.2.- Atributos en XML Schema

El elemento "atributo" permite definir los atributos de los elementos en el documento xsd para usarlos adecuadamente en el documento XML.

Su sintaxis es la siguiente:

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

Donde:

- ✓ name: nombre del atributo.
- ✓ ref: el atributo referenciado se encuentra definido en otra parte del esquema. No puede aparecer al mismo tiempo que name.
- ✓ type: tipo de dato del atributo. No puede aparecer al mismo tiempo que ref.
- ✓ use (opcional): indica si la aparición del atributo es opcional (optional), obligatoria (required) o prohibida (prohibited). Por defecto toma el valor "optional".
- ✓ default (opcional): valor que tomará el atributo al ser procesado si en el documento XML no hay ningún valor. Sólo se puede usar con tipos de datos cadena de caracteres. No puede aparecer al mismo tiempo que fixed.
- ✓ fixed (opcional): valor fijo que toma el atributo. No puede aparecer al mismo tiempo que default.

Los atributos sólo pueden aparecer en los elementos de tipo compuesto y su declaración debe aparecer siempre al final de la definición del elemento del que es atributo, justo antes del cierre de .

Ejercicio Resuelto

Crear el código xsd para el siguiente xml

```
<libro codigo="001"><br />  
  <titulo>Don Quijote</titulo> <br />  
  <autor>Miguel de Cervantes</autor><br />  
</libro>
```

[Mostrar retroalimentación](#)

```
<xsd:element name="libro"><br />  
  <xsd:complexType><br />  
    <xsd:sequence><br />  
      <xsd:element name="nombre" type="xs:string"/><br />  
      <xsd:element name="autor" type="xs:string"/>  
    </xsd:sequence><br />
```

```
<xs:attribute name="codigo" type="xs:string"/><br />
</xs:complexType><br />
</xs:element>
```

3.3.- 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.



floresyplantas.net. Tipos (CC BY-NC-SA)

Algunos de estos valores predefinidos son:

- ✓ string, se corresponde con una cadena de caracteres UNICODE. Puede contener caracteres, saltos de línea y tabulaciones.
 - ⇒ XSD:

```
<xs:element name="poblacion" type="xs:string"/>
```

- ⇒ XML válido:

```
<poblacion>La Puebla de Vícar</poblacion><br /><strong><i>0 lo que es lo mismo:</i></strong><poblacion> La Puebla de Vícar </poblacion>
```

- ✓ boolean, representa valores lógicos, es decir que solo pueden tomar dos valores, true o false.

- ⇒ XSD:

```
<xs:attribute name="cancelado" type="xs:boolean"/>
```

- ⇒ XML válido:

```
<vuelo cancelado="true">LK345</vuelo><br />
```

- ✓ integer, número entero positivo o negativo.

- ⇒ XSD:

```
<xs:element name="precio" type="xs:integer"/>
```

- ⇒ XML válido:

```
<precio>94</precio>
```

- ✓ positiveInteger, número entero positivo.

- ✓ negativeInteger, número entero negativo.

- ✓ decimal, número decimal, por ejemplo, 8,97.

- ⇒ XSD:

```
<xs:element name="precio" type="xs:decimal"/>
```

- ⇒ XML válido:

```
<precio>8,97</precio><br /><strong>o también</strong><br /><precio>8</precio><br />
```

- ✓ dateTime, representa una fecha y hora absolutas. Tiene el siguiente formato: "YYYY-MM-DDThh:mm:ss. Sólo es válido si se especifican todos los componentes.

- ⇒ XSD:

```
<xs:element name="fecha" type="xs:dateTime"/>
```

⇒ XML válido:

```
<fecha><span>2020-05-20T08:20:00</span></fecha><br /><strong>o bien</strong><br /><fe
```

- ✓ duration, representa una duración de tiempo expresado en años, meses, días, horas, minutos segundos. El formato utilizado es: PnYnMnDTnHnMnS. Para indicar una duración negativa se pone un signo – precediendo a la P.

⇒ XSD:

```
<xss:element name="periodo" type="xs:duration"/>
```

⇒ XML válido:

```
<strong>Por ejemplo para representar una duración de 2 años, 4 meses, 3 días, 5 horas <periodo>P2Y4M3DT5H6M7S</periodo><br /><strong>Se pueden omitir los valores nulos, lu
```

- ✓ 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.

⇒ XSD:

```
<xss:element name="fecha" type="xs:gYearMonth"/>
```

⇒ XML válido:

```
<fecha><span>2020-05-20T08:20:00</span></fecha> <strong>Mayo de 2020</strong><br />
```

⇒ XML NO válidos:

```
<fecha><span>20-05</span></fecha> <strong>El año no puede dividirse</strong><br /><s
```

- ✓ gYear, indica un año gregoriano, el formato usado es CCYY.
✓ gMonthDay, representa un día de un mes mediante el formato --MM-DD.

⇒ XSD:

```
<xss:element name="fecha" type="xs:gMonthDay"/>
```

⇒ XML válido:

```
<fecha><span>--05-19</span></fecha> <strong>19 de Mayo</strong><br />
```

⇒ XML NO válidos:

```
<fecha><span>05-19</span></fecha> <strong>Los guiones delante del mes son obligatori
```

- ✓ 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.

⇒ XSD:

```
<xss:element name="web" type="xs:anyURI"/>
```

⇒ XML válido:

```
<web>www.iesaguadulce.es</web> <strong>URI absoluta, incluso una URL<br /></strong>o
```

XML NO válidos:

```
<web>www.iesaguadulce.es#texto#text01</web> Demasiados identificadores #<br />
```

- ✓ language, representa los identificadores de lenguaje, sus valores están definidos en [RFC 1766](#).
- ✓ ID, IDREF, ENTITY, NOTATION, NMTOKEN. Representan lo mismo que en los DTD's (ver apartado 2.4).

Autoevaluación

¿Cuál de los siguientes tipos no hace referencia a un dato de tiempo?

- [dateTime](#).
- [duration](#).
- [gDayMonth](#).
- [gMonthDay](#).

No es correcta porque este tipo representa una fecha y hora absolutas.

Incorrecta, porque la duración es una medida de tiempo.

Correcta, muy bien.

No es correcta ya que este tipo corresponde a un día de un mes de un año cualquiera.

Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta
4. Incorrecto

Para saber más

En este enlace encontrarás los tipos de datos admitidos por el estándar.

[XML Schema Tipos de datos](#)

En el siguiente enlace podrás ver ejemplos válidos y no válidos de los distintos tipos de datos.

[Tipos de datos](#)

3.4.- Facetas de los tipos de datos.

¿Cuáles son las restricciones que podemos aplicar sobre los valores de los datos de un elemento o atributo?

Están definidas por las facetas, que solo pueden aplicarse sobre tipos simples utilizando el elemento xs:restriction, que tiene como atributo "base" en el que se indica el tipo de dato a partir del cual se define la restricción.

Las facetas se expresan como un elemento dentro de una restricción y se pueden combinar para lograr restringir más el valor del elemento. Son, entre otros:

- ✓ length, minlength, maxlength: Longitud del tipo de datos.
- ✓ enumeration: Restringe a un determinado conjunto de valores.



Cartelera Teatral chilena.
Facetas (CC BY-NC-SA)

Ejercicio resuelto

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

[Mostrar retroalimentación](#)

```
<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>
```

- ✓ whitespace: Define el tratamiento de espacios (preserve/replace, collapse).

Ejercicio resuelto

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

[Mostrar retroalimentación](#)

```

<xs:simpleType name="nombre">
    <xs:restriction base="xs:string">
        <xs:whiteSpace value="preserve"/>
    </xs:restriction>
<xs:simpleType>

```

- ✓ (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.

Ejercicio resuelto

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

[Mostrar retroalimentación](#)

```

<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>

```

- ✓ 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.

Elementos para hacer patrones.

Patrón	Significado	Patrón	Significado
[A-Z a-z]	Letra.	AB	Cadena que es la concatenación de las cadenas A y B.
[A-Z]	Letra mayúscula.	A?	Cero o una vez la cadena A.
[a-z]	Letra minúscula.	A+	Una o más veces la cadena A.
[0-9]	Dígitos decimales.	A*	Cero o más veces la cadena A.

\D	Cualquier carácter excepto un dígito decimal.	[abcd]	Alguno de los caracteres que están entre corchetes.
(A)	Cadena que coincide con A.	[^abcd]	Cualquier carácter que no esté entre corchetes.
A B	Cadena que es igual a la cadena A o a la B.		

Ejercicio resuelto

El ejemplo siguiente muestra la utilización de pattern para crear la máscara de un DNI.

[Mostrar retroalimentación](#)

```
<xs:simpleType name="dni">
    <xs:restriction base="xs:string">
        <xs:pattern value="[0-9]{8}[A-Z]" />
    </xs:restriction>
</xs:simpleType>
```

Recomendación

Visita el siguiente enlace para conocer un poco más sobre las expresiones regulares:

[Expresiones regulares](#)

3.5.- Definición de tipos de datos simples 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. En este apartado vamos a indicar las tres diferentes maneras que existen de extender un tipo de datos simple:

✓ Restricción.

Se hace una restricción sobre un tipo de datos XSD ya definido y se establece el rango de valores que puede tomar. Las restricciones son conocidas como facetas.



Víctor Gómez. Elementos simples y compuestos ([CC BY-NC-ND](#))

Ejercicio resuelto

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.

[Mostrar retroalimentación](#)

```
<xss:simpleType name="edad">
    <xss:restriction base="xsd:positiveInteger">
        <xss:maxExclusive value="19"/>
        <xss:minInclusive value="12"/>
    </xss:restriction>
</xss:simpleType >
```

✓ Unión.

Consiste en combinar dos o más tipos de datos diferentes en uno único.

Ejercicio Resuelto

Crear un tipo de dato para las medidas de longitud que una el sistema internacional (cm, m y km) con el sistema inglés (pulgada, pie, yarda)

[Mostrar retroalimentación](#)

```

<xs:simpleType name="LongitudInternacional"><br />
    <xs:restriction base="xs:string"><br />
        <xs:enumeration value="cm" /><br />
        <xs:enumeration value="m" /><br />
        <xs:enumeration value="km" /><br />
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="LongitudSajona"><br />
    <xs:restriction base="xs:string"><br />
        <xs:enumeration value="pulgada" /><br />
        <xs:enumeration value="pie" /><br />
        <xs:enumeration value="yarda" /><br />
    </xs:restriction>
</xs:simpleType>

```

Unimos los dos tipos en uno único:

```

<xs:simpleType name="Longitud">
    <xs:union memberTypes="LongitudInternacional LongitudSajona">
</xs:simpleType>

```

✓ Lista.

Permite asignar a un elemento un valor dentro de un número de valores válidos separados por espacios en blanco. Puede ser creada de manera similar a una unión, con la diferencia de que sólo puede contener un tipo de elementos.

Ejercicio Resuelto

Crear una lista con las unidades de medida de volumen sajonas.

[Mostrar retroalimentación](#)

```

<xs:simpleType name="LongitudSajona">
    <xs:restriction base="xs:string">
        <xs:enumeration value="pulgada" />
        <xs:enumeration value="pie" />
        <xs:enumeration value="yarda" />
    </xs:restriction>
</xs:simpleType>
<br /><br /><xs:simpleType name="Longitud">
    <xs:list itemType="LongitudSajona">
</xs:simpleType>

```

- ✓ También podemos usar el atributo de simpleType, derivedBy, para crear una lista de elementos.

Ejercicio resuelto

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

[Mostrar retroalimentación](#)

```
<xs:simpleType name="dia_semana" base="xs:string" derivedBy="list"/>
  <dia_semana>Lunes Martes Miercoles Jueves Viernes Sabado Domingo</dia_sema
</xs:simpleType>
```

Otra opción:

```
<xs:simpleType name="dia_semana">
  <xs:restriction base="xs:string">
    <xs:pattern value="Lunes|Martes|Miercoles|Jueves|Viernes|Sabado|Domingo"/> <x
  </xs:simpleType>
```

3.6.- Definición de tipos de datos complejos XML Schema

Tipo complejo: Pueden estar compuestos por otros elementos y/o atributos. Su contenido está definido entre las etiquetas de inicio y de cierre del elemento .

- ✓ Esquema, xs:schema, contiene la definición del esquema, es el elemento tipo complejo básico. Contiene el atributo xmlns (XML NameSpace), que indica el espacio de nombres para el esquema.

↳ XSD:

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

- ✓ xs:complexType, entre sus etiquetas de inicio y cierre se definen los elementos de tipo complejo. Pueden estar formados por subelementos predefinidos en XML Schema como:

↳ Secuencias, xs:sequence, permite construir elementos complejos mediante la enumeración de los elementos que los forman en un orden concreto. Si se altera dicho orden en el documento xml, dicho documento no será válido.

↳ XSD:

```
<xs:element name="Direccion">
  <xs:complexType>
    <name="calle" type="xs:string" />
    <name="poblacion" type="xs:string" />
    <name="provincia" type="xs:string" />
    <name="codigo_postal" type="xs:int" />

  </xs:complexType> <br /></xs:element><br /><br />o lo que es lo mismo:<br />
<xs:element name="Direccion">
  <xs:complexType><br />      <xs:complexContent>
    <name="calle" type="xs:string" />
    <name="poblacion" type="xs:string" />
    <name="provincia" type="xs:string" />
    <name="codigo_postal" type="xs:int" />

  <xs:complexContent>
</xs:complexType> <br /></xs:element>
```

● XML válido:

```
<Direccion><br />  <calle>Lago de Enol, nº 32</calle><br />  <poblacion>Aguadu
<br />
```

● XML NO válido:

```
<Direccion><br />  <poblacion>Aguadulce</poblacion><br />  <provincia>Almería<
```

↳ Alternativa, xs:choice, representa alternativas, hay que tener en cuenta que es una o exclusiva. Especifica una lista concreta de elementos de los que sólo puede aparecer

uno.

- XSD:

```
<xs:element name="FormaPago">
  <xs:complexType>

    name="paypal" type="xs:string" />
    name="transferencia" type="xs:string" />
    name="bizum" type="xs:string" />
    name="contado" type="xs:int" />

  </xs:complexType> <br /></xs:element>
```

- XML válido:

```
<FormaPago>transferencia</FormaPago>
```

- ◆ Secuencias no ordenadas, xs:all, representa a todos los elementos que componen el elemento de tipo compuesto en cualquier orden. Pueden aparecer en el documento xml en cualquier orden y dicho documento es válido.
- ◆ Contenido mixto, definido dando valor true al atributo mixed del elemento <xs:complexType mixed="true">, permite mezclar texto con elementos hijo. Los elementos hijo se definen con las opciones anteriores; xs:sequence, xs:choice o xs:all.
- ◆ Elemento vacío, el elemento no puede contener texto ni otros subelementos; únicamente atributos.

- XSD:

```
<xs:element>
  <xs:complexType>
    <xs:attribute name='codigo' type='xs:integer' />
  </xs:complexType>
</xs:element>
```

- XML válido:

```
<asignatura codigo='MAT-1920'><b>(vacío)</b></asignatura>

<strong><i>0 lo que es lo mismo y más correcto:</i></strong>

<asignatura codigo='MAT-1920' />
```

- ◆ Elemento simple con atributo, es un elemento simple porque no contiene a otros elementos, sólo datos, pero es de tipo complejo (complexType) porque tiene un atributo.

- XSD:

```
<xs:element name="cuota">
  <xs:complexType><br />          <xs:simpleContent>
    <xs:attribute name='moneda' type='xs:string' />
  </xs:simpleContent>  <br />  </xs:complexType>
</xs:element>
```

- XML válido:

```
<cuota moneda='Euro'><b>1200</b></cuota>
```

3.7.- Asociación con documentos XML.

Una vez que tenemos creado el fichero XSD ¿cómo lo asociamos a un fichero XML?

El modo de asociar un esquema a un documento XML es un espacio de nombres al ejemplar del documento, donde se indica la ruta de localización de los ficheros esquema mediante su URI, precedida del prefijo "xs:".

Para que un documento XML siga las reglas definidas en un esquema, no disponemos de etiqueta !DOCTYPE; en su lugar utilizamos atributos especiales en el elemento raíz del documento XML.

Primero, al igual que en el documento XMLSchema, necesitamos definir los dos espacios de nombres, el correspondiente al documento XML (que se suele usar sin abreviatura, es decir como espacio por defecto) y el espacio de nombres de XMLSchema (que suele utilizar el prefijo xs, aunque se puede utilizar otro).

Además es necesario indicar dónde está el archivo XMLSchema que contiene las reglas de validación que se aplican al documento. Esto se hace gracias al atributo llamado `schemaLocation` (perteneciente al espacio de nombres del esquema, por lo que se usa normalmente como `xs:schemaLocation`).

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<documento xmlns="http://www.iesaguadulce.es/doc" xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:schemaLocation="esquema.xsd">
.....
</documento>
```

Se indica el espacio por defecto de nombres en el documento (coincide con el declarado en el propio archivo del esquema), se indica el espacio de nombres correspondiente al esquema (siempre es la misma dirección de Internet) y se asocia a este espacio el prefijo xs (se puede elegir otro prefijo, pero no es nada conveniente).

En el caso de que en el Schema únicamente definamos el espacio de nombres `xmlns:xs="http://www.w3.org/2001/XMLSchema"`, en el documento xml habrá que indicar:

```
<?xml version="1.0" encoding="UTF-8"?>
<documento xmlns:xs="http://w3.org/2001/XMLSchema-instance">
<span>xs:noNamespaceSchemaLocation</span>="esquema.xsd">
...
</documento>
```

Ejercicio resuelto

Un documento XML asociado al esquema que se ha realizado anteriormente para estructurar la información personal sobre los alumnos de un centro educativo (apartado 3.1) puede ser:

[Mostrar retroalimentación](#)

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

<alumnos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:SchemaLocat

    <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>
```



Autoevaluación

La asociación de un documento XML a un esquema se hace en:

- El prólogo del documento XML.**
- La definición de tipo de datos.**
- El ejemplar.**
- En una sección llamada declaración del esquema que se sitúa entre el prólogo y el ejemplar.**

No es correcta porque el prólogo de un documento XML define la versión, el código y la independencia del documento para su interpretación.

Incorrecta, porque este apartado sirve para incorporar los DTD.

Muy bien, correcto.

Incorrecto, este apartado no existe.

Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta
4. Incorrecto

3.8.- Documentación del esquema.

Una vez que hemos visto como crear un esquema vamos a ver el modo de incorporar cierta documentación (quién es el autor, limitaciones de derechos de autor, utilidad del esquema, etc.) al mismo.

Podemos pensar que un método para añadir esta información es utilizar comentarios. El problema es que los analizadores no garantizan que los comentarios no se modifiquen al procesar los documentos y por tanto, que los datos añadidos no se pierdan en algún proceso de transformación del documento.

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.



[soyignatius. Documentación](#)
(CC BY-NC-ND)

Ejercicio resuelto

Ejemplo de documentación de un esquema.

[Mostrar retroalimentación](#)

```
<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.<
            <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 t
            <xs:appinfo>
        </xs:annotation>
    </xs:element>
</xs:schema>
```

Autoevaluación

La mejor solución para documentar un esquema es usar los comentarios:

- Verdadero.**
- Falso.**

No es correcta. Aunque pueden usarse, no es la mejor opción porque no se puede garantizar que no se pierdan en alguna transformación del fichero.

Genial, correcto.

Solución

- 1. Incorrecto**
- 2. Opción correcta**

4.- Herramientas de creación y validación.

Caso práctico

Antes de comenzar a trabajar con la normalización de los documentos que utiliza la empresa de María y Félix. Marina le presenta a Juan un informe sobre las diferentes herramientas que pueden facilitarles el trabajo de edición y validación de los documentos XSD y XML.

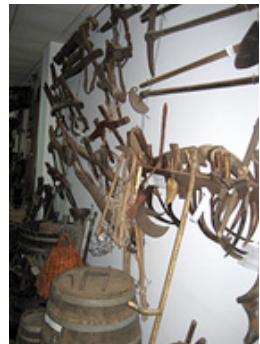
Juan hará un estudio de costes y escogerá alguna de ellas para realizar el trabajo.



[Jonny Goldstein. Juan \(CC BY\)](#)

Igual que hasta ahora, para crear y validar los documentos XML y los esquemas, basta con un editor de texto plano y un navegador. ¿Pero, no hay ninguna herramienta que nos facilite el trabajo? Pues sí, existen aplicaciones que permiten al usuario visualizar, validar y editar documentos en el lenguaje XML. Algunos de estos productos de escritorio son:

- ✓ Notepad ++ (Gratis)
- ✓ Editix XML Editor (Gratis)
- ✓ XML Copy Editor (Gratis)
- ✓ NetBeans (Gratis)



[xornalcerto. Herramientas \(CC BY\)](#)

También tenemos herramientas de validación online, a continuación algunas web que nos validan un xml, dtd y schema:

- ✓ <http://www.xmlvalidation.com>
- ✓ <http://www.utilities-online.info/xsdvalidation/>

5.- Ejercicios resueltos.

Caso práctico

La empresa Reggio tiene establecimientos por toda Italia, pero su sede central está en Cesena, al igual que el almacén donde se distribuye a todos los demás establecimientos. Esta empresa distribuye alpiste para pájaros, así como otros artículos ornitológicos.

Cada establecimiento tiene una tienda, así como un almacén, que pueden o no estar en la misma ubicación. Cuando se hace un pedido a la fábrica por parte de los establecimientos, éstas reciben los artículos en su almacén, y la documentación (albarán y pago) se remite a la tienda.

Cada pedido debe tener los datos siguientes:

- ✓ Datos del establecimiento que realiza el pedido (Nombre, dirección para envío y dirección almacén (si es la misma, sólo aparecerá una)).
- ✓ Código de pedido (Cadena de caracteres formada por: Código establecimiento (1 letra y 2 números), número pedido (4 números), un guión y Año (dos números), por ejemplo: E180021-17)
- ✓ Nombre del empleado que realiza el pedido.
- ✓ Fecha de pedido.
- ✓ Tipo de envío (cuyos valores son: Normal o Urgente)

Respecto a los artículos del pedido, se guardarán los siguientes datos:

- ✓ Código del artículo (formado por tres letras y 3 números, por ejemplo: ZZZ134).
- ✓ Número de unidades.
- ✓ Precio por unidad.
- ✓ Observaciones.

Ejercicio "Reggio" Resuelto

Crear un DTD que cumpla las especificaciones del caso práctico

[Mostrar Retroalimentación](#)

```
<!ELEMENT pedido (establecimiento,empleado,fecha_pedido,articulos)>
  <!ATTLIST pedido cod_pedido CDATA #REQUIRED>
  <!ATTLIST pedido tipo (normal|urgente) #REQUIRED>
<!ELEMENT establecimiento (nom_estab,direccion_envio,direccion_almacen?)>
  <!ELEMENT nom_estab (#PCDATA)>
  <!ELEMENT direccion_envio (via,numero,localidad,provincia,cp)>
  <!ELEMENT direccion_almacen (via,numero,localidad,provincia,cp)>
    <!ELEMENT via (#PCDATA)>
    <!ELEMENT numero (#PCDATA)>
    <!ELEMENT localidad (#PCDATA)>
```

```
<!ELEMENT provincia (#PCDATA)>
<!ELEMENT cp (#PCDATA)>
<!ELEMENT empleado (#PCDATA)>
<!ELEMENT fecha_pedido (#PCDATA)>
<!ELEMENT articulos (articulo+)>
<!ELEMENT articulo (unidades,precio,observaciones?)>
    <!ATTLIST articulo cod_articulo CDATA #REQUIRED>
<!ELEMENT unidades (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
    <!ATTLIST precio moneda CDATA #REQUIRED>
<!ELEMENT observaciones (#PCDATA)>
```

Crear un XML que valide el DTD anterior

[Mostrar retroalimentación](#)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pedido SYSTEM "tarea04_01.dtd">
<pedido cod_pedido="E180021-17" tipo="urgente">
    <establecimiento>
        <nomb_estab>Manitoba</nomb_estab>
        <direccion_envio>
            <via>Vittorio</via>
            <numero>1</numero>
            <localidad>Cesena</localidad>
            <provincia>Cesena</provincia>
            <cp>00400</cp>
        </direccion_envio>
        <direccion_almacen>
            <via>Vittorio</via>
            <numero>5</numero>
            <localidad>Cesena</localidad>
            <provincia>Cesena</provincia>
            <cp>00400</cp>
        </direccion_almacen>
    </establecimiento>
    <empleado>Roberto Rossini</empleado>
    <fecha_pedido>2017-05-01</fecha_pedido>
    <articulos>
        <articulo cod_articulo="ZZZ134">
            <unidades>1</unidades>
            <precio moneda="EUR">1.05</precio>
            <observaciones>Tenere lontano dalla portata de
        </articulo>
        <articulo cod_articulo="ZZZ137">
            <unidades>2</unidades>
            <precio moneda="EUR">3.50</precio>
        </articulo>
        <articulo cod_articulo="ZZZ139">
            <unidades>5</unidades>
            <precio moneda="EUR">5.50</precio>
        </articulo>
    </articulos>
</pedido>
```

Crear un XML Schema que cumpla las especificaciones del caso práctico

Mostrar retroalimentación

```

<?xml version="1.0" encoding="UTF-8"?>
<xsschema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qua
    <!-- Documentacion del esquema -->
    <xss:annotation>
        <xss:documentation xml:lang="es-es">
            <titulo>Reggio</titulo>
            <modulo>Lenguajes de marcas y sistemas de gestión de i
            <fecha_creacion>2017</fecha_creacion>
            <autor>Profesor de LMSGI </autor>
        </xss:documentation>
    </xss:annotation>
    <!-- Definicion del vocabulario -->
    <xss:element name="pedido">
        <xss:complexType>
            <xss:sequence>
                <xss:element ref="establecimiento"/>
                <xss:element ref="empleado"/>
                <xss:element ref="fecha_pedido"/>
                <xss:element ref="articulos"/>
            </xss:sequence>
            <xss:attributeGroup ref="attlist.pedido"/>
        </xss:complexType>
    </xss:element>
    <xss:attributeGroup name="attlist.pedido">
        <xss:attribute name="cod_pedido" type="cod_ped">
            <xss:simpleType>
                <xss:restriction base="xs:string">
                    <xss:enumeration value=
                    <xss:enumeration value=
                </xss:restriction>
            </xss:simpleType>
        </xss:attribute>
    </xss:attributeGroup>
    <xss:simpleType name="cod_ped">
        <xss:annotation>
            <xss:documentation>
                El código del pedido tendrá la
            </xss:documentation>
        </xss:annotation>
        <xss:restriction base="xs:string">
            <xss:length value="10"/>
            <xss:pattern value="[A-Za-z][0-9]{2}[0-
        </xss:restriction>
    </xss:simpleType>
<xss:element name="establecimiento">
    <xss:complexType>
        <xss:sequence>
            <xss:element ref="nombr_estab"/>

```

```

                <xs:element ref="direccion_envio"/>
                <xs:element ref="direccion_almacen"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
<xs:element name="nomb_estab" type="xs:string" />
<xs:element name="direccion_envio">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="via"/>
            <xs:element ref="numero"/>
            <xs:element ref="localidad"/>
            <xs:element ref="provincia"/>
            <xs:element ref="cp"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="via" type="xs:string" />
<xs:element name="numero" type="xs:string" />
<xs:element name="localidad" type="xs:string" />
<xs:element name="provincia" type="xs:string" />
<xs:element name="cp">
    <xs:annotation>
        <xs:documentation>El código postal está formado por 5</xs:documentation>
    </xs:annotation>
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:length value="5" />
            <xs:pattern value="[0-9]{5}"></xs:pattern>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="direccion_almacen">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="via"/>
            <xs:element ref="numero"/>
            <xs:element ref="localidad"/>
            <xs:element ref="provincia"/>
            <xs:element ref="cp"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="empleado" type="xs:string" />
<xs:element name="fecha_pedido" type="xs:date">
    <xs:annotation>
        <xs:documentation>
            El formato de entrada de la fecha es AAAA-MM-D
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="articulos">
    <xs:complexType>
        <xs:sequence>
            <xs:element maxOccurs="unbounded" ref="articulo"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

```

```

<xs:element name="articulo">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="unidades"/>
            <xs:element ref="precio"/>
            <xs:element ref="observaciones" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>
                        El campo observaciones es opcional
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
        <xs:attributeGroup ref="attlist.articulo"/>
    </xs:complexType>
</xs:element>
<xs:attributeGroup name="attlist.articulo">
    <xs:attribute name="cod_articulo" type="cod_articulo" use="req">
</xs:attributeGroup>
<xs:simpleType name="cod_articulo">
    <xs:annotation>
        <xs:documentation>
            El código de articulo está formado por 3 letra
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:length value="6"/>
        <xs:pattern value="[A-Za-z]{3}[0-9]{3}"/>
    </xs:restriction>
</xs:simpleType>
<xs:element name="unidades" type="xs:positiveInteger"/>
<xs:element name="precio">
    <xs:annotation>
        <xs:documentation>
            Pasamos de un tipo simple a uno complejo
        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:decimal">
                <xs:attributeGroup ref="attlist.precio" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:attributeGroup name="attlist.precio">
    <xs:attribute name="moneda" type="xs:NMTOKEN" fixed="EUR" use="require">
</xs:attributeGroup>
<xs:element name="observaciones" type="xs:string"/>
</xs:schema>

```

Crear un XML que valide el XML Schema anterior

[Mostrar retroalimentación](#)

```

<?xml version="1.0" encoding="UTF-8"?>
<pedido xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xsi:noNamespaces
          <establecimiento>
            <nomb_estab>Manitoba</nomb_estab>
            <direccion_envio>
              <via>Vittorio</via>
              <numero>1</numero>
              <localidad>Cesena</localidad>
              <provincia>Cesena</provincia>
              <cp>00400</cp>
            </direccion_envio>
            <direccion_almacen>
              <via>Vittorio</via>
              <numero>5</numero>
              <localidad>Cesena</localidad>
              <provincia>Cesena</provincia>
              <cp>00400</cp>
            </direccion_almacen>
          </establecimiento>
          <empleado>Roberto Rossini</empleado>
          <fecha_pedido>2017-05-01</fecha_pedido>
          <articulos>
            <articulo cod_articulo="ZZZ134">
              <unidades>1</unidades>
              <precio moneda="EUR">1.05</precio>
              <observaciones>Tenere lontano dalla portata de
            </articulo>
            <articulo cod_articulo="ZZZ137">
              <unidades>2</unidades>
              <precio moneda="EUR">3.50</precio>
            </articulo>
            <articulo cod_articulo="ZZZ139">
              <unidades>5</unidades>
              <precio moneda="EUR">5.50</precio>
            </articulo>
          </articulos>
        </pedido>

```

En los siguientes vídeos tenéis las explicaciones de este ejercicio.

Explicación DTD y xml

<http://www.youtube.com/embed/GGK5MCndyj8>

[Resumen textual alternativo](#)

Explicación xsd y xml

<http://www.youtube.com/embed/ZF7xSle8QPU>

[Resumen textual alternativo](#)

Caso práctico

En Almería existe una gran pasión por el cine, lo que genera toda una industria a su alrededor. La empresa **Cine de Almería SL** se dedica a gestionar todo lo relacionado con el cine en la provincia y su exportación al exterior. **Cine de Almería** tiene sucursales repartidas por el territorio español que pueden dedicarse a la gestión de la **producción de películas** o a su **distribución**, pero no a ambas cosas.

Las **sucursales de producción** tienen un espacio físico y una web desde donde se puede gestionar todo lo relacionado con la producción de películas de manera indistinta. Se encargan de gestionar todo lo relacionado con el atrezo para las películas y el vestuario.

Las **sucursales de distribución**, que abarcan una comunidad autónoma, pueden tener una exposición física o una web, pero no las dos cosas. Estas sucursales se encargan únicamente de la publicidad y distribución de la filmografía.

Los pedidos se realizan a las sucursales de producción o distribución dependiendo del tipo de pedido y se reciben en la propia sucursal etiquetados con la dirección del cliente que lo realiza, así como toda la documentación, y será la propia sucursal la que los distribuya oportunamente.

Un **pedido** incluye unos datos generales de la sucursal de producción o de distribución según el caso y los datos específicos del cliente de ese pedido, así como los productos o servicios que solicita. Cada **pedido** debe tener los siguientes **datos generales**:

- **Código de pedido.** Cadena de caracteres formada por: Código (será PRO para sucursal de producción o DIS para sucursal de distribución, seguido de un número de 2 cifras que indica el número de la sucursal, un guión, año (2 números), un guión y número pedido (4 números), por ejemplo: PRO01-19-0001)
- **Datos de quién recibe el pedido para realizarlo (operador):** Razón social de la sucursal de producción o distribución, teléfono, dirección física y dirección web en caso de tenerla.
- **Nombre completo y NIF del responsable** que realiza el pedido.
- **Fecha** de pedido.
- **Importe total del pedido.**
- **Forma de pago.** Los valores pueden ser: Pay Pal, transferencia bancaria o contrareembolso.
- **Método de envío.** Los valores pueden ser: Estándar (48 horas) o urgente (12 horas).
- **Fecha tope de entrega:** Será una fecha 30 días posterior a la solicitud del pedido.
- **Observaciones** sobre el pedido. (Dato opcional)

Respecto a los **datos de los clientes** para el **pedido**, se guardarán los siguientes, todos ellos obligatorios:

- **Código de cliente.** (2 letras, iniciales del nombre y primer apellido, y 3 números, por ejemplo: GH128)
- **Nombre completo.**
- **Dirección completa de envío.**
- **Importe del pedido.** (Se guardarán la Base imponible, IVA-21% y total)

Respecto a **productos de cada pedido de cada cliente**, se guardarán los siguientes datos, todos ellos obligatorios:

- **Código de producto.** (Mínimo de 3 caracteres y máximo de 8, pueden ser letras y/o números, por ejemplo: G128)
- **Número de unidades.**
- **Precio por unidad.** (IVA no incluido)
- **Descripción.**

Ejercicio "Cine de Almería SL" Resuelto

Crear el vocabulario para el documento XML que utiliza Cine de Almería SL para gestionar los pedidos utilizando un DTD externo.

[Mostrar retroalimentación](#)

```
<!ELEMENT pedido (operador, clientes)>
  <!ATTLIST pedido codigo_pedido CDATA #REQUIRED> <br />
  <!ELEMENT operador (sucursal, responsable, fecha, precio_total, forma_pago, metodo_envio, observaciones)>
    <!ELEMENT sucursal (tipo, razon_social, telefono, direccion, web)>
      <!ELEMENT tipo (#PCDATA)>
      <!ELEMENT razon_social (#PCDATA)>
      <!ELEMENT telefono (#PCDATA)>
        <!ELEMENT direccion (via, localidad, provincia, cp, pais)>
          <!ELEMENT via (#PCDATA)>
        <!ELEMENT localidad (#PCDATA)>
          <!ELEMENT provincia (#PCDATA)>
        <!ELEMENT cp (#PCDATA)>
          <!ELEMENT pais (#PCDATA)>
        <!ELEMENT web (#PCDATA)> <br />
    <!ELEMENT responsable (nombre, nif)>
      <!ELEMENT nombre (#PCDATA)>
      <!ELEMENT nif (#PCDATA)> <br />
    <!ELEMENT fecha (#PCDATA)>
    <!ELEMENT precio_total (#PCDATA)>
      <!ATTLIST precio_total moneda CDATA #REQUIRED>
    <!ELEMENT forma_pago (#PCDATA)>
    <!ELEMENT metodo_envio (#PCDATA)>
    <!ELEMENT fecha_tope_entrega (#PCDATA)>
      <!ATTLIST fecha_tope_entrega unidad CDATA #REQUIRED>
    <!ELEMENT observaciones (#PCDATA)> <br />
  <!ELEMENT clientes (cliente+)>
    <!ELEMENT cliente (nombre_cliente, direccion_cliente, productos)>
      <!ATTLIST cliente codigo_cliente CDATA #REQUIRED>
      <!ELEMENT nombre_cliente (#PCDATA)>
      <!ELEMENT direccion_cliente (via, localidad, provincia)>
      <!ELEMENT productos (producto+)>
        <!ELEMENT producto (unidades, precio, descripcion)>
        <!ATTLIST producto cod_producto CDATA #REQUIRED>
          <!ELEMENT unidades (#PCDATA)>
          <!ELEMENT precio (#PCDATA)>
            <!ATTLIST precio moneda CDATA #REQUIRED>
            <!ELEMENT descripcion (#PCDATA)>
        <!ELEMENT factura (base, iva, total, forma_pago)>
          <!ELEMENT base (#PCDATA)>
            <!ATTLIST base moneda CDATA #REQUIRED>
          <!ELEMENT iva (#PCDATA)>
            <!ATTLIST iva tipo CDATA #REQUIRED>
            <!ATTLIST iva moneda CDATA #REQUIRED>
```

```
<!ELEMENT total (#PCDATA)>
<!ATTLIST total moneda CDATA #
```

Crear ese mismo vocabulario utilizando el lenguaje XML Schema debidamente documentado con fecha de creación, nombre del autor y utilidad del esquema.

[Mostrar retroalimentación](#)

```
<?xml version="1.0" encoding="UTF-8"?>
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema" elementFormDefault="qua
    <!-- Documentacion del esquema -->
    <xss:annotation>
        <xss:documentation xml:lang="es-es">
            <titulo>APC</titulo>
            <modulo>Lenguajes de marcas y sistemas de gestión de i
            <fecha_creacion>2019</fecha_creacion>
            <autor>Profesorado de LMSGI </autor>
        </xss:documentation>
    </xss:annotation>
    <!-- Definicion del vocabulario -->
    <xss:element name="pedido">
        <xss:complexType>
            <xss:sequence>
                <xss:element ref="operador"/>
                <xss:element ref="cliente"/>
            </xss:sequence>
            <xss:attributeGroup ref="attlist.pedido"/>
        </xss:complexType>
    </xss:element>
    <xss:attributeGroup name="attlist.pedido">
        <xss:attribute name="codigo_pedido" type="cod_ped" use="require
        <xss:attribute name="forma_pago" type=" pago" use="required"/>
    </xss:attributeGroup>
    <xss:simpleType name="cod_ped">
        <xss:annotation>
            <xss:documentation>
                El código del pedido tendrá la siguiente estru
            </xss:documentation>
        </xss:annotation>
        <xss:restriction base="xs:string">
            <xss:length value="13"/>
            <xss:pattern value="((PRO)|(DIS))[0-9]{2}-[0-9]{2}-[0-9
        </xss:restriction>
    </xss:simpleType>
    <xss:simpleType name=" pago">
        <xss:annotation>
            <xss:documentation>
                La forma de pago se elegirá entre Paypal, tras
            </xss:documentation>
        </xss:annotation>
        <xss:restriction base="xs:string">
            <xss:enumeration value="PayPal" />
            <xss:enumeration value="trasferencia bancaria" />
            <xss:enumeration value="contrareembolso" />
        </xss:restriction>
    </xss:simpleType>
</xss:attributeGroup>
```

```

        </xs:restriction>
    </xs:simpleType>
    <xs:element name="operador">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="oficina"/>
                <xs:element ref="responsable"/>
                <xs:element ref="fecha_pedido"/>
                <xs:element ref="productos"/>
                <xs:element ref="importe_pedido"/>
                <xs:element ref="fecha_tope_entrega"/>
                <xs:element ref="observaciones" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>
                            El campo observaciones es opcional
                        </xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="oficina">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="tipo" />
                <xs:element ref="razon_social" />
                <xs:element ref="telefono" />
                <xs:element ref="direccion_oficina" />
                <xs:element ref="direccion_plato" minOccurs="0"/>
                <xs:element ref="direccion_almacen" minOccurs="0"/>
                <xs:element ref="web" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>
                            La dirección web puede aparecer o no si la oficina
                        </xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="tipo">
        <xs:annotation>
            <xs:documentation>
                Puede ser de tipo producción o distribución.
            </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="producción"/>
                <xs:enumeration value="distribución"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>

    <xs:element name="razon_social" type="xs:string" />
    <xs:element name="telefono">

```

```

<xs:annotation>
    <xs:documentation>
        El teléfono está compuesto de 9 dígitos.
    </xs:documentation>
</xs:annotation>
<xs:simpleType>
    <xs:restriction base="xs:string">
        <xs:length value="9" />
        <xs:pattern value="[0-9]{9}" />
    </xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="direccion_oficina">
<xs:complexType>
    <xs:sequence>
        <xs:element ref="via" />
        <xs:element ref="localidad" />
        <xs:element ref="provincia" />
        <xs:element ref="cp" />
        <xs:element ref="pais" />
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="direccion_plato">
<xs:complexType>
    <xs:sequence>
        <xs:element ref="via" />
        <xs:element ref="localidad" />
        <xs:element ref="provincia" />
        <xs:element ref="cp" />
        <xs:element ref="pais" />
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="direccion_almacen">
<xs:complexType>
    <xs:sequence>
        <xs:element ref="via" />
        <xs:element ref="localidad" />
        <xs:element ref="provincia" />
        <xs:element ref="cp" />
        <xs:element ref="pais" />
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="via" type="xs:string" />
<xs:element name="localidad" type="xs:string" />
<xs:element name="provincia" type="xs:string" />
<xs:element name="cp">
    <xs:annotation>
        <xs:documentation>
            El código postal está formado por 5 dígitos.
        </xs:documentation>
    </xs:annotation>
<xs:simpleType>
    <xs:restriction base="xs:string">
        <xs:length value="5" />
        <xs:pattern value="[0-9]{5}" />
    </xs:restriction>
</xs:simpleType>

```

```

        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="pais" type="xs:string" />
<xs:element name="web" type="xs:string" />
<xs:element name="responsable">
<xs:complexType>
    <xs:sequence>
        <xs:element ref="nombre_resp" />
        <xs:element ref="nif_responsable" />
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="nombre_resp">
<xs:complexType>
    <xs:sequence>
        <xs:element name="nombre" type="xs:string" />
        <xs:element name="primer_apellido" type="xs:string" />
        <xs:element name="segundo_apellido" type="xs:string" />
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="nif_responsable">
<xs:annotation>
    <xs:documentation>
        El nif está formado por 8 números y una letra.
    </xs:documentation>
</xs:annotation>
<xs:simpleType>
    <xs:restriction base="xs:string">
        <xs:length value="9" />
        <xs:pattern value="[0-9]{8}[A-Za-z]" />
    </xs:restriction>
</xs:simpleType>
</xs:element>

<xs:element name="fecha_pedido" type="xs:date">
<xs:annotation>
    <xs:documentation>
        El formato de entrada de la fecha es AAAA-MM-D
    </xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="productos">
<xs:complexType>
    <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="product">
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="producto">
<xs:complexType>
    <xs:sequence>
        <xs:element ref="unidades" />
        <xs:element ref="descripcion" />
        <xs:element ref="precio" />
    </xs:sequence>

```

```

                <xs:attributeGroup ref="attlist.producto"/>
            </xs:complexType>
        </xs:element>
<xs:attributeGroup name="attlist.producto">
    <xs:attribute name="cod_producto" type="cod_pro" use="required"
</xs:attributeGroup>
<xs:simpleType name="cod_pro">
    <xs:annotation>
        <xs:documentation>
            Mínimo de 3 caracteres y máximo de 8, pueden s
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:normalizedString">
        <xsmaxLength value="8"/>
        <xs:minLength value="3"/>
    </xs:restriction>
</xs:simpleType>
<xs:element name="unidades" type="xs:positiveInteger" />
<xs:element name="descripcion" type="xs:string" />
<xs:element name="precio">
    <xs:annotation>
        <xs:documentation>
            Pasamos de un tipo simple a uno complejo
        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:decimal">
                <xs:attributeGroup ref="attlist.precio" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:attributeGroup name="attlist.precio">
    <xs:attribute name="moneda" type="xs:NMTOKEN" fixed="EUR" use="require
</xs:attributeGroup>

<xs:element name="importe_pedido">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="base" />
            <xs:element ref="iva" />
            <xs:element ref="total" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="base">
    <xs:annotation>
        <xs:documentation>
            Pasamos de un tipo simple a uno complejo
        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:decimal">
                <xs:attributeGroup ref="attlist.base" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>

```

```

</xs:complexType>
</xs:element>
<xs:attributeGroup name="atlist.base">
    <xs:attribute name="moneda" type="xs:NMTOKEN" fixed="EUR" use="require">
</xs:attributeGroup>
<xs:element name="iva">
<xs:annotation>
    <xs:documentation>
        Pasamos de un tipo simple a uno complejo
    </xs:documentation>
</xs:annotation>
<xs:complexType>
    <xs:simpleContent>
        <xs:extension base="xs:decimal">
            <xs:attributeGroup ref="atlist.iva" />
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
</xs:element>
<xs:attributeGroup name="atlist.iva">
    <xs:attribute name="tipo" type="xs:NMTOKEN" fixed="21" use="required">
        <xs:attribute name="moneda" type="xs:NMTOKEN" fixed="EUR" use="require">
</xs:attributeGroup>
<xs:element name="total">
<xs:annotation>
    <xs:documentation>
        Pasamos de un tipo simple a uno complejo
    </xs:documentation>
</xs:annotation>
<xs:complexType>
    <xs:simpleContent>
        <xs:extension base="xs:decimal">
            <xs:attributeGroup ref="atlist.total" />
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
</xs:element>
<xs:attributeGroup name="atlist.total">
    <xs:attribute name="moneda" type="xs:NMTOKEN" fixed="EUR" use="require">
</xs:attributeGroup>

<xs:element name="fecha_tope_entrega" type="xs:date">
    <xs:annotation>
        <xs:documentation>
            El formato de entrada de la fecha es AAAA-MM-DD
        </xs:documentation>
    </xs:annotation>
</xs:element>

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

<xs:element name="cliente">
<xs:complexType>
    <xs:sequence>
        <xs:element ref="nombre_cliente" />
        <xs:element ref="nif" />

```

```

        <xs:element ref="direccion_cliente" />
    </xs:sequence>
    <xs:attributeGroup ref="attlist.cliente"/>
</xs:complexType>
</xs:element>
<xs:attributeGroup name="attlist.cliente">
    <xs:attribute name="codigo_cliente" type="cod_cli" use="requir
</xs:attributeGroup>
<xs:simpleType name="cod_cli">
    <xs:annotation>
        <xs:documentation>
            El código de cliente está formado por dos letr
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:length value="5"/>
        <xs:pattern value="[A-Za-z][A-Za-z][0-9]{3}"/>
    </xs:restriction>
</xs:simpleType>
<xs:element name="nombre_cliente">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="nombre" type="xs:string" />
            <xs:element name="primer_apellido" type="xs:string" />
            <xs:element name="segundo_apellido" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="nif">
    <xs:annotation>
        <xs:documentation>
            El nif está formado por 8 números y una letra.
        </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:length value="9" />
            <xs:pattern value="[0-9]{8}[A-Za-z]" />
        </xs:restriction>
    </xs:simpleType>
</xs:element>
<xs:element name="direccion_cliente">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="via" />
            <xs:element ref="localidad" />
            <xs:element ref="provincia" />
            <xs:element ref="cp" />
            <xs:element ref="pais" />
        </xs:sequence>
    </xs:complexType>
</xs:element>

</xs:schema>

```

Realizar un fichero XML que se corresponda con una instancia del vocabulario diseñado y asociarle el DTD considerando que dicho DTD se llama "tarea4_01_Sol.dtd".

[Mostrar retroalimentación](#)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE pedido SYSTEM "tarea4_01_Sol.dtd">
<pedido codigo_pedido="PR001-19-0001" forma_pago="transferencia_bancaria">
  <operador>
    <oficina tipo="produccion">
      <razon_social>Producciones Costa SL</razon_social>
      <telefono>666555444</telefono>
      <direccion_oficina>
        <via>Calle Azul nº 4</via>
        <localidad>Aguadulce</localidad>
        <provincia>Almería</provincia>
        <cp>04720</cp>
        <pais>España</pais>
      </direccion_oficina>
      <direccion_plato>
        <via>Calle Azul nº 6</via>
        <localidad>Aguadulce</localidad>
        <provincia>Almería</provincia>
        <cp>04720</cp>
        <pais>España</pais>
      </direccion_plato>
    </oficina>
    <responsable>
      <nombre_resp>
        <nombre>Pedro</nombre>
        <primer_apellido>López</primer_apellido>
        <segundo_apellido>Martínez</segundo_apellido>
      </nombre_resp>
      <nif_responsable>12345678A</nif_responsable>
    </responsable>
    <fecha_pedido>2018-12-01</fecha_pedido>
    <productos>
      <producto cod_producto="JARR02">
        <unidades>1</unidades>
        <descripcion>Jarrón chino</descripcion>
        <precio moneda="EUR">34</precio>
      </producto>
    </productos>
    <importe_pedido>
      <base moneda="EUR">34</base>
      <iva tipo="21" moneda="EUR">7.14</iva>
      <total moneda="EUR">41.14</total>
    </importe_pedido>
    <fecha_tope_entrega unidad="fecha">2019-01-30</fecha_tope_entrega>
    <observaciones>Pedido frágil</observaciones>
  </operador>
  <cliente codigo_cliente="GH128">
    <nombre_cliente>
      <nombre>Gloria</nombre>
      <primer_apellido>Hortal</primer_apellido>
      <segundo_apellido>Giocci</segundo_apellido>
    </nombre_cliente>
```

```

</nombre_cliente>
<NIF>12457845S</NIF>
<direccion_cliente>
  <via>Cerezuela, 1</via>
  <localidad>Roquetas de Mar</localidad>
  <provincia>Almería</provincia>
  <cp>04740</cp>
  <pais>España</pais>
</direccion_cliente>
</cliente>
</pedido>

```

Modificar ese fichero XML para asociarle el esquema diseñado y cuyo fichero se llama "tarea4_02_Sol.xsd".

[Mostrar retroalimentación](#)

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE pedido>
<pedido xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" codigo_pedido="P
  <operador>
    <oficina>
      <tipo>producción</tipo>
      <razon_social>Producciones Costa SL</razon_social>
      <telefono>666555444</telefono>
      <direccion_oficina>
        <via>Calle Azul nº 4</via>
        <localidad>Aguadulce</localidad>
        <provincia>Almería</provincia>
        <cp>04720</cp>
        <pais>España</pais>
      </direccion_oficina>
      <direccion_plato>
        <via>Calle Azul nº 6</via>
        <localidad>Aguadulce</localidad>
        <provincia>Almería</provincia>
        <cp>04720</cp>
        <pais>España</pais>
      </direccion_plato>
    </oficina>
    <responsable>
      <nombre_resp>
        <nombre>Pedro</nombre>
        <primer_apellido>López</primer_apellido>
        <segundo_apellido>Martínez</segundo_apellido>
      </nombre_resp>
      <nif_responsable>12345678A</nif_responsable>
    </responsable>
    <fecha_pedido>2018-12-01</fecha_pedido>
    <productos>
      <producto cod_producto="JARR02">
        <unidades>1</unidades>
        <descripcion>Jarrón chino</descripcion>
        <precio moneda="EUR">34</precio>
      </producto>
    </productos>
  </operador>
</pedido>

```

```
</productos>
<importe_pedido>
    <base moneda="EUR">34</base>
    <iva tipo="21" moneda="EUR">7.14</iva>
    <total moneda="EUR">41.14</total>
</importe_pedido>
<fecha_tupe_entrega>2019-01-30</fecha_tupe_entrega>
<observaciones>Pedido frágil</observaciones>
</operador>
<cliente codigo_cliente="GH128">
    <nombre_cliente>
        <nombre>Gloria</nombre>
        <primer_apellido>Hortal</primer_apellido>
        <segundo_apellido>Giocci</segundo_apellido>
    </nombre_cliente>
    <nif>12457845S</nif>
    <direccion_cliente>
        <via>Cerezuela, 1</via>
        <localidad>Roquetas de Mar</localidad>
        <provincia>Almería</provincia>
        <cp>04740</cp>
        <pais>España</pais>
    </direccion_cliente>
</cliente>
</pedido>
```

Anexo I.- Documentación de apoyo.

- ✓ DTD: http://www.mclibre.org/consultar/xml/lecciones/xml_dtd.html
- ✓ Resumen DTD: [Descargar PDF \(56 KB\)](#)
- ✓ Ejercicios DTD: <http://www.mclibre.org/consultar/xml/ejercicios/dtd.html>
- ✓ DTD y XML Schema: [Descargar PDF \(1,1 MB\)](#)
- ✓ [Video] XML Schema (Estructura): http://www.youtube.com/watch?list=PLKLLGxMgKZ_QCcAMbq8L9dEXa6hMdb1q5&v=JKhfLpkVh3o&feature=player_detailpage
- ✓ [Video] Mi primer DTD: https://www.youtube.com/watch?feature=player_embedded&v=EfnWCeQNTQI
- ✓ [Video] Elementos DTD: https://www.youtube.com/watch?feature=player_embedded&v=ryoW-B_6cGs
- ✓ [Video] Creación DTD sencillo Parte I: https://www.youtube.com/watch?feature=player_embedded&v=fPU1ex7bSgg
- ✓ [Video] Creación DTD sencillo Parte II: https://www.youtube.com/watch?feature=player_embedded&v=4NB89iXyxMU
- ✓ [Descarga] Referencia rápida Tipos de Datos: [SchemaDataTypes \(41 KB\) - Datatype Hierarchy \(35 KB\)](#)

Condiciones y términos de uso de los materiales

Materiales desarrollados inicialmente por el Ministerio de Educación, Cultura y Deporte y actualizados por el profesorado de la Junta de Andalucía bajo licencia Creative Commons BY-NC-SA.



GOBIERNO
DE ESPAÑA
MINISTERIO
DE EDUCACIÓN
Y FORMACIÓN PROFESIONAL



Antes de cualquier uso leer detenidamente el siguiente [Aviso legal](#)

Historial de actualizaciones

Versión: 02.00.01

Fecha de actualización: 09/01/22

Actualización de materiales y correcciones menores.

Versión: 02.00.00

Fecha de actualización: 13/04/21

Autoría: Silvia Thomas Barros

Ubicación: Toda la unidad

Mejora (tipo 3): 1.- Añadir ejemplo. 1.1.- Modificaciones de errores existentes.

1.2.- Añadir ejemplos de espacios de nombres para su mejor comprensión.

2.1.- Modificaciones de errores existentes.

2.2.- Añadir ejemplos de tipos de elementos terminales y modificar el formato de los que hay para unificar la unidad.

2.4.- Añadir ejemplos de estructura de atributos y modificar todos los ejemplos existentes.

2.5.- Reestructurar el apartado para su correcto entendimiento y añadir ejemplos. Explicar correctamente las entidades.

2.6.- Modificar el formato de los ejemplos para unificar la unidad.

3.- Reestructurar el apartado completo para explicar el Schema adecuadamente y añadir ejemplos en esta página. Cambiado “Debes conocer” por “Para saber más” debido al exceso de información contenido en el “Debes conocer”

3.1.- Tipos de elementos en XML Schema Antes “Elementos del lenguaje”. Cambiarlo totalmente, añadir explicaciones de los tipos de elementos y ejemplos y cambiar el “Debes conocer” por “Para saber más”.

3.2.- Atributos en XML Schema Apartado nuevo. Explicar los atributos, sus tipos y ejemplos.

3.3.- Tipos de datos. Cambiar “Debes conocer” por “Para saber más”. Añadir ejemplos.

3.6.- Definición de tipos de datos complejos XML Schema. NUEVO Antes “Documentación del esquema”. Explicar los tipos de datos complejos, su estructura y añadir ejemplos.

4.- Herramientas de creación y validación. Corregir errores

5.- Ejercicio resuelto. Añadir ejercicio resuelto.

Ubicación: LMSGI04

Mejora (Mapa conceptual): Actualización mapa conceptual

Ubicación: LMSGI04

Mejora (Orientaciones del alumnado): Actualización del índice

Versión: 01.03.00

Fecha de actualización: 09/01/20

Autoría: Silvia Thomas Barros

Ubicación: 2.1, 2.2., 2.4, 2.5, 3, 3.1 y 4

Mejora (tipo 2):

Cambios menores en:

2.1.- Quitado SYSTEM del Debes conocer, estaba mal
2.2.- Añadidos ejemplos y código explicativo de elementos terminales.
2.4.- Atributos. Añadir ejemplos de distintas definiciones de atributos.
2.5. Entidades. Reestructurado el apartado añadiendo ejemplos de entidades y explicaciones.
3.- XML Schema.- añadir aclaraciones sobre los distintos prefijos posibles. Arreglado el ejercicio resuelto que estaba mal
3.1.- Reestructuración
4.- Eliminada la herramienta BaseX porque no tiene sentido en esta unidad.

Ubicación: 2.1

Mejora (tipo 1): En el apartado 2.1.- Declaración de la DTD, en el Debes Conocer pone:

Aunque es más correcto si se pone de forma pública, hacerlo así:

```
!DOCTYPE cine PUBLIC filmoteca SYSTEM http://cine.com/filmoteca.dtd>
```

La profesora me ha dicho que no debe de aparecer SYSTEM.

Versión: 01.02.00

Fecha de actualización: 10/06/17

Autoría: Víctor Javier Gómez Arques

Ubicación: Sección 2

Mejora (tipo 1): Inclusión de ejemplos correctos e incorrectos en las declaraciones de tipos.

Ubicación: Sección 4

Mejora (tipo 1): Actualización de herramientas de creación y validación e inclusión de herramientas online

Ubicación: Anexo I

Mejora (tipo 1): Actualización de enlaces con información adicional y eliminación de enlaces en inglés. Inclusión de resumen de DTD. Material teórico adicional de DTD y XML Schema. Enlaces a ejercicios de DTD.

Ubicación: Toda la unidad

Mejora (tipo 1): Los fragmentos de código no se visualizan correctamente. Hay que convertirlos a formato código.

Ubicación: Sección 1

Mejora (tipo 1): Actualizar licencia general de la unidad

Ubicación: Toda la unidad

Mejora (tipo 1): Actualizar licencias de recursos en las imágenes y eliminar Anexo final

Ubicación: Ejercicios resueltos y sección 3.3.

Mejora (tipo 1): En la unidad completa faltan los ejercicios resueltos y en la sección 3.3 la pregunta de autoevaluación esta mal, solo tiene puntos.

Ubicación: No especificada.

Mejora (tipo 2): Falta un tutorial más extenso y completo con muchos ejercicios resueltos sobre como realizar una DTD o un esquema en un documento XML. Por ejemplo para la DTD he elaborado el siguiente documento compartido en Google Drive: <https://drive.google.com/open?id=0ByKos1mgJL8gZmIpQnh5NVdXX00&authuser=0>

Para los esquemas he elaborado este otro:<https://drive.google.com/open?id=0ByKos1mgJL8genM5aU94ckdER00&authuser=0>

Versión: 01.01.00

Fecha de actualización: 05/01/15

Autoría: Víctor Javier Gómez Arques

Múltiples conceptos erróneos, erratas varias, ejemplos y ejercicios resueltos arreglados, glosario, enlaces de apoyo, videos explicativos, descargas de referencia rápida, cuestionarios corregidos.

Versión: 01.00.00

Fecha de actualización: 28/04/14

Versión inicial de los materiales.

