XML

Manual completo para el alumno

Documentos XML

- Contenidos:

  Introducción.

  Sintaxis de un documento XML.

  Arquitectura de manipulación de XML-data.

  Verificación de un documento XML.

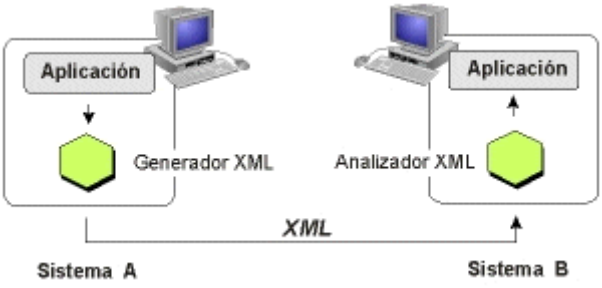
  1. Validación de documentos XML:
     a. Validación de documentos XML: DTD.
     b. Validación de documentos XML: XML Schema.
  2. Lenguajes de acceso a la información para documentos XML.
  3. Transformación de documentos XML.

XML es el acrónimo de eXtensible Markup Language (Lenguaje Extensible de Marcado).

• Lenguaje: conjunto de reglas que definen una sintaxis y una gramática. (XML es un metalenguaje que permite crear otros lenguajes)
• Marcado: método para escribir o incrustar metadatos (a través de marcas o etiquetas).
• Metadato: Es la información que podemos aportar sobre otro conjunto de datos.
(XML es un conjunto de reglas que describen cómo podemos escribir metainformación en un texto).

XML es una tecnología en realidad muy sencilla que tiene a su alrededor otras tecnologías que la complementan y la hacen mucho más grande y con unas posibilidades enormes y básicas para la sociedad de la información.

XML, con todas las tecnologías relacionadas, representa una manera distinta de hacer las cosas, más avanzada, cuya principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes.



Sintaxis de un documento XML.

Todo documento XML tendrá la siguiente **estructura**:

1.- El prólogo o Cabecera: contendrá información determinante sobre el resto de datos del documento, instrucciones de procesamiento (informa al intérprete encargado de procesar el documento de todos aquellos datos que necesita para realizar su trabajo), y estará compuesto por:

- La declaración del XML, <? … ¿>, donde se indicará obligatoriamente la versión de XML que se utiliza y, con carácter opcional, el conjunto de caracteres en el que está codificado el resto del documento y la autonomía del documento, seguido de cero o más comentarios o instrucciones de procesado.

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

  - version: indica la versión de XML (la más extendida sigue siendo la versión 1.0, aunque ya existe la versión 1.1
  - encoding: indica la codificación de caracteres.
  - standalone: indica si el documento está listo para procesarse independientemente o requiere de archivos externos (por defecto: no).
    standalone=”yes”: el documento XML es independiente.

- De manera opcional, también podemos encontrar la declaración del tipo de documento, que puede ser una referencia a una DTD externa o parte de una DTD, que será usada por el parser XML para comprobar la validez del documento durante el proceso de validación, seguida de cero o más comentarios o instrucciones de procesado.

```
<!-- Esto es un comentario en XML -->
<!DOCTYPE nombre SYSTEM "arch.dtd" >
```

  - Visualización: XML no dispone de una visualización concreta en el navegador puesto que refleja datos y no presentación, pero podemos establecer la forma de representar el documento mediante una hoja de estilos (CSS) o una hoja de transformaciones (XSLT).

```
<?xml-stylesheet type="text/css" href="ejemplo.css" ?>
<?xml-stylesheet type="text/xsl" href="transforma.css" ?>
```

2.- El cuerpo del documento (datos que se quieren procesar):

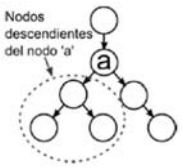
Está formado por un único elemento, que recibe el nombre de Elemento Raíz.
El Elemento Raíz contendrá el resto de elementos del documento XML con la información a tratar.

```
<Raíz>
    …
</Raíz>
```

Cualquier documento deberá contener forzosamente un elemento raíz, y un elemento a su vez podrá contener uno o más elementos. Esto nos permite poder representar los documentos XML mediante un árbol jerárquico, donde la raíz del mismo será el elemento raíz del cuerpo del documento.

▪ Elementos o Etiquetas anidadas: básicamente un elemento de XML-data es un contenedor de contenido. El elemento es la unidad básica del documento.
En el resto del documento se deben escribir etiquetas.
Un elemento o etiqueta es un grupo formado por una etiqueta de apertura, otra de cierre, y el contenido que hay entre ambas.
Las etiquetas se escriben anidas, unas dentro de otras, siempre respetando la sintaxis de etiqueta de apertura <etiqueta> y la etiqueta de cierre </etiqueta>. En los anidamientos de etiquetas deben cerrarse en sentido inverso al que se aperturaron.

```
<ETIQ1>...
        <ETIQ2>...</ETIQ2>
        …
</ETIQ1>
```



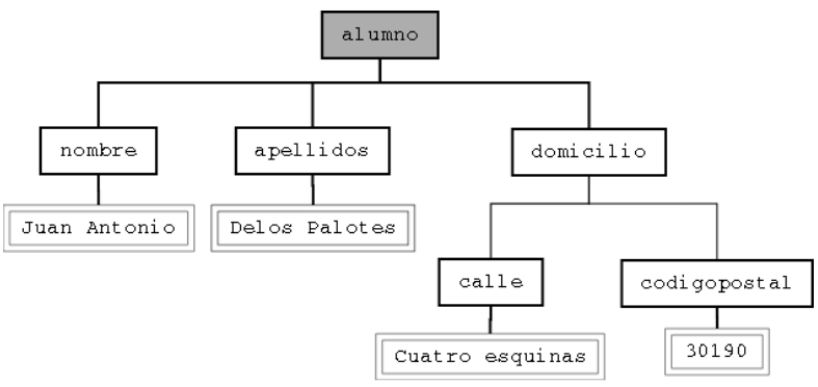
Todos los elementos deben tener un nombre que ha de cumplir unas normas:
- Puede contener letras, dígitos, guiones, guiones bajos, comas y dos puntos.
- Aunque puede empezar por letra, guión bajo o dos puntos, se aconseja que sea por letra.
- Son case-sensitive, así que se recomienda utilizar minúsculas para evitar confusión.
- No puede contener caracteres de espaciado.
- Los nombres que comienzan por las letras XML en cualquier combinación de mayúsculas y minúsculas se reservan para estandarización.

▪ Atributos: son un medio para añadir información adicional sobre el contenido de un elemento. Cualquier etiqueta puede tener atributos.
Los atributos son pares nombre-valor que se sitúan en la etiqueta de apertura del elemento. Los valores deben ir encerrados entre comillas simples o dobles.
Un atributo no puede repetirse dentro del mismo elemento (no pueden existir múltiples ocurrencias de un atributo en un mismo elemento).

```
<ETIQ atributo1="valor1" atributo2="valor2"...>
```

Para definir qué etiquetas y atributos debemos utilizar al escribir en XML tenemos que fijarnos en la manera de guardar la información de una forma estructurada y ordenada.
Ejemplo:

Representación jerárquica de un XML



```
<alumno>
        <nombre> Juan Antonio </nombre>
        <apellidos> Delos Palotes </apellidos>
        <domicilio>
                <calle> Cuatro esquinas </calle>
                <codigopostal> 30190 </codigopostal>
        </domicilio>
</alumno>
```

▫ Criterios para decidir si un dato debe estructurarse mediante elemento o atributo:

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.

▪ Contenido: los elementos pueden contener texto, otro elemento, o estar vacíos.

- Otro elemento: anidamiento de elementos formando la estructura jerárquica.
- Texto: representan la información de un elemento y pueden contener cualquier carácter, aunque para algunos hay que utilizar su entidad:

Carácter	Entidad	Carácter	Entidad
&	&amp;	'	&apos;
<	&lt;	"	&quot;
>	&gt;		

- Secciones CDATA (Character DATA): Un documento XML puede contener este tipo de secciones para escribir texto que no se desea que sea analizado (El procesador XML no interpreta este contenido como marcas sino como texto).

  <![CDATA[…]]>

  El contenido que no se analizará se introducirá en el espacio entre corchetes

  Por ejemplo:

  <![CDATA[Los caracteres < y & no pueden escribirse si no es como comienzo de marcas]]>

- Elementos sin contenido: existen la posibilidad de tener elementos sin contenido. Los elementos sin contenido pueden tener atributos y se abren y cierran en una sola etiqueta.

  Por ejemplo:

  <separador ancho=”80”/>

▪ Entidades: En XML se pueden definir entidades y utilizarlas para reutilizar información, para insertar contenido.

<!ENTITY nombre "valor">

Una entidad se define mediante una etiqueta que comienza por <!ENTITY y termina por > y contiene el nombre y el valor de la entidad.

Para hacer referencia a una entidad se escribe sin espacios intermedios el carácter "&", el nombre de la entidad y el carácter ";"

&nombre;

Al abrir el documento XML el procesador sustituye la referencia a la entidad por su valor.

Por ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE autor [
    <!ENTITY yo "Nombre Apellidos">
    <!ELEMENT autor (#PCDATA)>
    ]>
<autor>&yo;</autor>
```

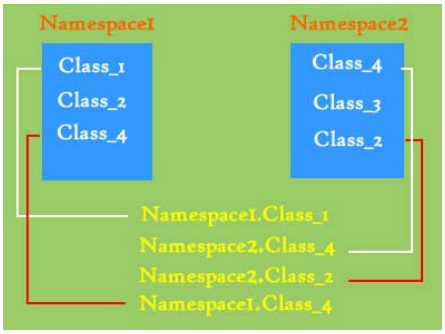
En la cabecera se ha definido la entidad "yo", cuyo valor es Nombre Apellidos".

Al procesar el archivo XML se sustituirá la llamada a la entidad "&yo" por el valor de la entidad "Nombre Apellidos".

3.- Epílogo de un documento XML. (opcional)

Está formado por cero o más comentarios o instrucciones de procesado.

▪ Espacios de Nombres XML o XML Namespaces.



Un espacio de nombres XML es un grupo de nombres que comparten un mismo contexto o propósito, agrupados bajo un nombre único.

(En general, en informática, un namespace o espacio de nombres es un medio para organizar clases dentro de un entorno, agrupándolas de un modo más lógico y jerárquico).

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 espacio de nombres XML no necesita que su vocabulario sea definido, aunque es una buena práctica utilizar un DTD o un esquema XML para definir la estructura de datos en la ubicación URI del espacio de nombres).

Los espacios de nombres se usan para combinar vocabularios y facilitan la incorporación de elementos no previstos inicialmente. Los espacios de nombres se crearon para que no existieran colisiones entre los diferentes módulos de software que eran capaces de reconocer las marcaciones (etiquetas y atributos) del lenguaje XML ya que los diferentes documentos que contienen marcaciones de distintas fuentes independientes entre sí, solían tener problemas de reconocimiento cuando habían sido creados para otros programas de software que, sin embargo, utilizaban el mismo tipo de elemento o nombre de atributo. Una de las motivaciones para esta modularidad es que, si existe un conjunto de marcaciones disponibles, que son entendibles y para las que existe software útil disponible, es mejor la reutilización de estas marcaciones que el hecho de reinventar unas nuevas. Así, se consideró que las construcciones de documentos debían tener nombres universales, cuyo ámbito se extendiera más allá del documento que las contuviera. La especificación Namespaces XML describe un mecanismo, los espacios de nombres XML, que lleva a cabo esta misión.

Los espacios de nombres (o namespaces) 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.

La especificación del espacio de nombres es un mecanismo que permite evitar los conflictos de nombres cuando en un documento XML existen elementos o atributos con el mismo nombre, pero diferente definición.
XML namespace, nos proporciona un mecanismo para que podamos indicar elementos de otros lenguajes XML (otros esquemas XML) o tipos de documentos, en nuestros propios XMLs.
De esta forma se pueden crear documentos XML de forma uniforme, sin duplicar nombres con significados distintos.

Por ejemplo, el elemento capital puede ser utilizado para denominar la capital de un estado o para indicar el importe monetario de una operación. Podríamos crear un espacio de nombres para adoptar dos soluciones:
1.- Podríamos crear un espacio de nombres para nuestros documentos en el que establezcamos que capital sea utilizado para almacenar el importe monetario de una operación y capitalestado para almacenar la capital de un estado, evitando que en distintos documentos se utilice el nombre capital para distintas finalidades.
(Si en una empresa se tienen que crear distintos documentos XML con funcionalidades distintas, pero se quiere crear un criterio uniforme a la hora de establecer nombres, de tal forma que un nombre de elemento o atributo signifique lo mismo en distintos documentos XML, aunque la funcionalidad del documento sea distinta).
2.- También podríamos utilizar el mismo nombre y hacer referencia a su distinto significado mediante un prefijo en el que establezcamos la diferencia a la hora de su uso: continuando con el ejemplo anterior c:capital para referenciarnos a capital de estado e i:capital para referenciarnos al importe monetario de una inversión. En el documento XML haríamos referencia a cada elemento no solo por su nombre, sino

añadiendo el prefijo establecido. El documento XML se validaría mediante la distinción de los elementos por su prefijo.

- Declaración: Podemos usar XML namespaces, añadiendo el atributo **xmlns**, a cualquier elemento de nuestro archivo, y especificando cómo su valor una URI.
  <elemento xmlns:prefijo="URI_espacio_nombres">
  prefijo: sirve como referencia. Puede ser cualquier texto, pero se recomienda que sea descriptivo.
  URI: debe ser único, aunque no es más que un identificador que no se comprueba mediante conexión alguna.

- Utilización: se antepone el prefijo definido en la declaración y dos puntos (:) al nombre del elemento o atributo.
  <prefijo:elemento prefijo:atributo="valor">
  El espacio de nombres puede ser referenciado anteponiendo su prefijo en el elemento en que se ha declarado y en sus descendientes.
  También se puede declarar un espacio de nombres nuevo para un elemento descendiente de otro en el que ya se había declarado un espacio de nombres.

Ejemplo:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xsl:stylesheet version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:output method="text" encoding="iso-8859-1"/>
        <xsl:template match="miDocumento">
                <xsl:apply-templates />
        </xsl:template>
</xsl:stylesheet>
```

Ejemplo:

```
<h:html xmlns:xdc="http://www.xml.com/libros"
         xmlns:h="http://www.w3.org/TR/html5/">
        <h:head> <h:title>Ficha del libro</h:title>
        </h:head>
        <h:body>
           <xdc:ficha_libro>
                <xdc:titulo>XML Edición 2012</xdc:titulo>
                <h:table>
                        <h:tr> <h:td>Autor</h:td><h:td>Precio</h:td> </h:tr>
                        <h:tr>
                                <h:td> <xdc:autor>M.A. Acera</xdc:author> </h:td>
                                <h:td> <xdc:precio>39.95</xdc:precio></h:td>
                        </h:tr>
                </h:table>
           </xdc:ficha_libro>
        </h:body>
</h:html>
```

Los atributos pueden pertenecer al mismo espacio de nombres al que pertenece el elemento al que están asociados o a otro diferente.

Los atributos que no tienen ningún prefijo no pertenecen a ningún espacio de nombres.
Dos atributos con el mismo nombre, pero distinto prefijo son diferentes y pueden estar asociados al mismo elemento.

Ejemplo:

```
<emp:empleado xmlns:emp="empresa:espacios:emp">
        <emp:datosPersonales trabajo:empno="1234"
           xmlns:trabajo="empresa:espacios:trabajo">
              <emp:departamento depto="10">Ventas</emp:departamento>
              <emp:nombre>Juan</emp:nombre>
              <emp:apellido>Pérez</emp:apellido>
        </emp:datosPersonales>
</emp:empleado>
```

- Espacio de nombres por defecto: Cuando en la declaración de un espacio de nombres no se especifica un prefijo estamos definiendo un espacio de nombres por defecto y su ámbito de aplicación se extiende al elemento en que se ha declarado y a todos sus descendientes, pero no a sus atributos.

Ejemplo:

```
<empleado xmlns="empresa:espacios:emp"
            xmlns:trabajo="empresa:espacios:trabajo">
        <datosPersonales trabajo:empno="1234" empno="e_1234">
              <departamento depto="10">Ventas</departamento>
              <nombre>Juan</nombre>
              <apellido>Pérez</apellido>
        </datosPersonales>
</empleado>
```

Se puede desasignar un elemento de un espacio de nombres por defecto asignando el valor vacío al atributo xmlns:

```
<elemento xmlns="">
```

Ejemplo:

```
<empleado xmlns="empresa:espacios:emp"
            xmlns:trabajo="empresa:espacios:trabajo">
        <datosPersonales trabajo:empno="1234" empno="e_1234">
              <departamento xmlns="" depto="10">Ventas</departamento>
              <nombre>Juan</nombre>
              <apellido>Pérez</apellido>
        </datosPersonales>
</empleado>
```

Ejemplos documentos XML:

Ejemplo: *documento XML para almacenar la información de notas de escritorio.*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<nota fecha=”12/11/99”>
        <para>Elisa</para>
        <de>Pedro</de>
        <titulo>Recordatorio</titulo>
        <cuerpo>No olvides nuestra cita</cuerpo>
</nota>
```

Ejemplo: *documento XML para almacenar la información de notas de escritorio.*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<mensajes>
        <nota id="001">
                <para>Elisa</para>
                <de>Pedro</de>
                <titulo>Recordatorio</titulo>
                <cuerpo>No olvides nuestra cita</cuerpo>
        </nota>
        <nota id="002">
                <para>Juan</para>
                <de>Francisco</de>
                <titulo>Cita</titulo>
                <cuerpo>Quedamos a comer en el Restaurante de abajo</cuerpo>
        </nota>
</mensajes>
```

Ejemplo: *documento XML para almacenar la información de distintos países, conteniendo su nombre y su capital*

```
<?xml version="1.0" encoding="UTF-8"?>
<paises>
        <pais>
                <nombre>España</nombre>
                <capital>Madrid</capital>
        </pais>
        <pais>
                <nombre>Francia</nombre>
                <capital>París</capital>
        </pais>
</paises>
```

Ejemplo: *el mismo documento, pero almacenando la información de distinta forma, haciendo uso de atributos en lugar de elementos*

```
<?xml version="1.0" encoding="UTF-8"?>
<paises>
        <pais nombre="España" capital="Madrid" />
        <pais nombre="Francia" capital="París" />
</paises>
```

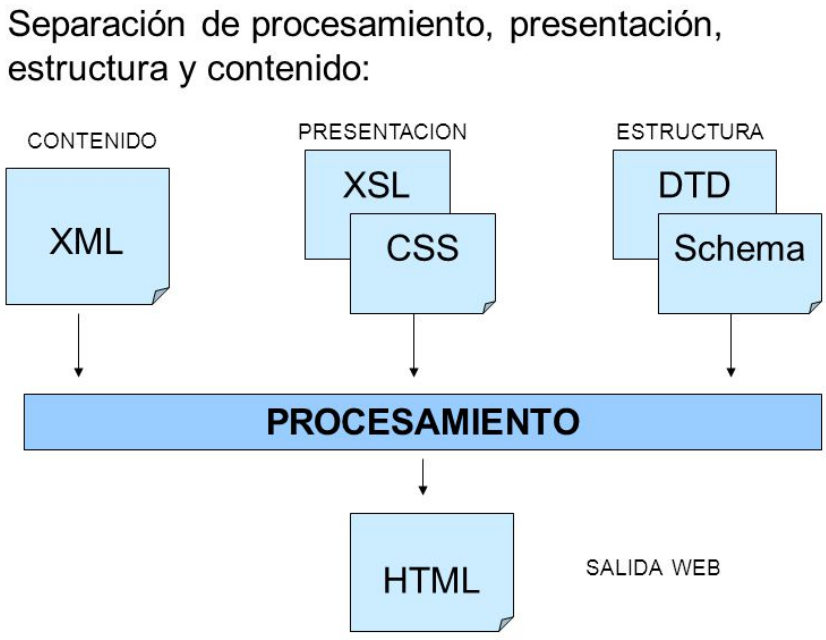
Ejemplo: *Listado de personal autorizado para el acceso. (El elemento departamento y responsable son opcionales. El elemento responsable está relacionado con el id de otro elemento)*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- LISTADO DE PERSONAL AUTORIZADO -->
<personal>
        <persona id="01">
                <nombre>&amp;quot; Directora &amp;quot; Nerea</nombre>
                <apellido>Fernández</apellido>
                <direccion>Gran Vía, 5</direccion>
                <matricula>0 &amp;#8364;</matricula>
        </persona>
        <persona id="100">
                <nombre>Isabel</nombre>
                <apellido>García</apellido>
                <direccion>Ancha, 3</direccion>
                <matricula>800 &amp;#8364;</matricula>
                <departamento>Administración</departamento>
                <responsable>01</responsable>
        </persona>
        <persona id="101">
                <nombre>Fernando</nombre>
                <apellido>Álvarez</apellido>
                <direccion>Mayor, 2</direccion>
                <matricula>800 &amp;#8364;</matricula>
                <departamento>Administración</departamento>
                <responsable>01</responsable>
        </persona>
</personal>
```

Ejemplo: *Documento XML validado con dtd, asociado a hojas de estilo y enlazado a transformación. El documento XML almacena información de un libro (título, catálogo, contenido, proceso de compra y copyright).*

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet href="librohtml.xsl" type="text/xsl"?>
<?xml-stylesheet href="librowml.xsl" type="text/xsl" media="wap"?>
<?cocoon-process type="xslt"?>

<!-- Referencia la DTD -->
<!DOCTYPE Libro SYSTEM "libro.dtd">

<!-- Comienza el contenido -->
<Libro xmlns="http://www.libros.com/libros/xml"
        xmlns:Catalogo="http://www.libros.com/catalogo">

        <Titulo>XML</Titulo>
        <Catalogo:Seccion>Lenguajes</Catalogo:Seccion>
        <Catalogo:SubSeccion>XML</Catalogo:SubSeccion>
        <Contenido>
```

```
            <Capitulo materia="XML">
                <Tema>Introducción</Tema>
                <subcapitulo1 apartados="7">Qué es</subcapitulo1>
                <subcapitulo1 apartados="3">Cómo se usa</subcapitulo1>
            </Capitulo>
             <Separacion/>
            <Capitulo materia="XML">
                <Tema>Creando XML</Tema>
                <subcapitulo1 apartados="0">Un documento XML
                </subcapitulo1>
                <subcapitulo1 apartados="2">La cabecera</subcapitulo1>
                <subcapitulo1 apartados="6">El contenido</subcapitulo1>
            </Capitulo>
        </Contenido>

         <Compra>
            <![CDATA[
                <Paso 1>Encontrar una librería.
                <Paso 2>Encontrar el libro.
                <Paso 3>Comprar el libro.
            ]]>
        </Compra>

        <Copyright>&EditorialCopyright;</Copyright>

</Libro>
```



Arquitectura de manipulación de XML-data.

Existen dos capas de cliente:

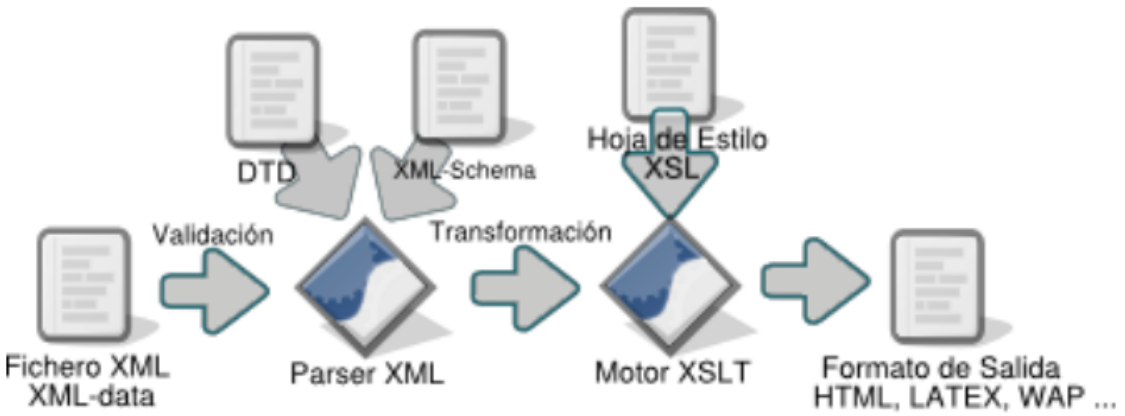
• La primera de estas capas es el **procesador XML** (o parser XML). El parser XML debe asegurarse de que el fichero XML cumple completamente con la normativa del lenguaje, proceso que se conoce como validación.

Uno de los objetivos en el diseño de XML era la Extensibilidad: que cada uno pudiera definir los tags del fichero XML como necesitase, sin limitar para nada el conjunto y la estructura de estos. La consecuencia directa de ello es, que podemos definir nuestro propio lenguaje de marcado, pero deberíamos poder comprobar que un determinado fichero cumple con las reglas que nos acabamos de inventar: XML contempla este hecho y nos permite especificar la estructura de nuestros lenguajes de marcado en ficheros independientes, llamados DTDs o XML-schemas. De manera opcional, podemos decirle al parser XML que nos compruebe si un fichero XML, es correcto de acuerdo con las reglas de formación de uno de estos ficheros y él, así lo hará dentro del proceso de validación del XML.

• La segunda de estas capas es el **procesador XSLT**, que a partir de un fichero XML validado es capaz de realizar transformaciones sobre este, de manera obtengamos la presentación del XML-data en el formato que deseemos: A esto se le conoce como **procesado o transformación del XML**.

El procesado del XML es necesario si queremos separar la información en sí de su presentación, puesto que resultaría muy incómodo leer los documentos en el XML plano, y siempre es preferible leerlos en PDF, HTML, en un móvil WAP, en papel, etc.
Para realizar estas transformaciones, la especificación propone el uso de hojas de estilo XSLs, que serán para el XML, como CSS para el HTML. En las hojas de estilo se especificarán las transformaciones que deben aplicarse sobre cada tag de nuestro documento XML, para obtener su representación en el formato de salida.

Arquitectura 2 capas para trabajo con XML



Verificación de un documento XML.

Siempre hay que **verificar que un documento XML** debe:

1) Estar **bien formado**: Los documentos bien formados aseguran que las reglas de XML se cumplen y que no hay ninguna incoherencia al usar el lenguaje.
(Un documento XML bien formado cumple las reglas de sintaxis definidas en XML).

2) Ser **válido**: Un documento XML es válido si ha pasado las reglas definidas en otro documento llamado documento de validación. De modo que, un documento XML que sea válido deberá indicar qué plantilla de reglas utiliza y deberá cumplirlas.
(Cumplirá las reglas establecidas en el documento de validación en cuanto a qué elementos y atributos pueden aparecer, en qué orden, qué valores pueden tomar, etc.)

   Por ejemplo, supongamos que disponemos de un servicio de Internet que permite informar sobre el tiempo que hace en nuestra zona. Supongamos también que entregamos esa información en XML. Para ello podríamos elaborar un XML con el tiempo de cada día y desde Internet cuando se solicite un día concreto, entregar el XML correspondiente.
   Para que ese servicio sea útil, la estructura del XML de cada día debe ser igual, de otro modo los servicios que requieren nuestra información no funcionarían adecuadamente.

   Parece que la validación supone un problema, pero en realidad es una ventaja; con la validación tenemos la seguridad de que los documentos cumplen unas reglas más concretas y de esa forma es fácil establecer un protocolo en las empresas para sus documentos. En definitiva, la validación permite establecer lenguajes propios de marcado para nuestros documentos.

   Para definir las reglas de validación de documentos XML utilizaremos: DTD (técnica más compatible) y XML Schema (técnica más coherente), aunque existen otras técnicas no tan avanzadas o difundidas.

Hay que distinguir entre un documento validado y un documento bien formado:
- El documento bien formado es simplemente el que cumple todas las reglas de XML para formar documentos.
- El documento validado debe, además, adaptarse al patrón que le marca la DTD o el Esquema.

Todo documento válido está bien formado, pero no todo documento bien formado es válido.