Ciclo Formativo de Grado Superior de Administración de Sistemas Informáticos en red

Módulo Profesional: LMSGI

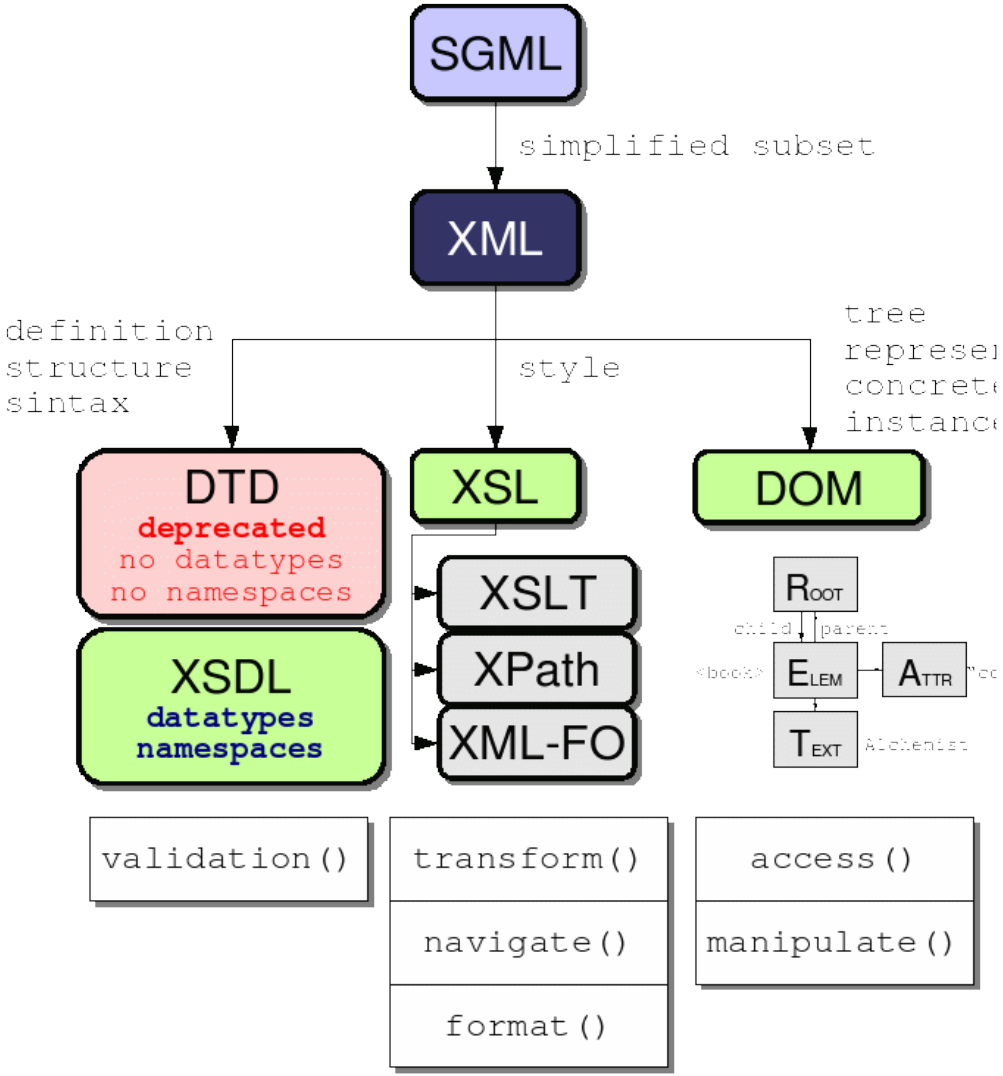
Unidad de Trabajo 7.- XML, Extensible Markup Language (lenguaje de marcas extensible). W3C DTD/XML Schema

Departamento de Informática y Comunicación
IES San Juan Bosco (Lorca-Murcia)
Profesor: Juan Antonio López Quesada

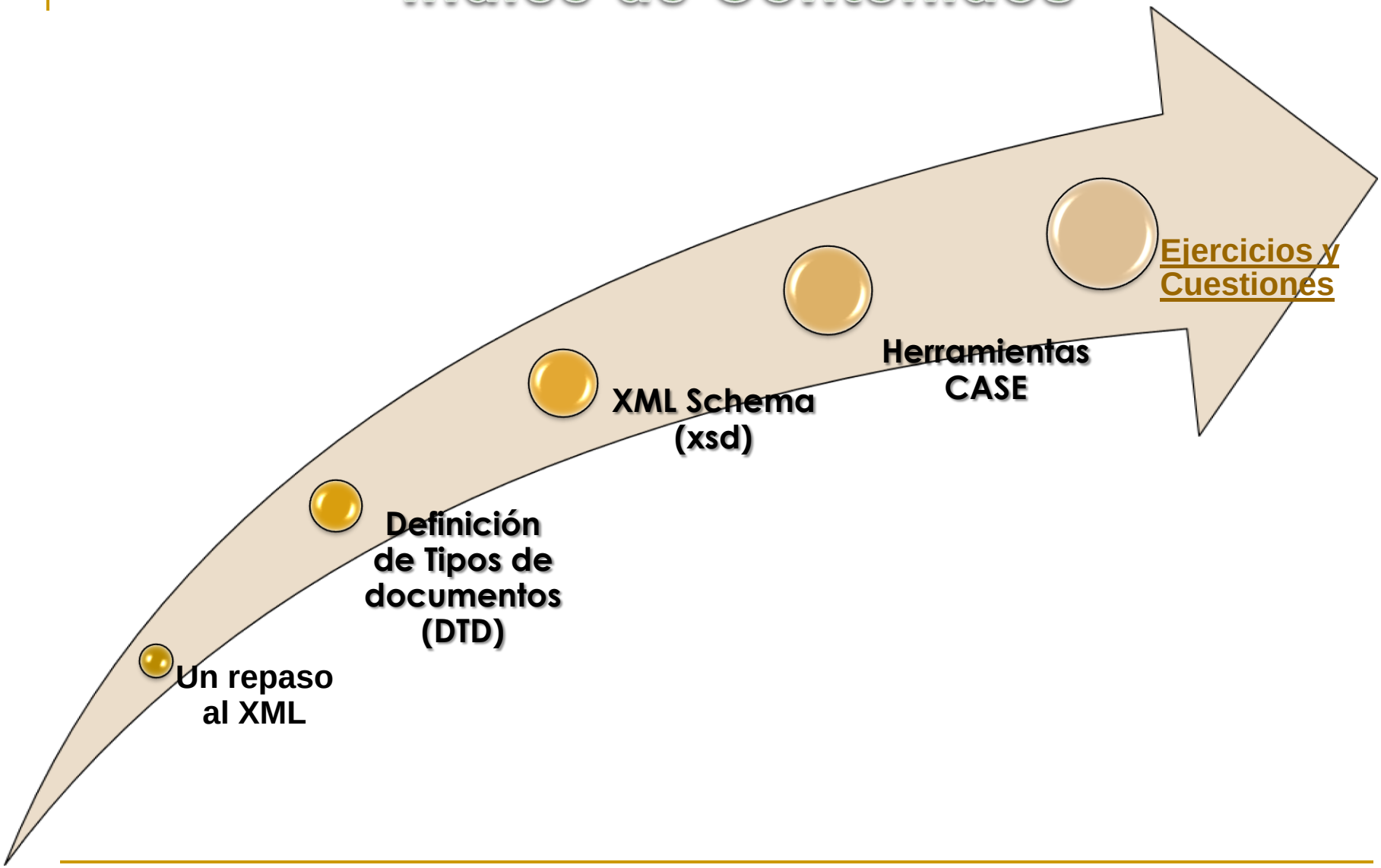
Abstract/Resumen

- La necesidad de jerarquizar y estructurar correctamente la información, no sólo para almacenarla, sino también para acceder a ella, se ha convertido en una labor que ha cobrado especial relevancia en los últimos años, en los que se han producido importantes avances en este campo.
- Inicialmente se usaron las Definiciones del Tipo de Documento (DTDs) para describir el vocabulario necesario para identificar todos los elementos de que iba a constar el documento y para expresar la estructura.
- Pero las DTDs no satisficieron todas las necesidades inherentes a XML y pronto se vio necesario utilizar otros métodos más rigurosos y sofisticados para tratar la estructura y la semántica dentro de un documento XML. Así surgieron los Esquemas XML (XML Schema), como una forma de ampliación y mejora de las primitivas DTDs. Las DTDs y los Schemas son usados por los analizadores sintácticos o parsers para comprobar si un documento XML es válido.
- Así pues, vemos que para proceder a la estructuración o especificación formal dentro de un documento XML existen distintas soluciones, entre las que cabe destacar principalmente dos: las DTDs y los XML Schemas

XML: eXtensible Markup Language



Índice de Contenidos



Un repaso al XML: Introducción

- Ya se adelantó la definición de documento XML como una información jerarquizada, en forma de texto, que constituye un objeto de datos que puede ser presentado mediante una estructura de árbol, que puede estar almacenado en un único archivo o estar dividido en varios.

- Para crear un documento XML se puede utilizar cualquier editor especializado; son cada vez más los paquetes de software que facilitan que sus datos puedan salvarse como tales documentos XML.

- La presentación textual de un documento XML, como en todo ML, se puede resumir de la forma : ***Texto XML = datos + marcado***. Esto significa que el texto de un documento XML consta de dos conjuntos : marcado y datos. El marcado corresponde a las instrucciones que el analizador XML debe procesar (que se incluyen entre los paréntesis angulares) mientras que los datos son el texto entre las marca o etiqueta delimitada, en inicio y final por paréntesis angulares. El procesador, una vez determinado que todos los caracteres de un documento son aceptables, los diferencia entre texto de marcado y caracteres de datos (CDATA).

Un repaso al XML: Introducción

- Es importante reseñar que desde el principio debe distinguirse entre datos analizables (Parsed Carácter Data o PCDATA) y no analizables, y que su mezcla, en principio, no es un problema ya que admite esta posibilidad sin problema alguno, de la misma forma que en un texto en castellano se puede incluir un texto inglés, siempre que se señale adecuadamente.

- Los caracteres de datos corresponden a todo lo que no es marcado. La secuencia es: un inicio < >, seguido de un contenido, y de una finalización < / > (la única excepción son las referencias a entidades que, como veremos, comienzan con el carácter “&”, y terminan con el carácter “;”).

Un repaso al XML: Sintaxis

Un repaso al XML: Sintaxis

Etiquetas de Elemento

- **Las etiquetas constituye el componente más evidente de la sintaxis XML y se emplean para describir elementos. <ciudad> y </ciudad>.**

Los elementos XML pueden tener contenido (más elementos, caracteres, o ambos a la vez), o bien ser elementos vac&iacuteos.

Un elemento con contenido es, por ejemplo:

<nombre>Fernando Damián</nombre>

<aviso tipo="emergencia" gravedad="mortal">Que no cunda el pánico</aviso>

Siempre empieza con una <etiqueta> que puede contener atributos o no, y termina con una </etiqueta> que debe tener el mismo nombre.

Un repaso al XML: Sintaxis

Etiquetas de Elemento

Un elemento puede estar vacío, por lo que no tiene contenido. Por ejemplo;

<identificador DNI="23123244"/>

<linea-horizontal/>

Al no tener una etiqueta de cierre que delimite un contenido, se utiliza la forma <etiqueta/>, que puede contener atributos o no.

Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento.

Por ejemplo, un elemento "chiste" puede tener un atributo "tipo" y un atributo "calidad", con valores "vascos" y "bueno" respectivamente.

<chiste tipo="vascos" calidad="bueno"> Esto es un día que Patxi y Josu van paseando... </chiste>

Un repaso al XML: Sintaxis

Etiquetas de Elemento

En una Definición de Tipo de Documento, se especifican los atributos que pueden tener cada tipo de elemento, así como sus valores y tipos de valor posible.

Al igual que en otras cadenas literares de XML, los atributos pueden estar marcados entre comillas simples (') o doble ("). Cuando se usa uno para delimitar el valor del atributo, el otro tipo se puede usar dentro.

<verdura clase="zanahoria" logitud='15" y media'>

<cita texto="'Hola buenos dias', dijo él">

A veces, un elemento con contenido, puede modelarse como un elemento vacío con atributos.

<gato><nombre>Micifú</nombre><raza>Persa</raza></gato>
<gatoraza="Persa">Micifú</gato>
<gato raza="Persa" nombre="Micifú"/>

Un repaso al XML: Sintaxis

Instrucciones de procesamiento

```
<?xml version=" 1.0 " encoding=" UTF-8 " standalone= " yes "?>
 <ficha>
<nombre> Angel </nombre>
<apellido> Barbero </apellido>
<direccion> c/Ulises, 36 </direccion>
</ficha>
```

Lo primero que tenemos que observar es la primera línea. Con ella deben empezar todos los documentos XML, ya que es la que indica que lo que la sigue es XML. Aunque es opcional, es recomendable incluirla. Puede tener varios atributos, algunos obligatorios y otros no:

- ***version:*** *Indica la versión de XML usada en el documento. Es obligatorio ponerlo, a no ser que sea un documento externo a otro que ya lo incluía.*
- ***encoding:*** *La forma en que se ha codificado el documento. Se puede poner cualquiera, y depende del parser el entender o no la codificación. Por defecto es UTF-8, aunque podrían ponerse otras, como UTF-16, US-ASCII, ISO-8859-1, etc.*
- ***standalone:*** *Indica si el documento va acompañado de un DTD ("no"), o no lo necesita ("yes"); en principio no hay porqué ponerlo, porque luego se indica el DTD* si se necesita.

Un repaso al XML: Sintaxis

Instrucciones de procesamiento

```
<?xml version="1.0" encoding='ISO-8859-1'?>
<?xml-stylesheet href="tienda0.xsl" type="text/xsl"?>
<tienda>
<nombre>La tiendecilla  </nombre>
<telefono>953 87 12 23  </telefono>
 </tienda>
```


```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template
match='/'>
<html>
<head><title>Generado con tienda-html.xsl</title></head>
<body>
<h1> <xsl:apply-templates /> </h1>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Un repaso al XML: Sintaxis

Declaración de Tipo de Documento

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

<!DOCTYPE lista_de_personas [
<!ELEMENT lista_de_personas (persona*)>
<!ELEMENT persona (nombre, fechanacimiento?, sexo?, numeroseguridadsocial?)>
<!ELEMENT nombre (#PCDATA) >
<!ELEMENT fechanacimiento (#PCDATA) >
<!ELEMENT sexo (#PCDATA) >
<!ELEMENT numeroseguridadsocial (#PCDATA)>
]>

<lista_de_personas>
  <persona>
    <nombre>José García</nombre>
    <fechanacimiento>25/04/1984</fechanacimiento>
    <sexo>Varón</sexo>
  </persona>
</lista_de_personas>
```

Un repaso al XML: Sintaxis

Declaración de Tipo de Documento

La "declaración de tipo de documento" define qué tipo de documento estamos creando para ser procesado correctamente. Es decir, definimos que declaración de tipo de documento (DTD) valida y define los datos que contiene nuestro documento XML.

En ella se define el tipo de documento, y dónde encontrar la información sobre su Definición de Tipo de Documento, mediante:

- **Un identificador público (PUBLIC):** que hace referencia a dicha DTD.
- **Identificador universal de recursos (URI):** precedido de la palabra SYSTEM.

Ejemplos:

```
<!DOCTYPE MESAJE SYSTEM "mesaje.dtd">
<!DOCTYPE HTML PUBLIC "-/ /W3C/ /DTD HTML 3.2 Final/ /EN">
<!DOCTYPE LABEL SYSTEM "http://azuaje.ulpgc.es/dtds/label.dtd">
```

Un repaso al XML: Sintaxis

Referencias a Entidades

Existen varios tipos de entidades que podemos definir en nuestras DTD's:

Entidades internas

También llamadas macros ó constantes de texto, las entidades internas son las que se asocian a una cadena de caracteres. Se referencian única y exclusivamente desde el fichero.

Ej: <!ENTITY nom "Juan Pérez López">

Así, si escribo en mi documento (en el código fuente): &nom; es como si estuviera escribiendo Juan Pérez López.

XML proporciona 5 entidades predefinidas que se declaran automáticamente y que están a disposición de todo documento XML:

< Es sustitído por el símbolo < , > Es sustitído por el símbolo > , & Es sustitído por el símbolo & , ' Es sustitído por el símbolo ' , " Es sustitído por el símbolo "

Entidades externas, de parámetros, ...

Un repaso al XML: Sintaxis

Comentarios

A veces es conveniente insertar comentarios en el documentos XML, que son ignorados por el procesado de la información y las reproducciones del documento. Los comentarios tienen el mismo formato que los comentarios de HTML. Es decir, comienza por la cadena "<!--" y termina con "-->".

<!-- Esto es un comentario -->

Se pueden introducir comentarios en cualquier parte del documento salvo dentro de las declaraciones, etiquetas, u otros comentarios.

Un repaso al XML: Sintaxis

Secciones CDATA

Existe otra construcción llamada CDATA (Character DATA) en XML que permite especificar datos, utilizando cualquier carácter, especial o no, sin que se interprete como marcado XML.

Ejemplo primero usando entidades predefinidas y luego con un bloque CDATA

```
<parrafo>Lo siguiente es un ejemplo de HTML.<html>
<ejemplo>
&lt;html>
&lt;head>&lt;title>Rock &amp; Roll&lt;/title>&lt;/head>
</ejemplo>
<ejemplo>
<![CDATA[
<html>
<head><title>Rock & Roll</title></head>
]]>
</ejemplo>
```

Como hemos visto dentro de una sección CDATA podemos poner cualquier cosa, que no será interpretada. Existe una excepción y es la cadena "]]>" con el que termina el bloque CDATA. Esta cadena no puede utilizarse dentro de una secció CDATA.

Un repaso al XML: Sintaxis

Espacios de Nombres

Un espacio de nombres XML es una recomendación W3C para proporcionar elementos y atributos con nombre único en una instancia XML. Una instancia XML puede contener nombres de elementos o atributos procedentes de más de un vocabulario XML. Si a cada uno de estos vocabularios se le da un espacio de nombres, se resuelve la ambigüedad existente entre elementos o atributos que se llamen igual. Los nombres de elementos dentro de un espacio de nombres deben ser únicos.

Un ejemplo sería una instancia XML que contuviera referencias a un cliente y a un producto solicitado por éste. Tanto el elemento que representa el cliente como el que representa el producto pueden tener un elemento hijo llamado "numero_ID". Las referencias al elemento "numero_ID" podrían ser ambiguas, salvo que los elementos, con igual nombre pero significado distintos, se llevaran a espacios de nombres distintos que los diferenciaran.

Un repaso al XML: Sintaxis

Espacios de Nombres

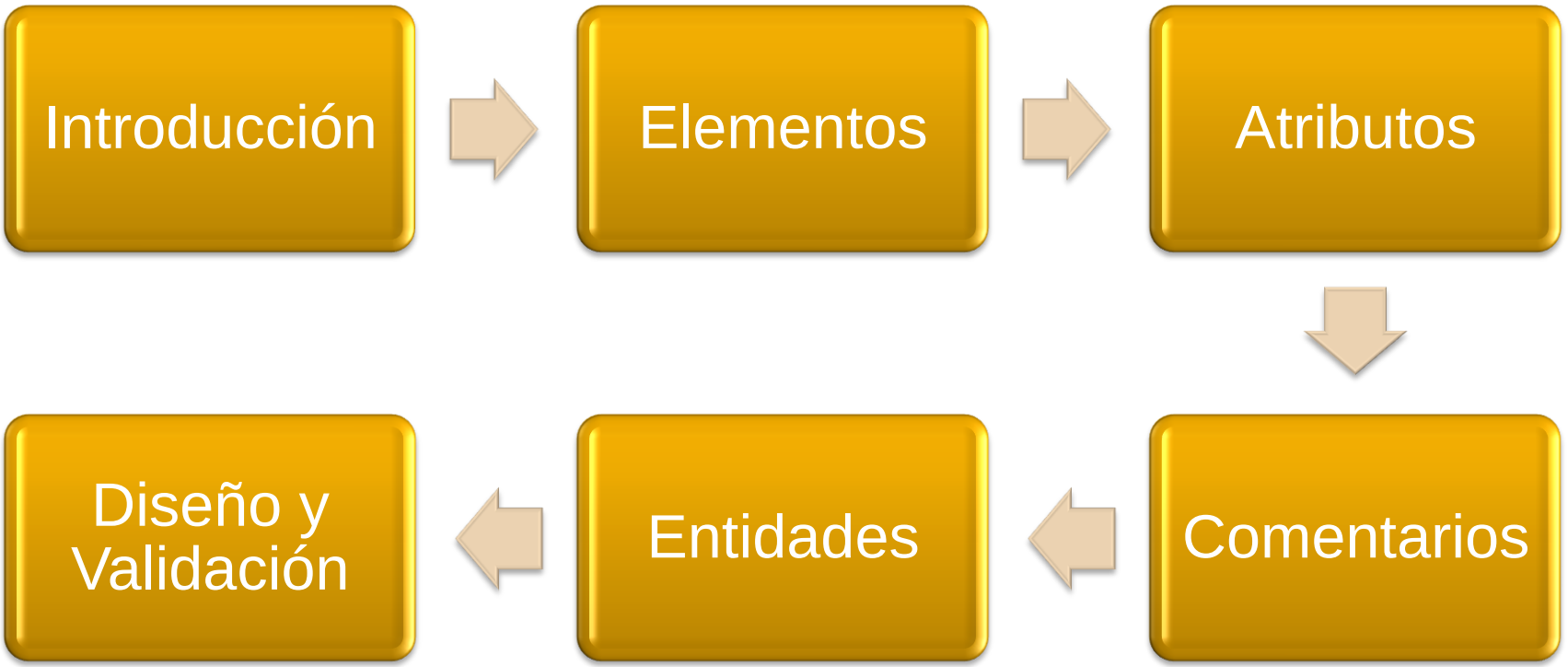
```
<?xml version="1.0"?>
<cli:cliente xmlns:cli='http://www.dominio.org/Espacio_de_nombres_XML/cliente'
xmlns:ped='http://www.dominio1.org/Espacio_de_nombres_XML/pedido'>
<cli:numero_ID>1232654</cli:numero_ID>
<cli:nombre>Fulanito de Tal</cli:nombre> <cli:telefono>99999999</cli:telefono>
<ped:pedido>
<ped:numero_ID>6523213</ped:numero_ID>
<ped:articulo>Caja de herramientas</ped:articulo> <ped:precio>187,90</ped:precio>
</ped:pedido>
</cli:cliente>
```

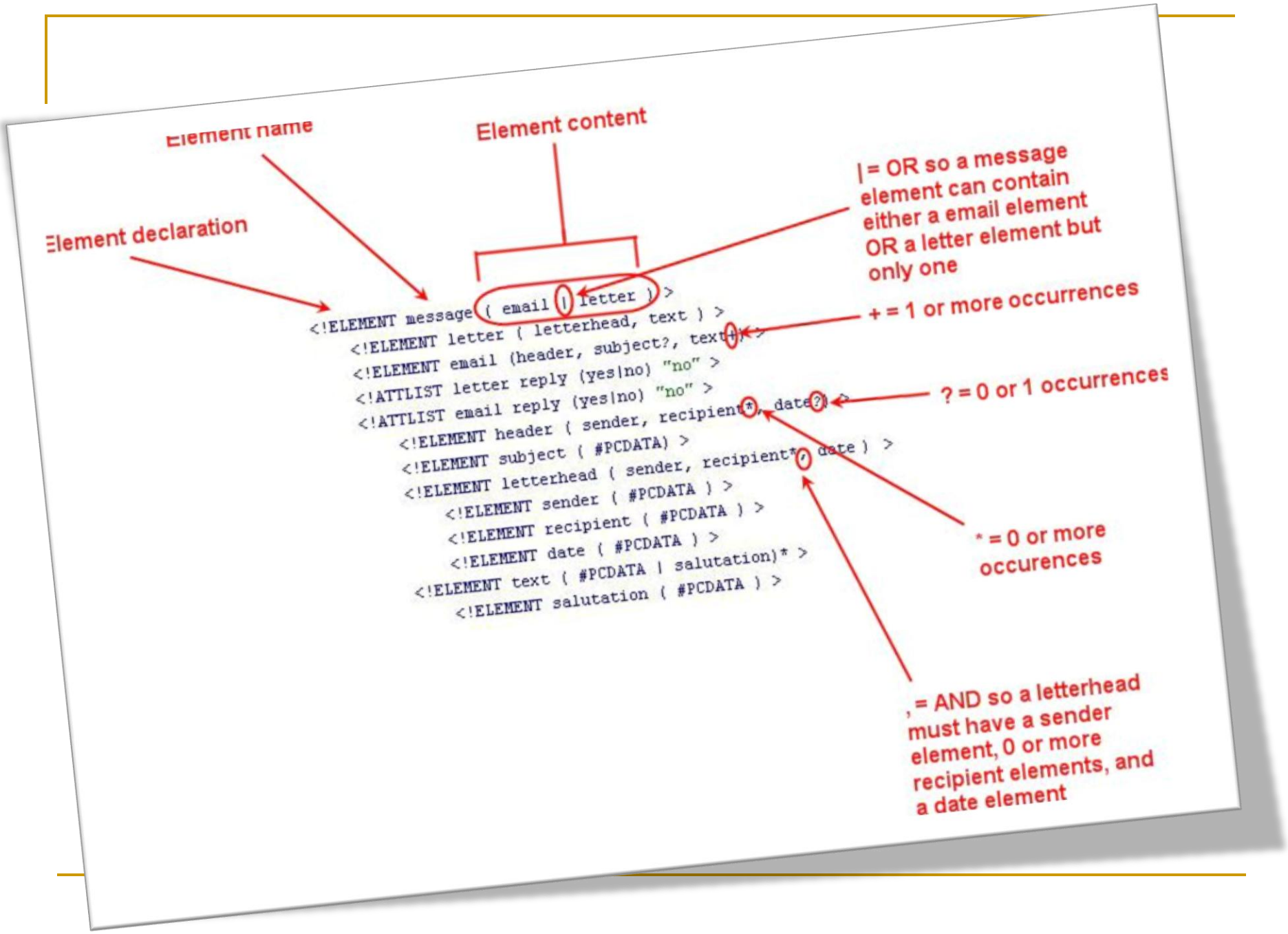
Un espacio de nombres se declara usando el atributo XML reservado xmlns, cuyo valor debe ser un identificador uniforme de recurso.

Por ejemplo:

xmlns="http://www.w3.org/1999/xhtml"

Definición de Tipos de documentos (DTD)




Element name
Element content
Element declaration
| = OR so a message element can contain either a email element OR a letter element but only one
+ = 1 or more occurrences
? = 0 or 1 occurrences
* = 0 or more occurences
, = AND so a letterhead must have a sender element, 0 or more recipient elements, and a date element
<!ELEMENT message (email | letter) >
<!ELEMENT letter (letterhead, text) >
<!ELEMENT email (header, subject?, text+) >
<!ATTLIST letter reply (yes|no) "no" >
<!ATTLIST email reply (yes|no) "no" >
<!ELEMENT header (sender, recipient*, date?) >
<!ELEMENT subject (#PCDATA) >
<!ELEMENT letterhead (sender, recipient*, date) >
<!ELEMENT sender (#PCDATA) >
<!ELEMENT recipient (#PCDATA) >
<!ELEMENT date (#PCDATA) >
<!ELEMENT text (#PCDATA | salutation)* >
<!ELEMENT salutation (#PCDATA) >

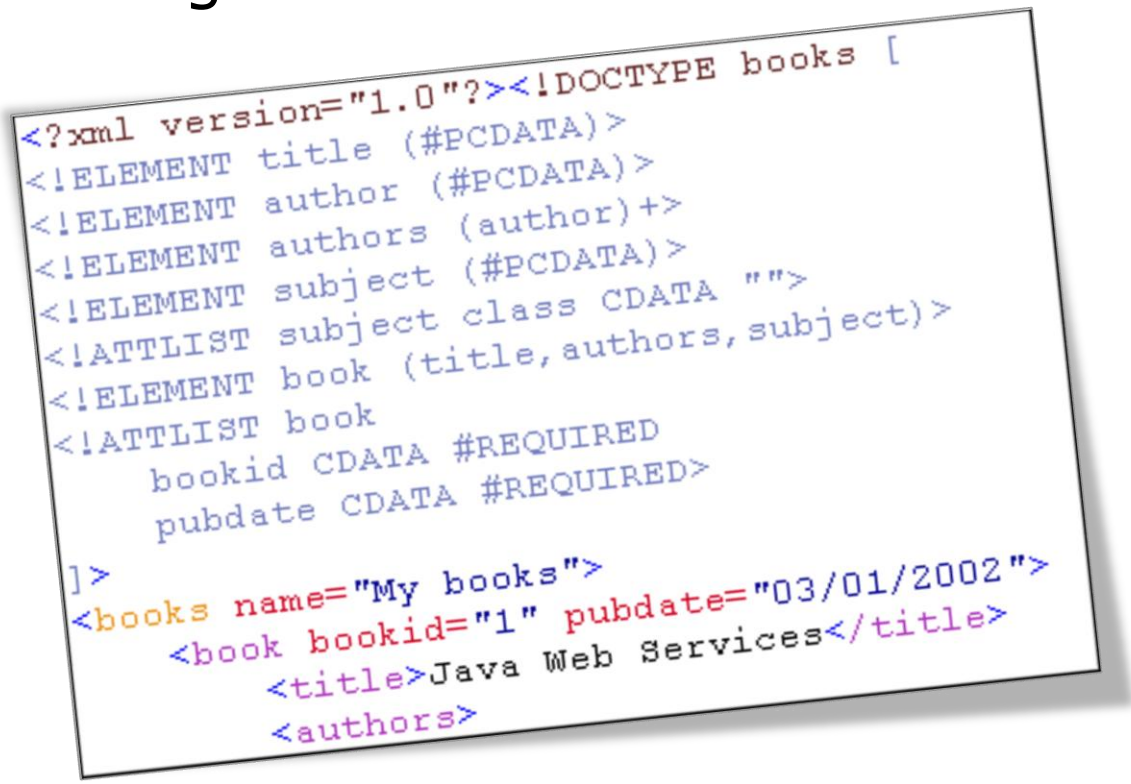
DTD: Introducción

- Una DTD indica:
  - Qué elementos pueden ser utilizados en un tipo de documento específico
  - Cuales son obligatorios y cuales opcionales
  - Cuales son repetibles y cuales no
  - En qué orden deben aparecer
  - Cómo deben anidarse los elementos que conforman un documento

- La DTD también contiene:
  - La declaración de las entidades que se utilizan en el documento:
    - Recursos externos XML
    - Recursos externos no XML: gráficos, multimedia, etc.
    - Texto que actúa como 'comodín' o 'abreviatura' para palabras de uso frecuente o términos que cambian con facilidad
  - La declaración de notaciones (intrucciones para procesar las entidades no xml)

DTD: Introducción

- La DTD utiliza una sintáxis especial para definir la estructura de un tipo de documento.
- Esta sintáxis utiliza los siguientes elementos:
  - ELEMENT
  - ATTLIST
  - ENTITY
  - NOTATION
  - Comentarios



DTD: Introducción

- La DTD contiene 'declaraciones' de los distintos elementos, entidades y notaciones que pueden aparecer en un documento XML.

- Cada declaración se escribe entre los caracteres <! Y >, indicando el tipo de componente que se declara, por ejemplo:

```
<!ELEMENT autor (#PCDATA)>
<!ENTITY graf SYSTEM "c:\graf.gif" NDATA gif>
```

DTD: Elementos

- Los elementos se corresponden con los componentes estructurales de un documento, y definen su estructura lógica.
- Un elemento puede contener datos de tipo carácter, otros elementos o ambos a la vez.
- Los elementos pueden contenerse unos a otros, formando una jerarquía o árbol.
- Un documento XML siempre tiene un elemento raíz o 'elemento documento', que engloba a todos los demás.

DTD: Elementos

- El elemento raíz se debe llamar igual que su tipo de documento (por ejemplo, si creamos un tipo de documento 'articulo', el elemento raíz deberá llamarse 'articulo'
- El nombre de los elementos puede contener caracteres a-z, A-Z y _. El nombre no debe contener el carácter & o empezar con las letras X,M,L.
- Los nombres de elementos son sensibles a la diferencia entre mayúsculas y minúsculas

  <Casa> no es igual que <casa>

DTD: Elementos

- La declaración de un elemento en la DTD indica:
  - El nombre del elemento
  - El contenido que puede tener, también llamada 'declaración de contenido'.
  - La 'declaración de contenido' se escribe entre paréntesis.
  - La declaración de un elemento se encierra entre las marcas <!ELEMENT y >.

  En la 'declaración de contenido' se puede indicar:

  - El nombre de otros elementos. No es necesario haber declarado un elemento para poder utilizarlo en la declaración de contenido de otro elemento de la misma DTD
  - La palabra reservada #PCDATA, que indica que el elemento puede contener datos de tipo carácter.

DTD: Elementos

- #PCDATA
  - Esta declaración de contenido indica que el elemento puede contener cualquier tipo de texto que no sea 'mark up'.
  - Es decir, cualquier letra menos <, > o &
  - En lugar de estos caracteres, su usarán las entidades < > ó &
  - Las comillas simples y dobles pueden sustituirse por las entidades " y '
  - Los elementos con contenido #PCDATA pueden contener referencias a entidades.

DTD: Elementos

Un elemento puede tener contenido de tipo:

- **datos ó #PCDATA,**
- **elementos (anida otros elementos)**
- **mixto (datos y elementos)**
- **vacío (el elemento no contiene ni datos ni elementos) EMPTY**
- **any (no hay restricciones para su contenido)**

```
<!ELEMENT batiburrillo ANY>
```

```
<!ELEMENT enfasis (#PCDATA)>
<!ELEMENT parrafo (#PCDATA|enfasis)*>
```

```
<!ELEMENT mensaje (remite, destinatario, texto)>
```

```
<!ELEMENT salto-de-pagina EMPTY>
```

DTD: Elementos

- La declaración de contenido ANY indica que un elemento puede contener cualquier combinación de datos de tipo carácter y de elementos (no hay ninguna restricción).
- Esta declaración de contenido no se suele utilizar. Sólo se usa en fases de diseño y prueba de DTDs.
- Si la declaración de contenido del elemento contiene otros elementos, se puede indicar:
  - **si estos elementos son obligatorios o no y**
  - **cuantas veces pueden aparecer**
- Estas restricciones se indican escribiendo un carácter especial tras el nombre del elemento utilizado
  - **? el elemento puede aparecer 0 ó 1 veces (opcional no repetible)**
  - *** el elemento puede aparecer 0 ó más veces (opcional y repetible)**
  - **+ el elemento debe aparecer 1 ó más veces (obligatorio y repetible)**
  - **el elemento debe aparecer 1 vez (obligatorio no repetible)**
  - **| elección entre una serie de elementos (equivale a OR)**
  - **, orden de aparición (equivale a AND)**

DTD: Elementos

```
<!ELEMENT message ( email | letter ) >
<!ELEMENT letter ( letterhead, text ) >
<!ELEMENT email (header, subject?, text+) >
<!ATTLIST letter reply ( yes | no ) "no" >
<!ATTLIST email reply ( yes | no ) "no" >
<!ELEMENT header ( sender, recipient*, date?) >
<!ELEMENT subject ( #PCDATA) >
<!ELEMENT letterhead ( sender, recipient*, date ) >
<!ELEMENT sender ( #PCDATA ) >
<!ELEMENT recipient ( #PCDATA ) >
<!ELEMENT date ( #PCDATA ) >
<!ELEMENT text ( #PCDATA | salutation )* >
<!ELEMENT salutation ( #PCDATA ) >
```

Symbol	Meaning	Example
,	AND	header (sender, recipient*, date)
|	OR	message (email | letter)
()	Occurs only Once	(email | letter)
+	must occur at least once	(header, subject?, text+)
?	occurs either once or not at all	(header, recipient* , date?)
*	can occur zero or more times	(sender, recipient*, date)

DTD: Elementos

- Los elementos vacíos no contienen datos de tipo carácter ni a otros elementos
- En la DTD, estos elementos se definen con una declaración de contenido EMPTY

  <!ELEMENT graphic EMPTY>
- Los elementos vacíos pueden contener atributos:

```
<!ELEMENT graphic EMPTY>
<!ATTLIST id ID #REQUIRED
        src ENTITY #REQUIRED
        height CDATA #IMPLIED
        weight CDATA #IMPLIED>
```

- Los elementos vacíos se utilizan en los documentos de dos formas:
  - Con una etiqueta de inicio y de fin, sin indicar ningún contenido entre ellas:

    <graphic id="01" src="form_1"></graphic>
  - Con una única etiqueta:

    <graphic id="01" src="form_1" />

DTD: Elementos

<!ELEMENT autor (nombre, apellido+)>

<!ELEMENT catalog (product+)>

<!ELEMENT product (specifications+, options?, price+, notes?)>

<!ELEMENT specifications (#PCDATA, step)*>

<!ELEMENT author (fname, surname, jobtitle?, address, bio)>

<!ELEMENT content (p | note)+>

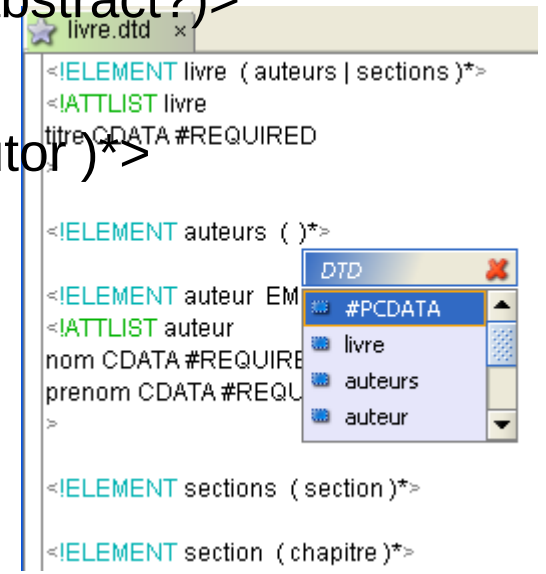
<!ELEMENT front (title, subt?, keyword*, author+, abstract?)>

<!ELEMENT seccion EMPTY>

<!ELEMENT nota (#PCDATA | mensaje | fecha | autor)*>

<!ELEMENT producto (pedido |10 | 20 | 30)>

<!ELEMENT sabor (#PCDATA | fresa | limon)>



DTD: Atributos

- Los elementos pueden tener atributos asociados, que permiten matizar su significado o área de aplicación.
- Los atributos tienen asociado un tipo de dato, un valor por defecto y un indicador que señala si son obligatorios o no.

```
<!ATTLIST mensaje fecha CDATA #REQUIRED>
    <mensaje fecha="15 de Diciembre de 1999">

<!ATTLIST mensaje fecha NMTOKEN #REQUIRED>
    <mensaje fecha="15-12-1999">
```

DTD: Atributos

- Un atributo puede recoger un tipo de dato:
  - CDATA (Character DATA) - datos de tipo carácter. Datos de caracteres no analizados sintácticamente. Estos atributos son los más sencillos, y pueden contener casi cualquier cosa.
  - Los atributos NMTOKEN (NaMe TOKEN) son parecidos, pero sólo aceptan: letras, números, puntos, guiones, subrayados y los dos puntos).
  - Los atributos NMTOKENS – Múltiples nombres NMTOKEN separados por espacios.
  - (valor | valor2 | … | valorn) - enumeración
  - ID – Identificador único
  - IDREF – Una referencia a un ID
  - ENTITY - referencias a una entidades
  - ENTITIES – Mútiples referencias a entidades separados por espacios en blanco

DTD: Atributos

```
<!ELEMENT elemento (#PCDATA)>
    <!ATTLIST  elemento
        aaa CDATA #IMPLIED
        bbb NMTOKEN #REQUIRED
        ccc NMTOKENS #REQUIRED>


<!DOCTYPE  elemento SYSTEM "tutorial.dtd">
    < elemento  aaa="#d1" bbb="a1:12" ccc=" 3.4 div    -4"> xx
    < /elemento>


<!DOCTYPE  elemento  SYSTEM "tutorial.dtd">
    < elemento  bbb="a1:12"
    ccc="3.4
    div
    -4"> xx < /elemento>
```

DTD: Atributos

```
<!ELEMENT XXX (AAA+ , BBB+ , CCC+)>
      <!ELEMENT AAA (#PCDATA)>
      <!ELEMENT BBB (#PCDATA)>
      <!ELEMENT CCC (#PCDATA)>
      <!ATTLIST AAA       id ID #REQUIRED>
      <!ATTLIST BBB       code ID #IMPLIED  list    NMTOKEN #IMPLIED>
      <!ATTLIST CCC       X ID #REQUIRED  Y       NMTOKEN #IMPLIED>


<!DOCTYPE XXX SYSTEM "tutorial.dtd">
      <XXX>
        <AAA id="a1"/>
        <AAA id="a2"/>
        <AAA id="a3"/>
        <BBB code="QWQ-123-14-6" list="14:5"/>
        <CCC X="zero" Y="16" />
      </XXX>


 <!DOCTYPE XXX SYSTEM "tutorial.dtd">
      <XXX>
        <AAA id="L12"/>
        <BBB code="QW" list="L12"/>
        <CCC X="x-0" Y="QW" />
        <CCC X="x-1" Y="QW" />
      </XXX>
```

DTD: Atributos

- Para indicar la obligatoriedad o carácter opcional de un atributo, se utilizan las palabras reservadas:
  - #IMPLIED - el atributo será opcional.
  - #REQUIRED - el atributo es obligatorio.
  - #FIXED - el atributo es asignado por defecto por el sistema, y no es necesario que el autor lo indique.
- Los atributos no son repetibles.

DTD: Atributos

Ejemplos:

```
<!ATTLIST emp fecnac CDATA #IMPLIED>

<!ATTLIST section link ID #REQUIRED
                  lastupdate CDATA #IMPLIED>

<!ATTLIST employee birthdate CDATA #REQUIRED
                   personId CDATA #REQUIRED
                   hiredate CDATA #REQUIRED
                   company (IBM | Lotus) "IBM">
```

Valor por defecto

DTD: Comentarios

- En una DTD se pueden añadir comentarios para documentar la semántica de los elementos y atributos que se declaran.
- Los comentarios se añaden entre las marcas especiales <!-- y -->.

Ejemplo:

```
<!ELEMENT graphic EMPTY>
<!--este elemento se usará para incluir gráficos-->
```

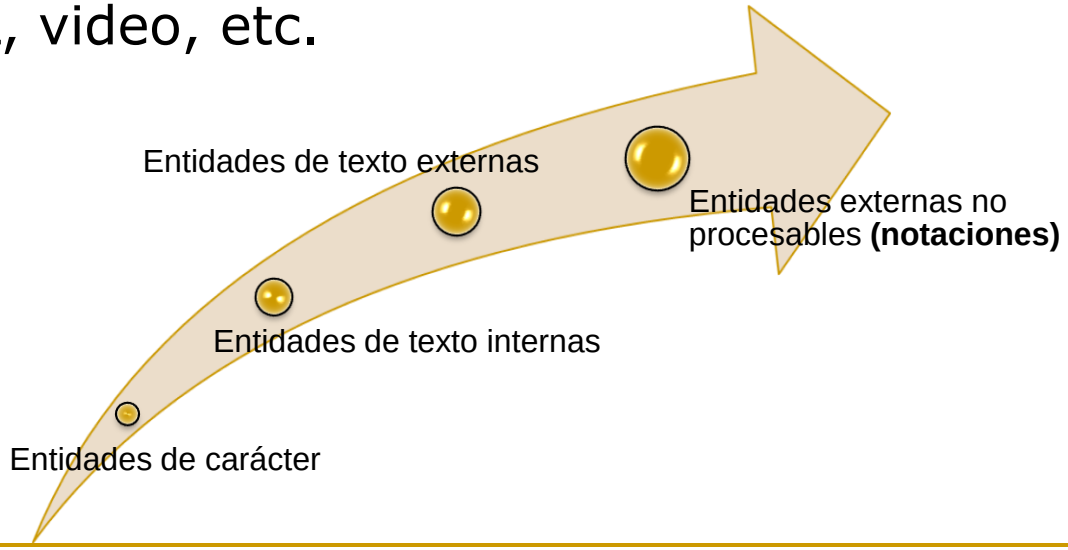
- Los comentarios pueden abarcar más de una línea.
- En el texto del comentario se pueden incluir los caracteres reservados que utiliza el mark up: <, >, &, etc.
- No pueden incluirse dos guiones seguidos a parte de los utilizados en su inicio y final.

DTD: Entidades

- Las entidades determinan la estructura física de un documento XML.
- Un documento XML siempre está formado al menos por una entidad - la entidad documento - que es el propio documento.
- Todas las referencias a archivos no xml (imágenes, multimedia, etc.) se gestionan mediante entidades.
- Las entidades cumplen distintas funciones:
  - Inserción de caracteres propios de juegos de caracteres específicos (vocales acentuadas) y caracteres utilizados en el mark up.
  - Utilización de ‘comodines’ para nombres no fijados o frases largas.
  - Inserción de componentes multimedia.
  - Modularización de los documentos.

DTD: Entidades

- Las entidades se pueden clasificar en grupos no excluyentes:
  - Internas y/o externas
    - Las internas se resuelven en la entidad documento
    - Las externas se refieren a un archivo externo
  - Procesables (o de texto) y no procesables
    - Las procesables son documentos XML
    - Las no procesables son archivos no XML: imágenes, HTML, video, etc.



DTD: Entidades

- Se utilizan en lugar de caracteres especiales (acentos, mark up, etc.) difíciles de introducir en el teclado o que se codifican de forma diferente en distintas plataformas.
- En el documento se hará referencia a las entidades escribiendo su código ASCII entre los caracteres &# y ;, por ejemplo:

  <ciudad>Alcalz de Henares</ciudad>
- XML incluye entidades predefinidas para los caracteres <, >, &, ' y "
- Para definir entidades para otros caracteres, tenemos que conocer su código Unicode (ISO 10646)
- Se puede utilizar el código decimal (precedido por &#) o el hexadecimal (precedido por &#x) seguidas de ;

DTD: Entidades

- Predefinidas en XML
  - lt <
  - gt >
  - apos ‘
  - quot “
  - amp &
- Otras entidades útiles
  - aacute
  - eacute
  - iacute
  - oacute
  - uacute
  - Aacute
  - Eacute
  - Iacute
  - Oacute
  - Uacute

DTD: Entidades

Entidades de Texto Internas

- Permiten sustituir una cadena de texto por unos caracteres más fáciles de recordar y de teclear
- Se declaran con la sintáxis:

  <!ENTITY nuevoprod “KTD-50-789890 A5”>
- Se referencian en el documento escribiendo el identificador de la entidad entre los caracteres & y ; por ejemplo:

  <producto>&nuevoprod;</producto>

DTD: Entidades

Entidades de Texto Externas

- Referencian documentos XML externos a la entidad documento, que pueden verse como si se tratase de una única unidad.
- Permiten la reutilización, el trabajo en colaboración y la modularidad.
- Se declaran con la sintáxis:

  <!ENTITY licencia SYSTEM “c:\licencia.xml”>
- Se referencian con la sintáxis habitual:

  <docbody>&licencia;</docbody >

DTD: Entidades

Entidades de Texto Externas

- En un documento modular, sólo la entidad documento puede contener una declaración de tipo de documento.
- Para poder asociar cada entidad de texto externa con un tipo de documento, se suele crear un documento XML con declaración que sólo contiene la declaración de la entidad.

```
<?xml version="1.0"?>
<!DOCTYPE libro SYSTEM "libro.dtd"[
<!ENTITY tema SYSTEM "tema.xml"> ]>
<libro>
<capitulo>&tema;</capitulo>
<capitulo>&tema;</capitulo>
</libro>
```

```
<?xml version="1.0"?>
<!ELEMENT libro (capitulo)*>
<!ELEMENT capitulo (#PCDATA)>

<seccion id="1">
<b> ccc </b>
</seccion>
<seccion id="2">
<b> bbbb </b>
</seccion>
```

DTD: Entidades

Entidades externas no procesables

- Referencian cualquier archivo que no sea XML.
- Se declaran utilizando el calificador SYSTEM o PUBLIC, y van acompañadas de una notación (información adicional de estas entidades no analizables).

  <!ENTITY logonscreen SYSTEM “c:\fm1.gif” NDATA gif>
- La notación se escribe al comienzo de la DTD.

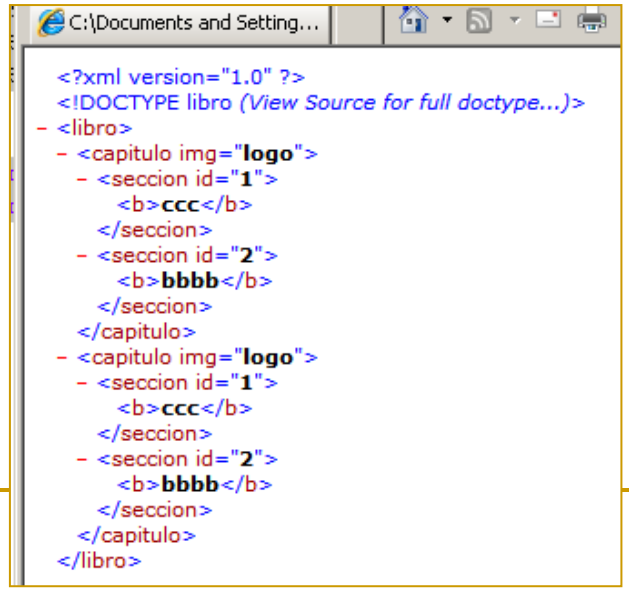
  <!NOTATION gif SYSTEM “/programas/viewer.exe”>

- Las notaciones pueden cumplir distintos propósitos:
  - Indicar el path del programa encargado de procesar la entidad (por ejemplo un visor especial)
  - Apuntar a un lugar en el que existe documentación sobre el formato
  - etc.
- La norma es abierta en este aspecto.

DTD: Entidades

Entidades externas no procesables

```
<?xml version="1.0"?>
<!DOCTYPE libro SYSTEM "libro.dtd" [
<!NOTATION PNG SYSTEM "IExplore.exe">
<!ENTITY tema SYSTEM "tema.xml">
<!ENTITY logo SYSTEM "http://www.google.es/images/srpr/nav_logo37.png" NDATA PNG>
]>
<libro>
<capitulo img="logo">&tema;</capitulo>
<capitulo img="logo">&tema;</capitulo>
</libro>
```




```
<?xml version="1.0"?>
<!ELEMENT libro (capitulo*,imagen)>
<!ELEMENT capitulo (#PCDATA)>
<!ATTLIST capitulo img ENTITY #REQUIRED>
```

DTD: Entidades

Entidades parámetro internas y externas

- Se utilizan exclusivamente en la DTD (se declaran en la DTD al igual que las entidades normales, pero se les hace referencia sólo en la DTD).
- Se declaran utilizando un carácter especial:

```
<!ENTITY % autorelem "nombre,apellido+">
```

- Para referenciarlas se escribe su nombre entre los caracteres % y ;, por ejemplo:

```
<!ELEMENT autores (noaut, %autorelem;)>
```

DTD: Entidades

<!ENTITY % p “subp, pp, foot”>
<!ELEMENT body (%p;)>

De esta forma, la declaración de contenido del elemento body equivale a (subp, pp, foot)

El ‘modelo de contenido’ al que sustituye la entidad se podrá reutilizar en otras partes de la DTD

```
<!DOCTYPE texto [
    <!ENTITY % elemento-alf "<!ELEMENT ALF (#PCDATA)>">
    ...
    %elemento-alf;
    ]>
```

También puede ser externa:

```
<!DOCTYPE texto [
    <!ENTITY % elemento-alf SYSTEM "alf.ent">
    ...
    %elemento-alf;
    ]
```

DTD: Diseño y Validación

- La DTD puede incluirse totalmente junto al documento XML (¡cuidado, en XML documento es la DTD y los datos!).

- La DTD puede estar definida en un archivo externo al documento al que se hará referencia desde este (en este caso, la DTD se podrá reutilizar y mantener con facilidad.

- La DTD puede estar definida en un documento externo, y también puede haber declaraciones en la entidad documento

DTD: Diseño y Validación

```
<?xml head version="1.0"?>
<!ELEMENT head (body)>
<!ELEMENT body
   (#PCDATA)>
```

```
<?xml head version="1.0"?>
<!DOCTYPE head SYSTEM "head.dtd">
<head>
<body>ejemplo</body>
</head>
```

```
<?xml version="1.0"?>
<!DOCTYPE head [
    <!ELEMENT head (body)>
    <!ELEMENT body (#PCDATA)>
    ]>
    <head>
    <body>ejemplo</body>
    </head>
```

```
<?xml version="1.0"?>
<!DOCTYPE head SYSTEM "head.dtd"
[
<!ENTITY car "coche">
]>
<head>
<body>Ejemplo &car;</body>
</head>
```

DTD: Diseño y Validación

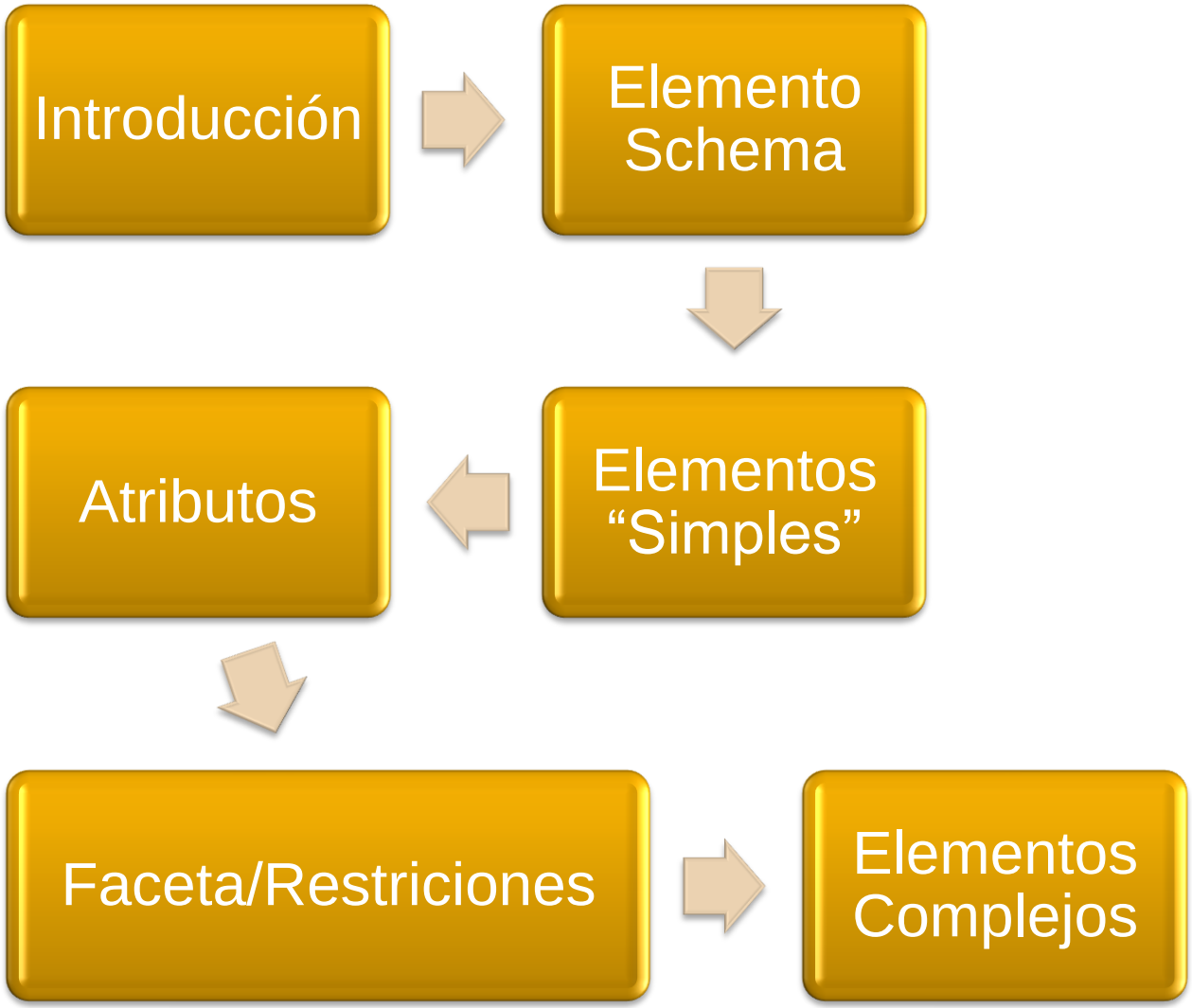
- Las DTDs externas son más fáciles de mantener, ya que los cambios se aplican automáticamente a todas sus instancias.
- Un documento será válido si cumple las restricciones que se indican en su DTD.
- Un documento será bien formato si los elementos están anidados correctamente, y si las entidades que referencia se han declarado.
- Un documento puede estar bien formado y ser no válido, ya que un documento XML puede no contener una declaración de tipo de documento.

DTD: Ejemplos

```
<?xml version="1.0"?>
 <!DOCTYPE agenda SYSTEM "ejemplo-agenda.dtd">
 <agenda>
<persona id="ricardo">
<nombre>Ricardo Borriquero</nombre>
<tlf>951345678</tlf>
</persona>
<persona id="eva">
<nombre>Eva Risto</nombre>
<tlf>955837659</tlf>
</persona>
</agenda>
```

```
<?xml version="1.0“ encoding="UTF-8"?>
 <!ELEMENT agenda (persona)+>
 <!ELEMENT persona (nombre, tlf)>
 <!ATTLIST persona id ID #REQUIRED>
 <!ELEMENT nombre (#PCDATA)>
 <!ELEMENT tlf (#PCDATA)>
```

XML Schema (xsd)



XML Schema (xsd): Introdución

- Son una sintáxis alternativa para las DTDs, propuesta inicialmente por Microsoft, …, etc.
- Utilizan la sintáxis propia de XML.
- Ventajas:
  - *Fáciles de aprender (se usa también XML).*
  - *Soportan tipos de datos: numéricos, fechas…*
  - *Procesables igual que los documentos XML.*
- Un esquema XML define la estructura válida para un tipo de documento XML (al igual que las DTD), es decir:
  - *Los elementos que pueden aparecer en el documento.*
  - *Los atributos que pueden utilizarse junto a cada elemento.*
  - *Cómo se pueden anidar los elementos (padres e hijos).*
  - *El orden en el que deben aparecer los elementos hijos de un mismo padre.*
  - *El número permitido de elementos hijos.*
  - *Si un elemento puede ser vacío o no.*
  - *Tipos de datos para elementos y atributos.*
  - *Valores por defecto y fijos para elementos y atributos.*

XML Schema (xsd): Introdución

- La propuesta inicial de Microsoft dio lugar a los llamados "esquemas XDR".
- Posteriormente, el W3C diseñó un modelo de esquemas que es la propuesta oficial y la que debemos conocer (llamados "esquemas XSD")
- XSD se publicó como una recomendación el 31 de marzo del 2001 (se considera oficial desde mayo)
- XSD es más complejo que otras alternativas anteriores, pero supuso un importante paso hacia adelante en la estandarización de XML

XML Schema (xsd): Introdución

- XDS permite una mayor precisión en la definición de tipos de datos mediante formatos y facetas

  Por ejemplo, la fecha:

  <date type="date">1999-03-11</date>

  ¿es el 11 de marzo o el 3 de noviembre?

- Los esquemas se definen como documentos XML, en un documento aparte con extensión .XSD
- En los documentos XML que se basen en ese esquema, incluiremos una referencia al archivo .XSD

XML Schema (xsd): Introdución

```
<?xml version="1.0“ encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "http://www.us.com/dtd/note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>
      Don't forget me this weekend!
</body>
</note>
```

```
<?xml version="1.0“ encoding="UTF-8"?>
<note  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation= “note.xsd">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>
                Don't forget me this weekend!
</body>
</note>
```

XML Schema (xsd): Introdución

```
<?xml version="1.0“ encoding="UTF-8"?>
<xsd:schema    xmlns:xsd="http://www.w3.org/2001/XMLSchema”>
    <xsd:element    name="note">
            <xsd:complexType>
                    <xsd:sequence>
                    <xsd:element name="to" type="xsd:string"/>
                    <xsd:element name="from" type="xsd:string"/>
                    <xsd:element name="heading" type="xsd:string"/>
                    <xsd:element name="body" type="xsd:string"/>
                    </xsd:sequence>
            </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

XML Schema (xsd): Elemento schema

- Los elementos utilizados en la creación de un esquema "proceden" del espacio de nombres:

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

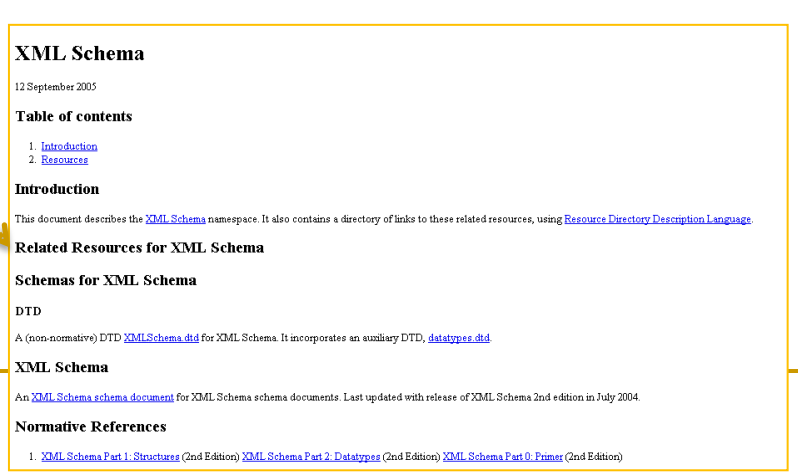
- El elemento *schema* es el elemento raíz del documento en el que se define el esquema:

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

…..

…..

</xsd:schema>



XML Schema

12 September 2005

Table of contents

1. Introduction
2. Resources

Introduction

This document describes the XML Schema namespace. It also contains a directory of links to these related resources, using Resource Directory Description Language.

Related Resources for XML Schema

Schemas for XML Schema

DTD

A (non-normative) DTD XMLSchema.dtd for XML Schema. It incorporates an auxiliary DTD, datatypes.dtd.

XML Schema

An XML Schema schema document for XML Schema schema documents. Last updated with release of XML Schema 2nd edition in July 2004.

Normative References

1. XML Schema Part 1: Structures (2nd Edition) XML Schema Part 2: Datatypes (2nd Edition) XML Schema Part 0: Primer (2nd Edition)

XML Schema (xsd): Elementos «Simples»

- Un elemento simple es un elemento que sólo puede contener texto (cualquier tipo de dato), pero no a otros elementos ni atributos
- Para definir un elemento simple, utilizamos la sintáxis:

```
<xsd:element name="xxx" type="yyy"/>
```

- Ejemplos:

```
<xsd:element name="apellido" type="xsd:string"/>
<xsd:element name="edad" type="xsd:integer"/>
<xsd:element name="fecNac" type="xsd:date"/>
```

XML Schema (xsd): Elementos «Simples»

- Los tipos de datos más utilizados son:
  - xsd:string
  - xsd:decimal
  - xsd:integer
  - xsd:boolean
  - xsd:date
  - xsd:time
- Un elemento simple puede tener un valor por defecto y un valor "fijo".
- Esto se indica mediante los atributos default y fixed.

```
<xsd:element name="color" type="xsd:string" default="red"/>
```

XML Schema (xsd): Atributos

- Los atributos se deben declarar de forma similar a los "elementos simples"
- Si un elemento puede ir acompañado de atributos, el elemento se deberá declarar como un elemento "complejo"
- Un atributo se declara de la siguiente forma:

  <xsd:attribute name="xxx" type="yyy"/>

  Ejemplo:

  <xsd:attribute name="idioma" type="xs:string"/>
- Los atributos tienen un tipo de dato: **xsd:string, xsd:decimal, xsd:integer, xsd:boolean, xsd:date, xsd:time**
- Los atributos pueden tener valores por defecto y valores fijos:

  <xsd:attribute name="idioma" type="xsd:string" default="ES"/>
- Por defecto, los atributos son opcionales.
- Para indicar que un atributo debe ser obligatorio, se debe añadir a su declaración "use"

  <xsd:attribute name="lang" type="xsd:string" use="required"/>
- El atributo use puede tomar el valor "optional" si el atributo no es obligatorio (opción por defecto)

XML Schema (xsd): Facetas/Restricciones

- *Las facetas o restricciones permiten restringir el valor que se puede dar a un elemento o atributo XML.*

- Mediante restricciones podemos indicar que un valor debe estar comprendido en un rango determinado, debe ser un valor de una lista de valores "cerrada", o debe ser mayor o menor que otro valor…

- Tipos de facetas o restriciones:
  - *Valor comprendido en un rango*
  - *El valor está restringido a un conjunto de valores posibles*
  - *Restringir el valor de un elemento a una serie de caracteres*
  - *Longitud de los valores de los elementos…*
  - *…..*

XML Schema (xsd): Facetas/Restricciones

```
<xsd:element name="age">
    <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="0"/>
            <xsd:maxInclusive value="100"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

XML Schema (xsd): Facetas/Restricciones

```
<xsd:element name="car">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Audi"/>
            <xsd:enumeration value="Golf"/>
            <xsd:enumeration value="BMW"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

XML Schema (xsd): Facetas/Restricciones

```
<xsd:element name="car" type="carType"/>

<xsd:simpleType name="carType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Audi"/>
        <xsd:enumeration value="Golf"/>
        <xsd:enumeration value="BMW"/>
    </xsd:restriction>
</xsd:simpleType>
```

XML Schema (xsd): Facetas/Restricciones

```
<xsd:element name="letter">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[a-z]"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

En este ejemplo, el elemento “letter” debe tomar como valor 1 letra minúscula (sólo 1)

XML Schema (xsd): Facetas/Restricciones

```
<xsd:element name="initials">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

En este ejemplo, el elemento “initials” debe tomar como valor 3 letras mayúsculas o minúscula (sólo 3)

XML Schema (xsd): Facetas/Restricciones

```
<xsd:element name="choice">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[xyz]"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

En este ejemplo, el elemento “choice” debe tomar como valor una de estas letras: x, y o z

XML Schema (xsd): Facetas/Restricciones

```
<xsd:element name="prodid">
    <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
            <xsd:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>
            </xsd:restriction>
     </xsd:simpleType>

</xsd:element>
```

XML Schema (xsd): Facetas/Restricciones

```
<xsd:element name="letter">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="([a-z])*"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

XML Schema (xsd): Facetas/Restricciones

```
<xsd:element name="password">
    <xsd:simpleType>
        <xsd:restriction base="xs:string">
            <xsd:pattern value="[a-zA-Z0-9]{8}"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

En este ejemplo, el valor del campo “password” debe ser 8 caracteres

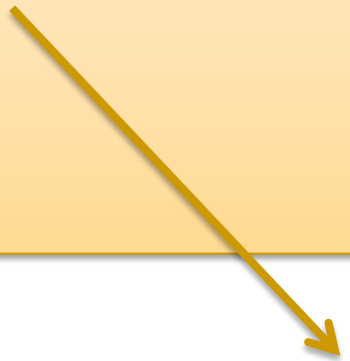
XML Schema (xsd): Facetas/Restricciones

```
<xsd:simpleType name="SKU">
    <xsd:restriction base="xsd:string">
            <xsd:pattern value="\d{3}-[A-Z]{2}"/>
    </xsd:restriction>
</xsd:simpleType>
```

SKU (Stock Keeping Unit): Código para identificar productos (p.ej. 976-FB)

XML Schema (xsd): Facetas/Restricciones

```
<xsd:simpleType name=“email">
    <xsd:restriction base="xsd:string">
            <xsd:pattern value=“\c{4,}[@]\c{2,}[.][a-zA-Z]{2,}"/>
    </xsd:restriction>
</xsd:simpleType>
```



{nim , max}

XML Schema (xsd): Facetas/Restricciones

```
<xsd:element name="password">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:length value="8"/>
            <xsd:pattern value="\d{2}[A-Za-z]{6}"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

Los elementos length, minLength y maxLength permiten indicar el número exacto, mínimo y máximo de caracteres que puede tener un valor de un elemento.

XML Schema (xsd): Facetas/Restricciones

```
<xsd:element name="password">
    <xsd:simpleType >
        <xsd:union memberTypes="rest1  rest2"/>
    </xsd:simpleType>
</xsd:element>
```

```
<xsd:simpleType name="rest1" >
</xsd:simpleType>
```

```
<xsd:simpleType name="rest2" >
</xsd:simpleType>
```

XML Schema (xsd): Facetas/Restricciones

enumeration	Establece una lista de valores “aceptados”
fractionDigits	Número de cifras decimales
length	Número de caracteres obligatorios
maxExclusive y maxInclusive	Valor máximo de un rango
minExclusive y minInclusive	Valor mínimo en un rango
maxLength y minLength	Número máximo y mínimo de caracteres permitidos
pattern	Define una secuencia de caracteres permitida
totalDigits	Número exacto de dígitos permitidos
whiteSpace	Indica cómo se deben de tratar los espacios en blanco

XML Schema (xsd): Elementos Complejos

- Son elementos que contienen a otros elementos hijos, o que tienen atributos.

- Se suelen dividir en 4 tipos:
  - *Elementos vacíos.*
  - *Elementos no vacíos con atributos.*
  - *Elementos con elementos hijos.*
  - *Elementos con elementos hijos y con "texto" o valor propio (como el contenido mixto de las DTD).*

XML Schema (xsd): Elementos Complejos

Ejemplos:

Caso 1.- *<product pid="1345"/>*

Caso 2.- *<food type="dessert">Ice cream</food>*

Caso 3.- *<description>Sucedió el <date>03.03.99</date> ….*
</description>

Caso 4.- *<employee>*
<firstname>John</firstname>
<lastname>Smith</lastname>
</employee>

XML Schema (xsd): Elementos Complejos

Para definir elementos complejos se utiliza la siguiente sintaxis:

```
<xsd:element name="employee">
    <xsd:complexType>
        <xsd:sequence>
        <xsd:element name="firstname" type="xsd:string"/>
        <xsd:element name="lastname" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

XML Schema (xsd): Elementos Complejos

- Podemos usar otra sintáxis para reutilizar la "definición" de los elementos hijos en varios elementos:

```
<xsd:element name="employee" type="personinfo"/>
<xsd:element name="student" type="personinfo"/>

<xsd:complexType name="personinfo">
    <xsd:sequence>
        <xsd:element name="firstname" type="xsd:string"/>
        <xsd:element name="lastname" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
```

En la declaración de elementos complejos, es posible utilizar un mecanismo de "herencia" para reutilizar o extender elementos definidos con anterioridad (ver la siguiente página)

```
<xsd:element name="employee" type="fullpersoninfo"/>

<xsd:complexType name="personinfo">
    <xsd:sequence>
        <xsd:element name="firstname" type="xsd:string"/>
        <xsd:element name="lastname" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="fullpersoninfo">
    <xsd:complexContent>
        <xsd:extension base="personinfo">
            <xsd:sequence>
                <xsd:element name="address" type="xsd:string"/>
                <xsd:element name="city" type="xsd:string"/>
                <xsd:element name="country" type="xsd:string"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

XML Schema (xsd): Elementos Complejos

- Para declarar un elemento vacío con atributos, se utilizará la siguiente sintáxis:

```
<xsd:element name="product">
    <xsd:complexType>
        <xsd:attribute name="prodid" type="xsd:positiveInteger"/>
    </xsd:complexType>
</xsd:element>

<product prodid="1345" />
```

XML Schema (xsd): Elementos Complejos

- Para declarar un elemento no vacío con atributos, y sin elementos hijos, se utilizará la siguiente sintáxis:

```
<xsd:element name="shoesize">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:attribute name="country" type="xsd:string" />
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
```

XML Schema (xsd): Elementos Complejos

- Para declarar un elemento con contenido "mixto", basta con añadir un atributo "mixed" al elemento xsd:complexType:

```
<xsd:element name="letter">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="orderid" type="xsd:positiveInteger"/>
            <xsd:element name="shipdate" type="xsd:date"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

```
<letter>Estimado cliente: <name>Juan Perez</name>. Su pedido número <orderid>1032</orderid> se enviará el día <shipdate>2001-07-13</shipdate>. </letter>
```

XML Schema (xsd): Elementos Complejos

- En los ejemplos anteriores hemos utilizado el elemento xsd:sequence como elemento hijo del elemento xsd:complexType.
- xsd:sequence indica que los elementos anidados en él deben aparecer en un orden determinado.
- Los esquemas XML nos ofrecen otras alternativas, además de **xsd:sequence**, para indicar cómo se deben tratar los elementos que aparecen anidados en un elemento complejo.
- Las opciones o "indicadores" son: **xsd:all y xsd:choice.**

XML Schema (xsd): Elementos Complejos

El indicador xsd:all indica que los elementos que contiene pueden aparecer en cualquier orden, pero como máximo sólo una vez.

```
<xsd:element name="person">
    <xsd:complexType>
        <xsd:all>
            <xsd:element name="firstname" type="xsd:string"/>
            <xsd:element name="lastname" type="xsd:string"/>
        </xsd:all>
    </xsd:complexType>
</xsd:element>
```

XML Schema (xsd): Elementos Complejos

- El indicador xsd:choice indica que puede aparecer sólo uno de los elementos que contiene

```
<xsd:element name="person">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element name="firstname" type="xsd:string"/>
            <xsd:element name="lastname" type="xsd:string"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
```

XML Schema (xsd): Elementos Complejos

- minOccurs, maxOccurs se utilizan para indicar el número máximo y mínimo de veces que puede aparecer un elemento hijo de un elemento complejo
- El atributo maxOccurs puede tomar el valor "unbounded", que indica que no existe ningún límite

```
<xsd:element name="person">
    <xsd:complexType>
        <xsd:sequence>
        <xsd:element name="full_name" type="xsd:string"/>
        <xsd:element name="child_name" type="xsd:string"
    maxOccurs="10"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

XML Schema (xsd): Elementos Complejos

- En esquemas XML también contamos con un modelo de contenido ANY, que permite incluir elementos no declarados inicialmente en el esquema.

```
<xsd:element name="person">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="firstname" type="xsd:string"/>
            <xsd:element name="lastname" type="xsd:string"/>
            <xsd:any minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

XML Schema (xsd): Elementos Complejos

- También contamos con un elemento que permite extender el número de atributos de un elemento:

```
<xsd:element name="person">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="firstname" type="xsd:string"/>
            <xsd:element name="lastname" type="xsd:string"/>
        </xsd:sequence>
        <xsd:anyAttribute/>
    </xsd:complexType>
</xsd:element>
```

XML Schema (xsd): Elementos Complejos

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:annotation>
  <xsd:documentation xml:lang="es">
   Esquema de hoja de pedido para Example.com.
   Copyright 2000 Example.com. Todos los derechos reservados.
  </xsd:documentation>
 </xsd:annotation>
<xsd:element name="empleado">
  <xsd:complexType>
    <xsd:sequence>
          <xsd:sequence>
                    <xsd:element name="nombre" type="xsd:string" />
                    <xsd:element name="fechaingreso" type="xsd:string" />
                    <xsd:element name="salario" type="xsd:string" />
          </xsd:sequence>
          <xsd:choice>
                    <xsd:element name="fe1" type="xsd:date" />
                    <xsd:element name="fe2" type="xsd:date" />
           </xsd:choice>
          <xsd:element name="comentario" type="xsd:string" minOccurs="1" maxOccurs="3"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

XML Schema (xsd): Elementos Complejos

```
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
  <xsd:element  name=”empleado”>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref=”nombre” />
        <xsd:element ref=”fechaingreso” />
        <xsd:element ref=”salario” />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name=”nombre” type="xsd:string" />
  <xsd:element name=”fechaingreso” type="xsd:date"/>
  <xsd:element name=”salario” type="xsd:double"/>
</xsd:schema>
```

Referencia a un elemento global

XML Schema (xsd)/DTD: Ejemplo


```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT report (toc, chapter+)>
<!ELEMENT toc (tocitem+)>
<!ELEMENT tocitem EMPTY>
<!ELEMENT chapter (sentence*)>
<!ELEMENT sentence (#PCDATA)>
<!ATTLIST tocitem
    chapter IDREF #REQUIRED
>
<!ATTLIST chapter
    id ID #REQUIRE
    title CDATA #RE
>
<!ATTLIST sentence
    ref CDATA #IM
>
```

Herramientas CASE

Xmlspy® Professional Edition.
XMetaL

XML Copy Editor es un editor de documentos XML libre (GPL 2.0) y multiplataforma cuya página web es http://xml-copy-editor.sourceforge.net/.
La última versión disponible actualmente (febrero de 2011) es la versión XML Copy Editor 1.2.0.7 (del 11 de diciembre del 2009).
La versión para Windows se puede descargar de SourceForge (6,28 MB). La versión para Ubuntu se puede instalar desde los repositorios oficiales de la distribución.

Ejercicios y Cuestiones