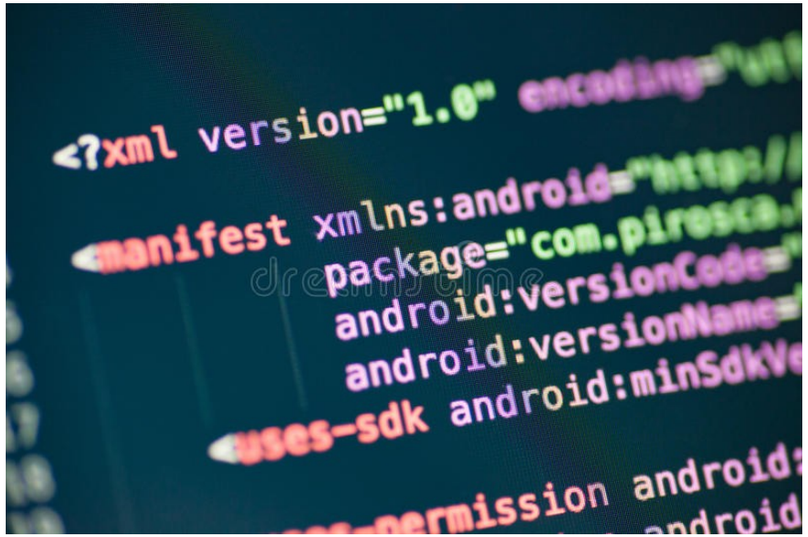
Lenguajes de marcas y sistemas de gestión de la información



UT05 – DTD y XML Schema

1 – Introducción – DTD – Validación de elementos

IES Clara del Rey – Madrid

Documentos bien formados

Para que un documento XML esté bien formado:

- Tiene que tener un solo elemento raíz.
- Todas las etiquetas (tags) abiertas deben tener sus respectivas etiquetas de cierre.
- XML distingue mayúsculas y minúsculas. Las aperturas y cierres de etiquetas tienen que ser exactamente iguales.
- Todos los elementos deben estar correctamente anidados.
- Los valores de los atributos deben ir entre comillas simples o dobles.

Documentos bien formados

Para que un documento XML esté bien formado:

- Los elementos vacíos pueden terminar /> o con un tag de cierre correspondiente al de apertura
- No se pueden repetir atributos en un mismo elemento.
- Los nombres de los elementos y atributos tienen que utilizar sólo los caracteres autorizados.
- Se deben escapar con entidades los caracteres especiales (<, >, &, etc.)

Documentos válidos

XML permite definir otros lenguajes con un propósito específico, definiendo la estructura de los documentos XML para un fin concreto.

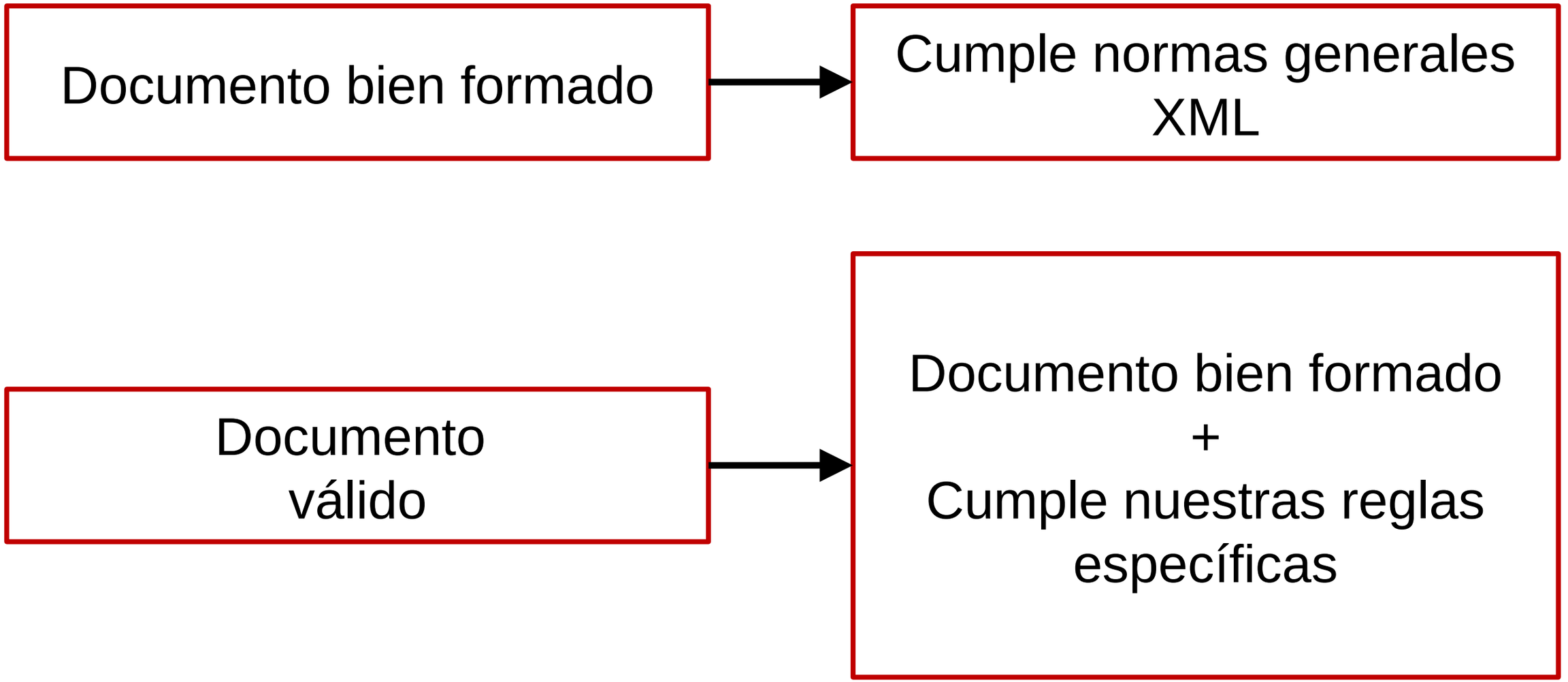
Cuando definimos un formato específico para cierta aplicación estamos creando un nuevo lenguaje, un dialecto con una aplicación específica.

El conjunto de los elementos y atributos admitidos en el lenguaje se denomina vocabulario.

Un documento XML bien formado no es siempre un documento válido.

Si un documento XML no se ajusta al vocabulario y normas que hemos establecido para cierto dialecto, no será un documento válido, aunque esté bien formado.

Documentos válidos



Limitaciones de XML para definir dialectos

Con sólo XML no podemos especificar:

- Qué nombres deben tener los elementos y sus atributos.
- Qué jerarquía de elementos se tiene que crear en el documento. Cómo se colocan y organizan. Cómo se anidan.
- Si un elemento se puede repetir o no. Y si puede repetirse, cuántas veces puede hacerlo.
- Qué tipo de datos (número, caracteres, etc.) puede contener un elemento.
- Qué conjunto de valores pueden tomar, si se puede limitar a una enumeración.
- Si los elementos o atributos son o no opcionales.

Definiendo reglas para un dialecto XML

Hay dos herramientas que se pueden utilizar para definir el vocabulario y reglas sintácticas de un dialecto XML específico:

- DTD: Document Type Definition
- XSD: XML Schema Definition

Ambos pueden servir, pero DTD es más antiguo (más antiguo que el propio XML), y tiene limitaciones que XSD vino a solucionar.

Pero DTD sigue utilizándose, porque en muchos casos es suficiente con las reglas básicas que se pueden definir.

DTD – Declaración

La declaración DTD define reglas sintácticas que debe seguir el documento XML.

Define los elementos, atributos y entidades y notaciones que pueden usarse en el documento.

Define restricciones estructurales (anidamiento) y de contenido, usando declaraciones:

- De tipos de elemento
- De listas de atributos para los elementos
- De entidades
- De notación

DTD – Declaración

La declaración DTD puede realizar de tres formas:

- Interna. Dentro del propio documento XML.
- Externa. En un fichero independiente.
- Mixta: una mezcla de ambas.

La declaración externa tiene la ventaja de que puede utilizarse para varios documentos del mismo dialecto.

Para declarar el tipo de documento se utiliza la referencia

```
<!DOCTYPE root >
```

en el documento XML, donde “root” es el elemento raíz del documento.

DTD – Declaración interna

Las declaraciones están incluidas dentro del propio documento XML. Son de ese documento, y no pueden reutilizarse en otros documentos

En este caso se debe añadir el atributo "standalone" en la declaración XML, y debe tener el valor "yes", que significa que el documento XML no depende de una DTD externa para validarse.

El atributo "standalone", cuando no está presente, se presupone "no".

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE root [
      <!-- Declaraciones -->
]>
<root> </root>
```

DTD – Declaración externa – Propia

Las declaraciones están incluidas en un fichero independiente del fichero XML. Este fichero de declaraciones DTD sí puede reutilizarse.

En este caso se debe omitir el atributo "standalone" en la declaración XML, o debe tener el valor "no".

Para referenciar ficheros DTD que hemos creado nosotros mismos es usando “SYSTEM”:

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE root SYSTEM "fichero-externo.dtd">
<root> </root>
```

DTD – Declaración externa – Pública

Es una referencia a un DTD bien conocido, que se aplica para un dialecto concreto, normalmente un estándar. Por ejemplo, XHTML, SVG, MATHML, etc.

En este caso también se debe omitir el atributo "standalone" en la declaración XML, o debe tener el valor "no".

En este caso se utiliza "PUBLIC". Además de la URL del fichero DTD, se añade un valor con la identificación del tipo de documento.

Ejemplo:

```
<!DOCTYPE html PUBLIC
     "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Declaraciones DTD

Entre los corchetes se declara la estructura del documento XML:

```
<!DOCTYPE root [
    <!– Declaraciones con la
        estructura y restricciones del dialecto XML -->
]>
```

Estas declaraciones pueden ser:

- Elementos
- Atributos
- Entidades
- Notaciones
- Comentarios

Declaraciones DTD – Elementos

La declaración de un elemento es:

```
<!ELEMENT nombre-elemento contenido>
```

Si contenido es…	El elemento…
EMPTY	Debe estar vacío, pero puede tener atributos
ANY	Contiene cualquier cosa o estar vacío
(#PCDATA)	Puedo contener solo caracteres (parsed character data), no otros elementos.
(nombreElemento)	Contiene otro elemento
(nombreElem1, nombreElem2, …)	Contiene una sucesión de elementos

Hay que declarar todos los elementos de un documento, incluido root

Declaraciones DTD – Normas

- La declaración del tipo de documento (doctype) debe aparecer al inicio del documento XML.
- La declaración del doctype y de los elementos y atributos debe comenzar con un signo de exclamación (!)
- El nombre de la declaración del DOCTYPE debe coincidir con el nombre del elemento raíz del documento XML.

Declaraciones DTD – Elementos – Ejemplos

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE persona [
    <!ELEMENT persona (nombre, mayorDeEdad, ciudad)>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT mayorDeEdad EMPTY>
    <!ELEMENT ciudad (#PCDATA)>
]>
<persona>
    <nombre>Juan Pérez</nombre>
    <mayorDeEdad />
    <ciudad>Madrid</ciudad>
</persona>
```

Declaraciones DTD – Elementos – Ejemplos

- *<!DOCTYPE persona* - El elemento raíz es "persona"
- *<!ELEMENT persona (nombre, mayorDeEdad, ciudad)>* - El elemento raíz persona tendrá tres elelementos hijos, "nombre", "mayorDeEdad" y "ciudad".
- *<!ELEMENT nombre (#PCDATA)>* - El contenido de "nombre" es texto.
- *<!ELEMENT mayorDeEdad EMPTY>* - El elemento "mayorDeEdad" no puede tener contenido (pero sí atributos).
- *<!ELEMENT ciudad (#PCDATA)>* - El contenido de "ciudad" es texto.

Declaraciones DTD – Elementos – Ejemplos

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE casasRurales [
    <!ELEMENT casasRurales (casa*)>
    <!ELEMENT casa (direccion, descripcion, estado, tamanio)>
    <!ELEMENT direccion (#PCDATA)>
    <!ELEMENT descripcion (#PCDATA)>
    <!ELEMENT estado (#PCDATA)>
    <!ELEMENT tamanio (#PCDATA)>
]>
<casasRurales>
    <casa>
        <direccion>Calle 1, Pueblo 1</direccion>
        <descripcion>Encantadora casa de campo ...</descripcion>
        <estado>Buen estado</estado>
        <tamanio>150 m²</tamanio>
    </casa>
```

Declaraciones DTD – Elementos – Ejemplos

- *<!DOCTYPE casasRurales* - El elemento raíz debe ser "casasRurales".
- *<!ELEMENT casasRurales (casa*)>* - "casasRurales" tiene que contener cero o más elementos "casa". Veremos *, + y ? más adelante.
- *<!ELEMENT casa (direccion, descripcion, estado, tamanio)>* - "casa" tiene que contener "dirección", "descripción", "estado" y "tamanio".
- *<!ELEMENT direccion (#PCDATA)>* - "direccion" contiene texto.
- *<!ELEMENT descripcion (#PCDATA)>* - "descripcion" contiene texto.
- *<!ELEMENT estado (#PCDATA)>* - "estado" contiene texto.

Declaraciones DTD – Orden y cardinalidad

Cuando indicamos que un elemento contiene una secuencia de otros:

- El orden en que se escriben las declaraciones debe respetarse.
- Se puede indicar cardinalidad:

Notación	Implica…
(descendiente)	Aparece 1 vez
(descendiente?)	Aparece entre 0 y 1 vez (elemento opcional)
(descendiente+)	Aparece al menos una vez (1 a n veces)
(descendiente*)	Puede no aparecer (0 a n veces)
(desc1, desc2, …)	Debe contener todos en el mismo orden
(desc1 | desc2)	Debe contener o uno u otro elemento (sólo uno de los dos)

Declaraciones DTD – Elementos – Ejemplos

```
<!DOCTYPE cv [
     <!ELEMENT cv (nombre, direccion, telefono,
                                        fax?, email+, idiomas)>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT direccion (#PCDATA)>
    <!ELEMENT telefono (#PCDATA)>
    <!ELEMENT fax (#PCDATA)>
    <!ELEMENT email (#PCDATA)>
    <!ELEMENT idiomas (idioma*)>
    <!ELEMENT idioma (#PCDATA)>
]>
```

El fax es opcional, el email debe aparecer al menos una vez.

El elemento "idiomas" debe aparecer, pero dentro de este, pueden aparecer cero o más idiomas.

Declaraciones DTD más complejas

Se pueden combinar las reglas para conseguir estructuras más complejas.

Ejemplo:

```
<!ELEMENT articulo ((codigo|id), nombre?)>>
```

Cada elemento "articulo":

- Debe tener un elemento "codigo" o "id". No los dos a la vez
- Puede tener (opcional) un elemento "nombre".

Anidando este tipo de declaraciones se puede realizar la definición / validación de dialectos más o menos complejos.