Taller de <?XML>

Prof. Hilda Contreras
Departamento de Computación
Escuela de Ingeniería de Sistemas
ULA

Contenido

1. Introducción
2. Preparándonos para trabajar con XML
3. Documentos XML bien formados
4. Documentos XML presentación y estructura
5. Usos de XML

1 - Introducción

1. Historia
2. Qué es XML? y Qué NO es XML?
3. Porqué XML?
4. Conceptos básicos
5. Ejemplos de usos

1.1. Historia

- Internet (1984)
- SGML (1986) gestación desde principios de los años 70
- Multimedia (1986)
- PDF (1992)
- NSF DLI National Science Foundation's Digital Libraries Initiative (1994)
- World Wide Web Consortium - **W3C** (1994)
- WWW (1994)
- XML (1998)

1.1. Historia

World Wide Web Consortium - W3C (1994)

http://www.w3c.org

- Constituido con el objetivo de desarrollar protocolos comunes para Internet
- Consorcio de industrias internacionales: MIT (EEUU), INRIA (Francia) y Keio University (Japón)
- Soporte oficial del DARPA (EEUU) y La Comisión Europea

1.2. Qué es XML?

- Es un subconjunto de SGML (Standard Generalized Markup Language) simplificado y adaptado a Internet
- Extensible Markup Language (XML), es más que un lenguaje de **marcado**
- Es un **meta-lenguaje**: es un lenguaje para definir lenguajes

1.2. Qué es XML?

"Lenguaje de **marcado"** basado en marcas:

- **Marcas** [mark-up] construcciones con etiquetas. Comienzan con "<" y terminan con ">".
- **Datos** resto de contenido del documento que se encuentra entre marcas
- Ejemplo:

  <autor>Deepak Chopra**</autor>**

1.2. Qué es XML?

“**Meta-lenguaje”**:

Información (dato): Deepak Chopra

Meta-información (marca): <autor>

- Describir otros lenguajes
- Crear etiquetas propias

1.2. Qué NO es XML?

- NO es una versión mejorada de HTML
- NO es un lenguaje para hacer mejores páginas Web
- NO es un lenguaje sustituto de HTML
- NO es un lenguaje difícil

1.3. Por qué XML?

- Es un estándar internacional reconocido por W3C (1998)
- Su utilización es libre y abierta
- Permite la utilización de múltiples **alfabetos** en diferentes plataformas
- Fácil procesamiento (reconocimiento, generación y transformación)
- Separa el contenido de los datos y de su presentación

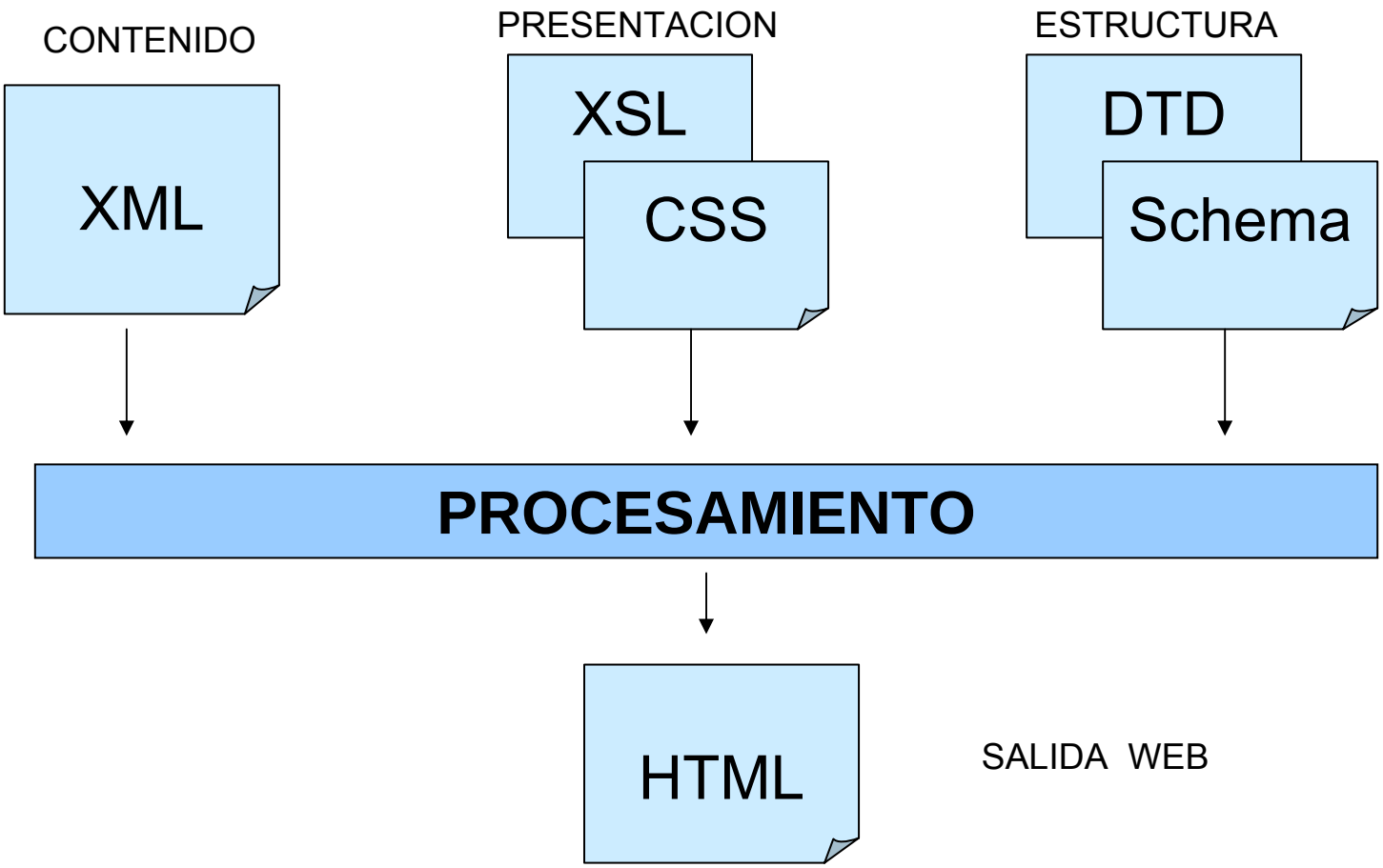
1.4. Conceptos básicos

Separación lógica de **contenido**, descripción de la **estructura** y **presentación**. Esto permite máxima independencia y flexibilidad.

- *Contenido:* datos. Documento XML.
- *Estructura:* reglas de estructura de los datos. DTD, XML Schema, etc.
- *Presentación:* el formato para mostrar la información. Hojas de estilo

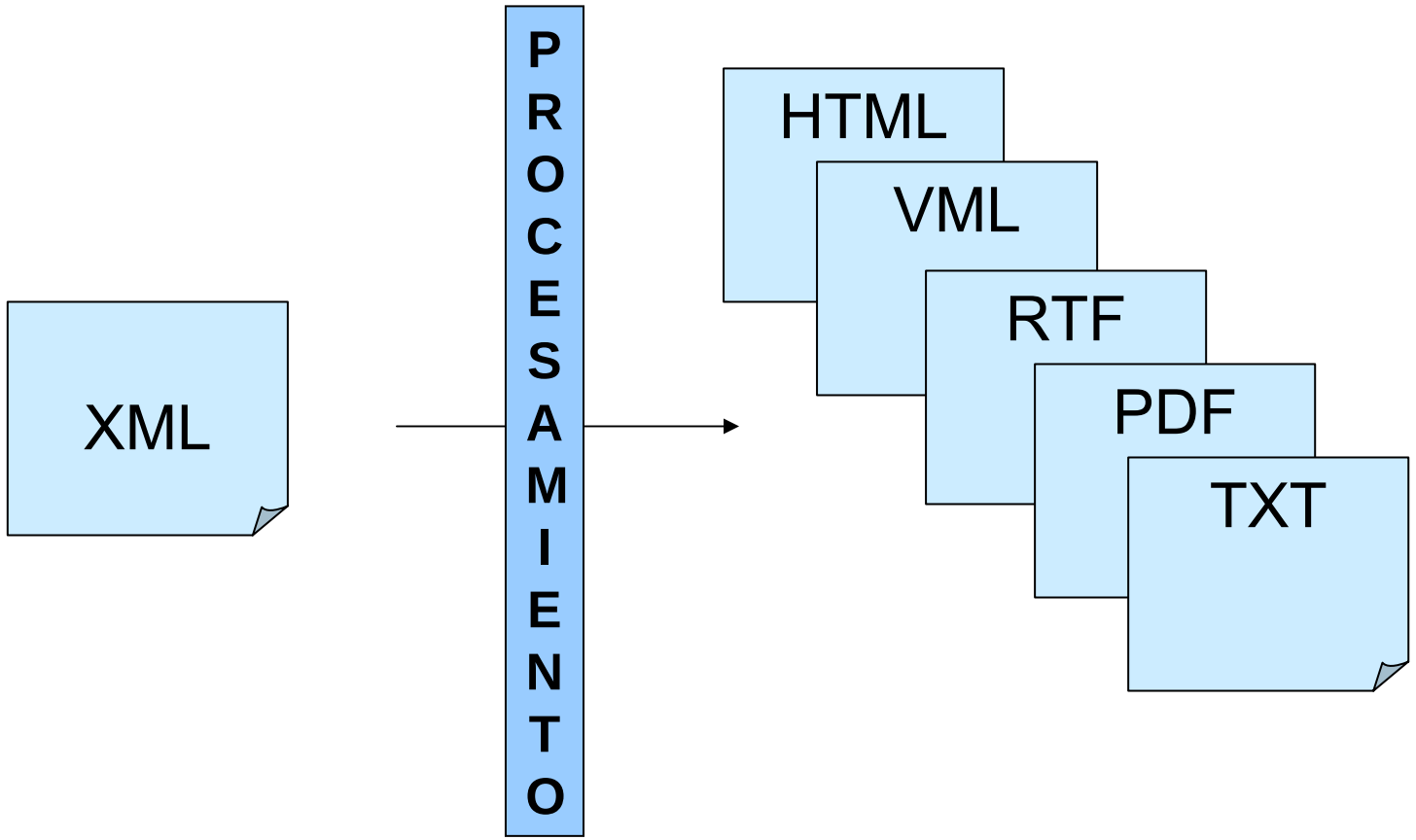
1.4. Conceptos básicos

Separación de procesamiento, presentación, estructura y contenido:



1.4. Conceptos básicos

Muchas formas de presentación a partir de un documento XML:



1.5. Ejemplos de Uso

- Un Ejemplo de HTML:

<B>Deepak Chopra

<B>El sendero del Mago</B>

precio: Bs. 30.000

1.5. Ejemplos de Uso

- El mismo Ejemplo en XML:

```
<libro>
    <autor>Deepak Chopra</autor>
    <titulo>El sendero del Mago</titulo>
    <precio moneda="bolivares">30.000</precio>
</libro>
```

Comentar las diferencias!.

2 - Preparándonos para trabajar con XML

1. Qué es un archivo XML?
2. Herramientas necesarias
3. Editando un archivo XML
4. Visualizando un archivo XML

PRÁCTICA #1: Edición y visualización de un archivo XML

2.1. Qué es un archivo XML?

- Un archivo de texto ASCII
- Un archivo con extensión “.xml”
- Un archivo cuyo contenido sigue las reglas sintácticas de XML

2.2. Herramientas necesarias

- Un Editor de Texto
- Un Explorador o Navegador de Internet **(*)**
- Un herramienta de procesamiento y transformación para XML o **parser (*)**

2.2. Herramientas necesarias

Parser: Analizador sintáctico

Procesa el contenido de un archivo XML para:

- Validar (Reconocimiento)
- Transformar

2.2. Herramientas necesarias

Parser

- Pueden incluir validación o no
- Pueden realizar transformaciones o no
- Pueden exponer la información de diferentes formas (DOM, SAX)
- Existen para la mayoría de lenguajes y plataformas de desarrollo (VB,Php,Perl,Java, etc.)

2.2. Herramientas necesarias

Lista de Parser:

- **Xerces (Apache)**
- **XML4J (IBM)**
- **Crimson (Apache)**
- **Project X (Sun Microsystems)**
- **MSXML (Microsoft)**
- **XP (James Clark)**
- **Ælfred (Microstar Software)**
- **Lark/Larval (Tim Bray)**
- **XJ (Data Channel)**

2.2. Herramientas necesarias

Usaremos el parser de PHP

- Realiza validaciones contra DTD
- Realiza transformaciones con soporte XSL
- Gratuito y redistribuible libremente
- Esta basado en DOM nivel 1

2.3 – Editando un archivo XML

PRÁCTICA #1:

- Ejecutar cualquier editor de texto
- Escribir este texto:

```
<libro>
<autor>Deepak Chopra</autor>
<titulo>El sendero del Mago</titulo>
<precio moneda="bolivares">30.000</precio>
</libro>
```

- Guardar como “libro.xml”

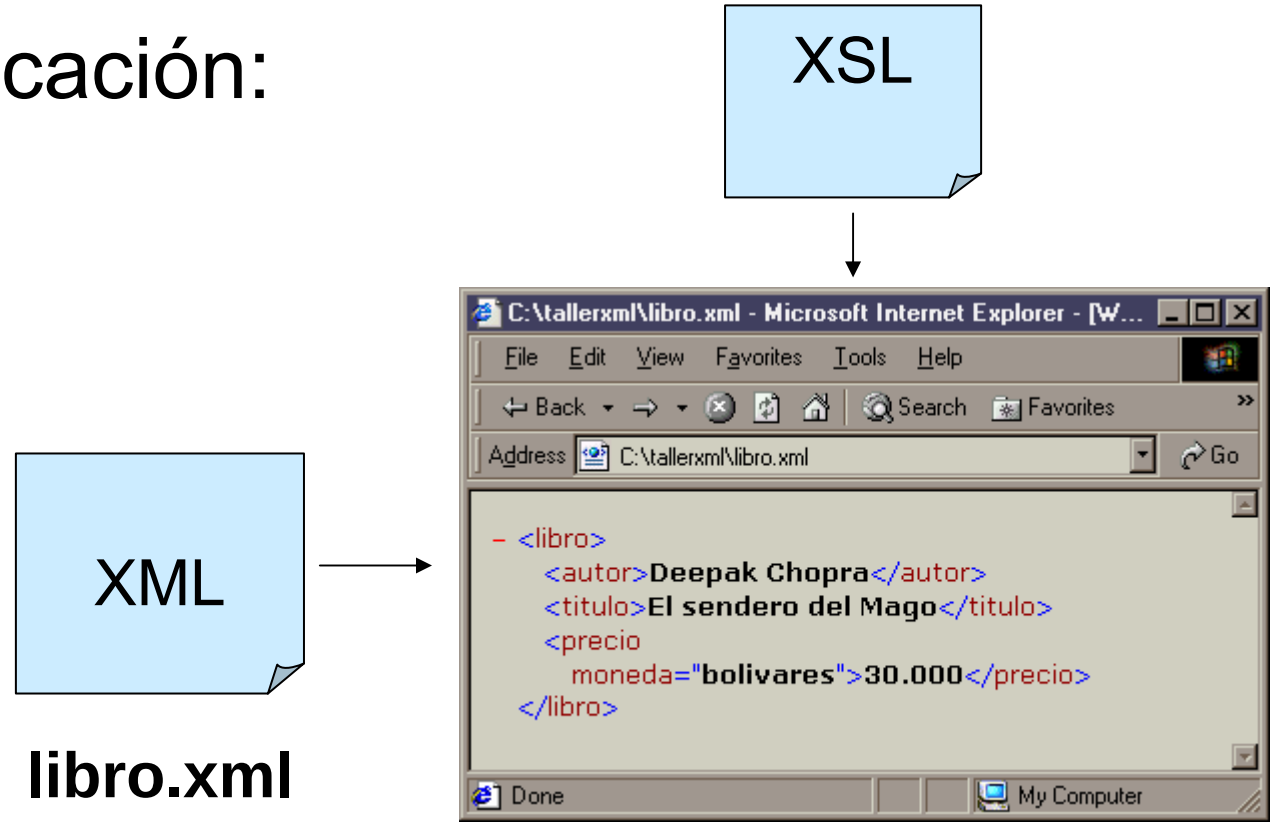
2.4 – Visualización de un archivo XML

PRÁCTICA #1:

- Ejecutar un navegador de Internet
- Abrir el archivo libro.xml
- Opciones: File – Open – Browse
- Observar la forma en que se muestra el archivo en el navegador.

PRÁCTICA #1

Explicación:



3 - Documentos XML bien formados

1. Reglas sintácticas XML
2. Documentos bien formados
3. Espacios de nombres XML (namespaces)

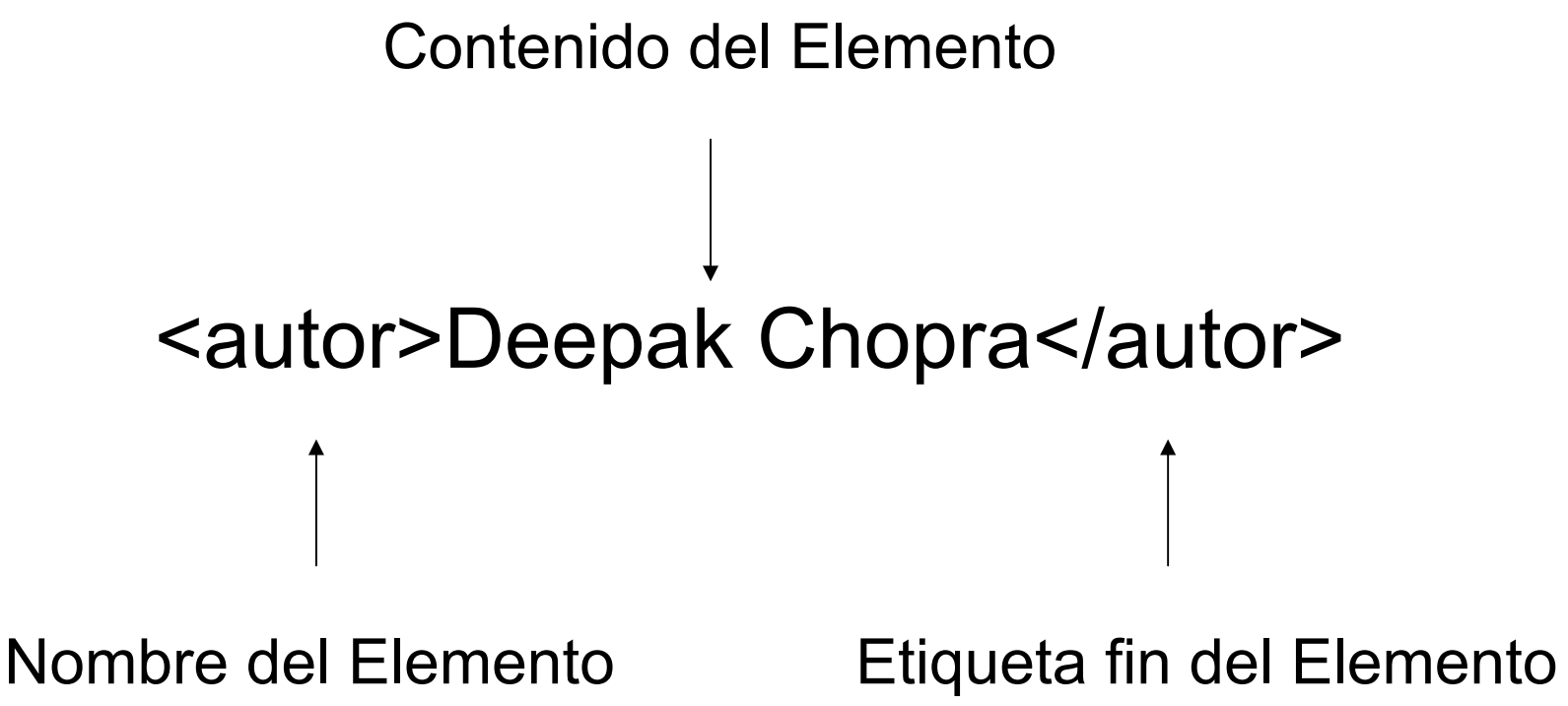
PRÁCTICA #2: Documentos XML bien formados

Referencia:

http://www.programacion.com/html/xml/principal.htm

3.1. Reglas sintácticas XML

- **Elemento** y Contenido



3.1. Reglas sintácticas XML

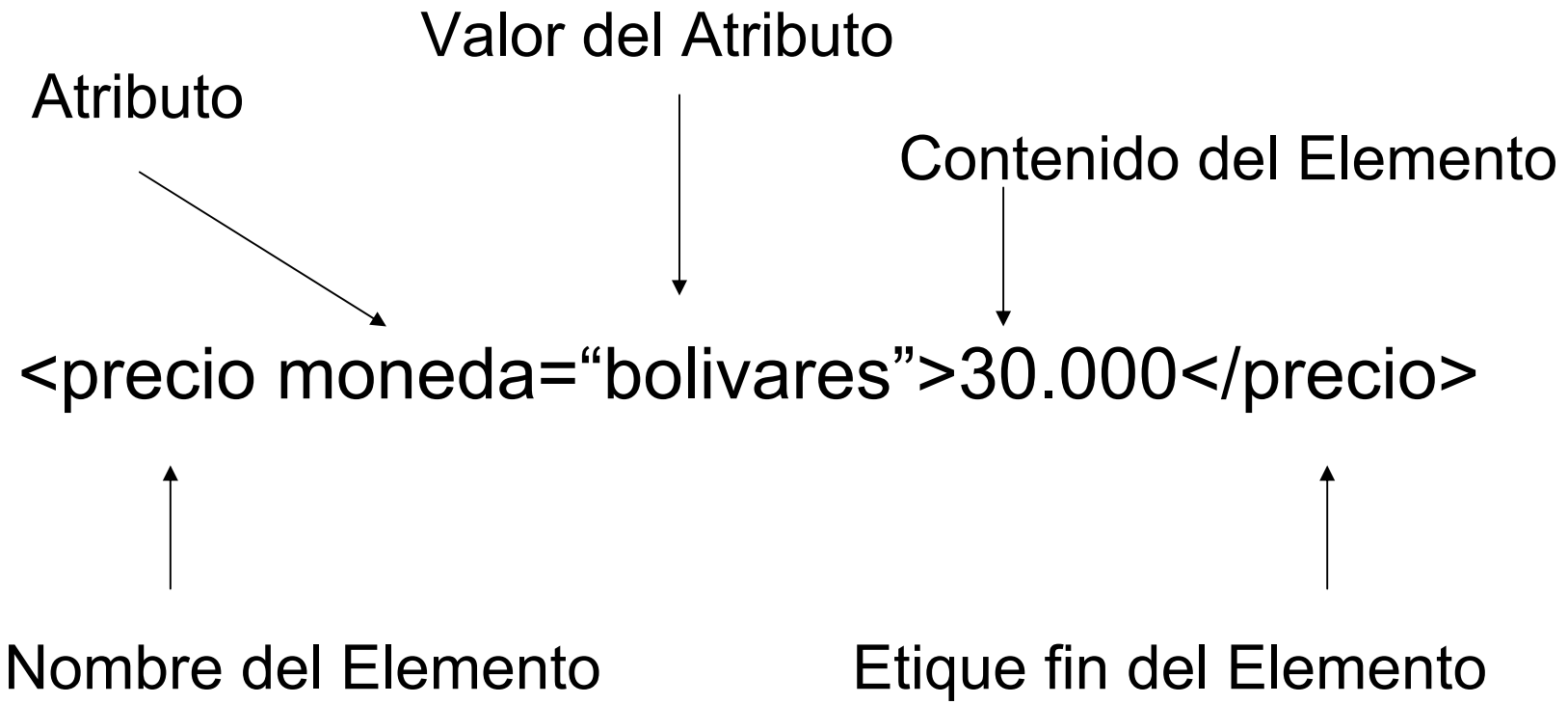
Cómo debo escribir los **Elementos**?

- Siempre comienzan con una <etiqueta>
- Siempre termina con una </etiqueta> del mismo nombre
- Elemento vacío <etiqueta/> o

  <etiqueta> </etiqueta>
- Es “case-sensitive”, es decir <autor> no es igual a <Autor>
- No se permiten espacios en blanco ni saltos de línea en el nombre de etiqueta

3.1. Reglas sintácticas XML

- **Atributo**



3.1. Reglas sintácticas XML

Cómo debo escribir los **Atributos**?

- Los atributos son un par de: nombre_atributo=“valor atributo”
- Los valores de los atributos siempre deben estar entre comillas simples (‘) o dobles (“).
- Si se permiten espacios en blanco en el valor del atributo.

3.1. Reglas sintácticas XML

Cómo debo escribir los **nombres** de atributos y elementos?

- Empezar por una letra
- Continuar con letras, dígitos, guiones, rayas, puntos o dos puntos.
- No se permiten espacios en blanco
- No usar la palabra “XML” como comienzo de un nombre

3.1. Reglas sintácticas XML

Cómo debo escribir el contenido de la información?:

- No usar **entidades predefinidas**
- Usar el conjunto de caracteres según la codificación especificada (**encoding**).
- Si deben usarse **entidades predefinidas** identificar como **CDATA**

3.1. Reglas sintácticas XML

Entidades Predefinidas: XML 1.0 define 5 entidades de caracteres especiales:

<	**<**
>	**>**
&	**&**
‘	**'**
“	**"**

<temperatura>< 0</temperatura>

3.1. Reglas sintácticas XML

Sección CDATA (Character Data)

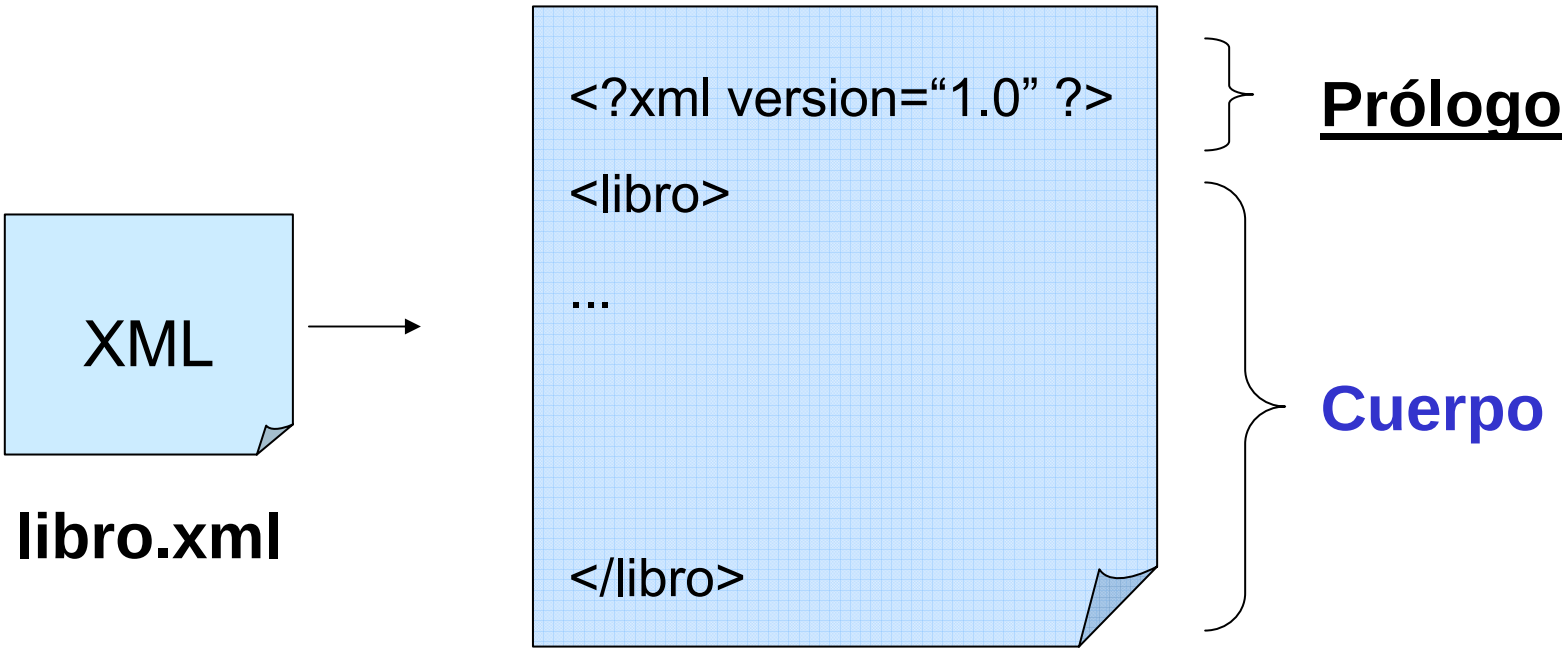
- Comienza con **<![CDATA[** y termina con **]]>**
- Puede contener cualquier caracteres incluso entidades predefinidas sin codificar, excepto la cadena de cierre "**]]>**"
- Uso insertar HTML o Javascript

```
<temperatura>&lt; 0</temperatura>
```

```
<temperatura><![CDATA[<0]]></temperatura>
```

3.1. Reglas sintácticas XML

- Estructura



3.1. Reglas sintácticas XML

Cómo debo escribir el **Prólogo**?

- El prólogo es opcional
- La primera línea debe comenzar con **<?xml** y terminar con **?>**.
- La primera línea especifica la **versión** y la **codificación** de caracteres

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

- La segunda línea especifica documentos asociados (DTD, Schema, XSL)

3.1. Reglas sintácticas XML

Ejemplos de prólogos:

<?xml version=“1.0” encoding=“ISO-8859-7”?>

<?xml version=“1.0” encoding=“UTF-16” standalone=“yes”?>

<?xml version=“1.0” encoding=“Big-5” standalone=“yes”?>
<!DOCTYPE clima SYSTEM “clima.dtd">

<?xml version=“1.0”>
<?xml:stylesheet type=”text/xsl” href=“prueba.xls”?>

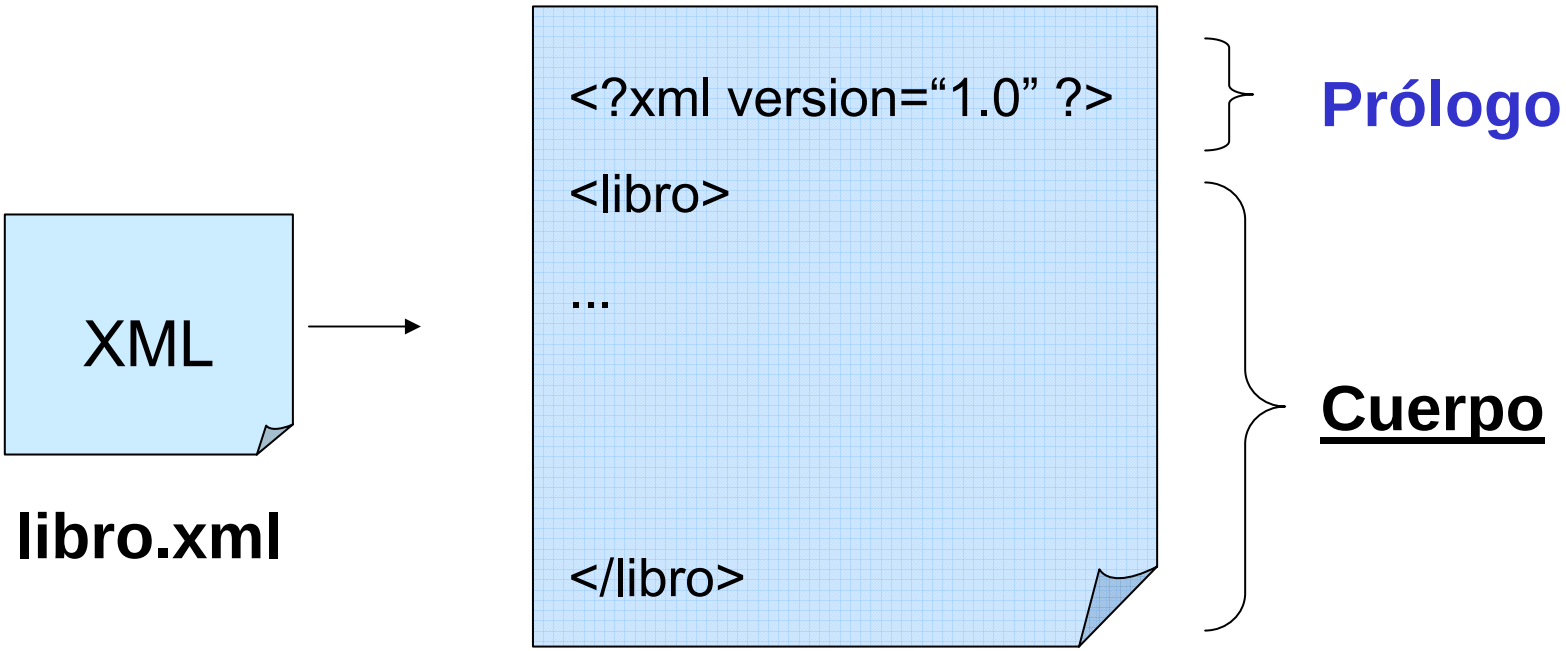
3.1. Reglas sintácticas XML

Codificación:

- Codificación de caracteres usada en el archivo XML
- Identificación del lenguaje basado en ISO o Unicode
- Valor implícito por defecto Unicode comprimido: encoding="UTF-8"

3.1. Reglas sintácticas XML

- Estructura



3.1. Reglas sintácticas XML

El cuerpo de un archivo XML tiene una estructura **jerárquica**:

- Estructura de árbol correctamente anidados
- No se pueden superponer elementos
- Debe haber un solo elemento raíz por documento XML
- Todas las etiquetas deben cerrarse

3.1. Reglas sintácticas XML

- Incorrectamente anidado

```
<libro>
<autor>Deepak Chopra
<titulo>El sendero del Mago</autor></libro></titulo>
```

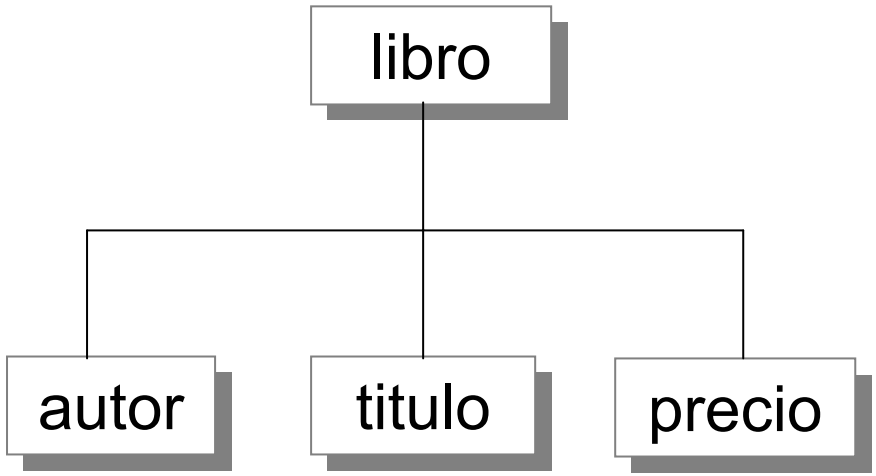
- Correctamente anidado

```
<libro>
<autor>Deepak Chopra</autor>
<titulo>El sendero del Mago</titulo>
</libro>
```

3.1. Reglas sintácticas XML

Estructura Jerárquica de Árbol

```
<libro>
<autor>Deepak Chopra</autor>
<titulo>El sendero del Mago</titulo>
<precio moneda="bolivares">30.000</precio>
</libro>
```



Modelos de datos XML

Está además asociado a la recomendación del W3C **DOM** (Document Object Model), aprobado también en 1998. Éste no es más que un modelo de objetos (en forma de API) que permite acceder a las diferentes partes que pueden componer un documento XML o HTML.

3.1. Reglas sintácticas XML

- Una pregunta antes de continuar:

  ¿Cómo decidir usar un dato como atributo o elemento?

  La respuesta: No hay claves, pero hay recomendaciones:

  - Se recomienda usar atributo cuando se vaya a categorizar información o cuando el valor sea único.

3.2. XML bien formado

- Un documento XML se dice “bien formado” (well-formed) cuando cumple las reglas descritas en la especificación oficial XML v1.0

http://www.programacion.com/html/xml/htmdsssl/xmlespes/Rec-xml.htm (español)

3.2. XML bien formado

- Cómo sé si un documento XML esta bien formado?

Uso un analizador sintáctico (parser)

Por ejemplo:

- Navegador de Internet
- Aplicación para editar XML (por ejemplo XML Spy)
- Analizadores en línea (w3c)

3.2. XML bien formado

- Una GLC para validar que libro1.xml esta bien formado:

S → <?xml version="1.0" ?>A

A → <libro>B</libro>

B → <autor>BC</autor>B |
<titulo>BC</titulo>B
<precio>BC</precio>B | C | ε

C → CC | ε | a | b | c | .... Z | . | $ |) | ... 0 | 1 | 2 |
...

3.3. Espacio de nombres

Responder a las preguntas:

- Puedo repetir el nombre de un elemento en cualquier documento XML?
- Quién decide el nombre del elemento?
- Cómo uso nombres estándares en los elementos para mejorar el intercambio de información?

3.3. Espacio de nombres

Colisión entre elementos:

```
<libro>
<autor>Deepak Chopra</autor>
<titulo>El sendero del Mago</titulo>
<precio moneda="bolivares">30.000</precio>
</libro>
```

libro.xml

```
<cliente>
<nombre>José Pérez</cliente>
<titulo>Dr.</titulo>
<email>perez@yahoo.com</email>
<tarjeta>45440029292</tarjeta>
</cliente>
```

cliente.xml

3.3. Espacio de nombres

Colisión entre elementos:

```
<orden>
<autor>Deepak Chopra</autor>
<titulo>El sendero del Mago</titulo>
<precio moneda="bolivares">30.000</precio>
<nombre>José Pérez</cliente>
<titulo>Dr.</titulo>
<email>perez@yahoo.com</email>
<tarjeta>45440029292</tarjeta>
</orden>
```

compra.xml

3.3. Espacio de nombres

XML namespaces

- Identifica la semántica de los elementos y atributos especialmente en el caso en donde el documento tiene elementos con el mismo nombre pero diferente significado.
- Ampliamente usado en aplicaciones para asegurar la consistencia del significado de los nombres.

3.3. Espacio de nombres

```
<orden
xmlns:bk=http://www.net-standard.com/namespaces/books
xmlns:cust="http://www.net-standard.com/namespaces/customer"
>
< bk:autor>Deepak Chopra</autor>
< bk:titulo>El sendero del Mago</titulo>
< bk:precio moneda="bolivares">30.000</precio>
<cust:nombre>José Pérez</cliente>
<cust:titulo>Dr.</titulo>
<cust:email>perez@yahoo.com</email>
<cust:tarjeta>45440029292</tarjeta>
</orden>
```

3.4. Espacio de nombres

- Se coloca en el momento de apertura del elemento que usa el namespace
- Los atributos no pertenecen al namespace del elemento. Hay que colocar el prefijo a los atributos, caso contrario se toma el de defecto.
- http://www.w3.org/TR/REC-xml-names/

PRÁCTICA #2

Abrir el archivo **libro2.xml** con el Navegador y determinar si está “bien formado”. En caso contrario hacer los cambios necesarios en su definición según las reglas de construcción de XML version 1.0.

PRÁCTICA #2

<?xml version="**2.0**"?>

<libro>

<autor>Deepak Chopra

<titulo>El sendero del Mago**</autor>**</titulo>

<isbn>950-15-1727**</Isbn>**

<editorial>Harmany Book</editorial>

<sumario>En esta obra, Deepak Chopra, autor de varios libros que han ocupado los primeros puestos en las listas de ventas, nos muestra cómo debemos ... Por medio de historias como Gail **&** Jarret... </sumario>

<precio moneda=**"bolivares**>30.000</precio>

<otro/>

</libro>

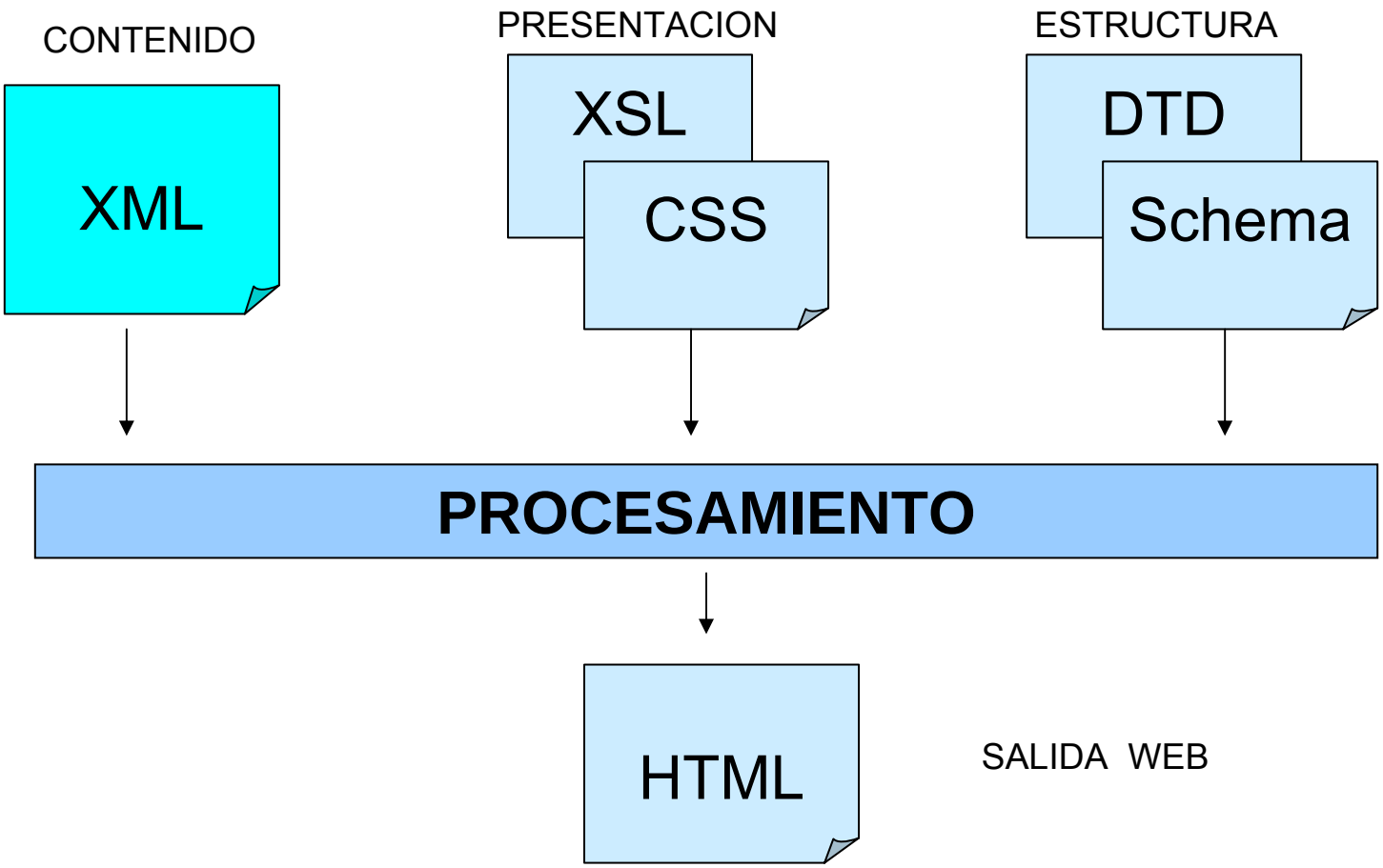
4 - Documentos XML: estructura y presentación

1. Presentación de documentos XML (CSS y XSLT)
2. Técnica de validación de la estructura de XML
    1. DTD
    2. XML Schemas
    3. Otras técnicas de validación
3. Lenguajes y Estándares XML
4. Editores XML

PRÁCTICA #3: Validando documentos XML con DTD

Presentación y estructura

Separación de procesamiento, presentación, estructura y contenido:



4.1. Presentación de XML

- CSS (Cascading Style Language): usado para escribir aspectos de presentación de HTML, XML, etc.
- XSLT (Extensible Styleshhet Languaje Transformations): usado para escribir transformaciones estructurales de los XML

4.1. Presentación de XML - CSS

Detalles de presentación: color, tamaño de fuente, posición, etc.

```
autor → etiqueta XML
{
background-color: #EEEEEE;
display: block;
color: #AAAAAA;
font-size: 20pt;
margin-left: 20pt;
}
```

Ver libro.css

4.1. Presentación de XML - CSS

En el encabezado del XML agregar el archivo css

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="libro.css"?>
```

Los navegadores web usan un archivo css por defecto para cada XML desplegado, si el archivo XML tiene uno usa el que se indica.

Ver libro2-correcto-css.xml

4.1. Presentación de XML - XSLT

Utiliza XPATH para tener acceso al árbol del modelo del documento XML de entrada y del XML de salida

XPATH: XML Path Language

- Seleccionar nodos
- Especificar condiciones para el procesamiento
- Generar texto para ser insertado en el árbol resultado (incluye manipulacion de string)

4.1. Presentación de XML - XSLT

El archivo libro2.xsl es un archivo de XSLT para procesar y mostrar el libro2-correcto.xml.

El XSL es una especificación estándar XML para transformar un archivo XML

4.1. Presentación de XML - XSLT

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" indent="yes" encoding="UTF-8"
      />

<xsl:template match="/">
      <table border="1">
      <xsl:apply-templates />
      </table>
</xsl:template>
```

4.1. Presentación de XML - XSLT

```
<xsl:template match="libro">
        <B><xsl:apply-templates /></B>
</xsl:template>

<xsl:template match="*">
        <tr><td>
        <xsl:value-of select="name()" />
        </td>
        <td>
        <xsl:value-of select="." />
        </td></tr>
</xsl:template>

</xsl:stylesheet>
```

4.1. Presentación de XML - XSLT

Un parser XSLT utiliza este lenguaje para transformar uno o varios archivos en XML.

PHP provee un parser XSLT (ver xslt.php):

```
<?php
   $xslDoc = new DOMDocument();
   $xslDoc->load("libro2.xsl");
   $xmlDoc = new DOMDocument();
   $xmlDoc->load("libro2-correcto.xml");
   $proc = new XSLTProcessor();
   $proc->importStylesheet($xslDoc);
   echo $proc->transformToXML($xmlDoc);
?>
```

4.1. Presentación de XML - XSLT

Un ejemplo: Mostrar mensajes aleatoriamente en una página web.

El XML de los mensajes es:

```
<mensajes>
    <mensaje id="1">
      <texto><![CDATA[Las palabras van al
    corazón cuando han salido del
    corazón]]></texto>
      <autor>Rabindranath Tagore</autor>
    </mensaje>
    …
</mensajes>
```

4.1. Presentación de XML - XSLT

Un ejemplo: Mostrar mensajes aleatoriamente en una página web.

El XSL que lo transforma:

```
<?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="html" indent="yes" encoding="ISO-8859-1" />

<xsl:param name="Id"/>
<xsl:template match="/">
     <xsl:apply-templates select="mensajes"/>
</xsl:template>

<xsl:template match="mensajes">
        <xsl:apply-templates select="mensaje[@id = $Id]"/>
</xsl:template>
```

4.1. Presentación de XML - XSLT

Un ejemplo: Mostrar mensajes aleatoriamente en una página web.

El XSL que lo transforma (continuación):

```
<xsl:template match="mensaje">
     <xsl:apply-templates select="texto"/>
</xsl:template>

<xsl:template match="texto">
     <b>
     <xsl:value-of select="." />
     </b><br/>
     <i><xsl:value-of select="../autor" /> </i>
</xsl:template>

<xsl:template match="*">
</xsl:template>
</xsl:stylesheet>
```

4.1. Presentación de XML - XSLT

Un ejemplo: Mostrar mensajes aleatoriamente en una página web.

En PHP se llama al parser XSLT:

```
$xslDoc = new DOMDocument();
$xslDoc->load("salida.xsl");
$xmlDoc = new DOMDocument();
$xmlDoc->load("m2009.xml");
$proc = new XSLTProcessor();
$proc->importStylesheet($xslDoc);
mt_srand (time());
$proc->setParameter(NULL, 'Id', mt_rand(1,15));
echo $proc->transformToXML($xmlDoc);
```

4.2. Técnica de validación XML

- Qué se valida?

La estructura: nombres y valores de los elementos, orden de los elementos, atributos y entidades

- Para qué validar?

Consistencia de datos, compartir datos válidos, uso de estándares.

4.2. Técnica de validación XML

Existen varias formas de definir los elementos que contiene un documento XML a través de reglas gramaticales de los elementos, atributos y entidades:

- **DTD (Document Type Definition)**

  Archivos con extensión .dtd

- **XML Schema**

  Archivos con extensión .xsd

4.2. Técnica de validación XML

DTD y XML Schema:

- Ambas formas nos permiten crear nuestro propio lenguaje de marcado.
- Ambas formas pueden residir en un archivo externo y ser compartidos por varios documentos XML.
- Un XML que se ajusta a cualquier técnica es un “XML válido”.
- Ambas técnicas son opcionales!

4.2. Validación con DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE libro SYSTEM "libro2.dtd">

<libro>
<autor>Deepak Chopra</autor>
<titulo>El sendero del Mago</titulo>
<isbn>950-15-1727</isbn>
<editorial>Harmany Book</editorial>
<sumario><![CDATA[En esta obra, Deepak Chopra, autor de
    varios libros que han ocupado los primeros puestos en
    las listas de ventas, nos muestra cómo debemos ...
    Por medio de historias como Gail & Jarret...
    ]]></sumario>
<precio moneda="bolivares">30.000</precio>
<otro/>
</libro>
```

4.2. Validación con DTD

GLC G = (V,T,P,S) del cuerpo de un libro en XML:

S → <libro>A</libro>

A → BCDEFGH | BCDEFG

B → <autor>K</autor> | BB

C → <titulo>K</titulo>

D → <isbn>K</isbn>

E → <editorial>K</editorial>

F → <sumario>K</sumario>

G → <precio moneda="K">K</precio>

H → <otro>K</otro>

K → KK|ε|a|b|…|z|0|1…|9|.|!...

4.2. DTD

Un ejemplo de DTD: libro2.dtd

```
<!ELEMENT libro (autor*, titulo, isbn, editorial, sumario,
      precio, otro?)>
<!ELEMENT autor   (#PCDATA)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
<!ELEMENT editorial (#PCDATA)>
<!ELEMENT sumario (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ATTLIST precio moneda CDATA #REQUIRED>
<!ELEMENT otro (#PCDATA)>
```

4.2. DTD

- Uso Externo

```
<?xml version="1.0"?>
<!DOCTYPE libro SYSTEM "libro2.dtd“>
```

- Uso Interno

```
<?xml version="1.0"?>
<!DOCTYPE libro[
        --definición del DTD—
    <!ELEMENT libro (autor, titulo, isbn, editorial, sumario, precio,
       otro?)>
...
    <!ELEMENT otro (#PCDATA)>
]>
```

Validador: http://www.w3schools.com/dom/dom_validate.asp

http://validator.w3.org/

4.2. DTD

- Validar en PHP un documento XML con DTD

```
$dom = new DOMDocument;
$dom->Load('libro2-correcto.xml');
if ($dom->validate()) {
    echo “El documento es valido!\n<BR>";
}
else
{
    echo “El documento es invalido!\n<BR>";
}
```

4.2. XML Schema

Un ejemplo de XML Schema: libro2.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified">
<xsd:element name="libro">
<xsd:complexType>
      <xsd:sequence>
      <xsd:element name="autor" type="xsd:string"/>
      <xsd:element name="titulo" type="xsd:string"/>  . . .
      <xsd:element name="precio">
        <xsd:complexType><xsd:simpleContent>
          <xsd:extension base="xsd:decimal">
          <xsd:attribute name="moneda" type="xsd:string" use="required"/>
          </xsd:extension>
        </xsd:simpleContent></xsd:complexType>
      </xsd:element>
      <xsd:element name="otro" type="xsd:string"/>
      </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

4.2. XML Schema

Un XML Schema es “similar” a un DTD, solo que:

- XML Schema usa sintaxis XML al contrario de los DTD
- Permite especificar los tipos y grupos de datos
- Son extensibles y tienen modularidad
- Usan namespace

4.2. XML Schema

- Uso Externo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<libro xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance"
    xsi:noNamespaceSchemaLocation="libro2.xsd">
<autor>Deepak Chopra</autor>
<titulo>El sendero del Mago</titulo>
<isbn>950-15-1727</isbn>
<editorial>Harmany Book</editorial>
<sumario><![CDATA[En esta obra, Deepak Chopra, autor
    de varios libros que han ... Por medio de
    historias como Gail & Jarret... ]]></sumario>
<precio moneda="bolivares">30.000</precio>
<otro/>
</libro>
```

4.2. XML Schema

- Validar en PHP un documento XML con XML Schema

```
$xdoc = new DomDocument;
$xmlfile = 'libro2-correcto-schema.xml';
$xmlschema = 'libro2.xsd';
$xdoc->Load($xmlfile);
if ($xdoc->schemaValidate($xmlschema)) {
   print "$xmlfile is valid.\n<BR>";
} else {
  print "$xmlfile is invalid.\n<BR>";
}
```

}

4.2. Otras Técnica de validación XML

RELAX NG, **Schematron** se está estandarizando como parte del DSDL (*Document Schema Definition Language)* de ISO

- RELAX NG

  http://www.relaxng.org/

- Schematron

  http://xml.ascc.net/schematron/1.5/

  http://www.schematron.com/

4.3. Lenguajes y Estándares XML

- XHTML (eXtended HTML)
- WML (Wireless Mark-up Language) para dispositivos inalambricos
- SVG (Scalable Vector Graphics) para producir imagenes
- RDF (Resource Definition Framework)
- VoiceXML
- SMIL Multimedia integrada

4.3. Lenguajes y Estándares XML

Lenguajes de Intercambio:

- ebXML - Comercio electrónico
- HL7 (Health Level Seven) – Hospitales y Salud
- NewsML – Noticias
- RSS (Really Simple Syndication) – Noticias
- SOAP (*Simple Object Access Protocol*)
- OAI-PMH (Open Archives Initiative – Protocol for Metadata Harvesting)
- DC (Dublin Core)

4.4. Editores XML

Editores:

* XML Pro de Vervet Logic (open source)
* XMLSpy de Altova
* <oXigen/> XML Editor
* Turbo XML de TIBCO (Plataforma de desarrollo integrado de XML)
* XML Notepad de Microsoft
* XMLwriter de Wattle Software

PRÁCTICA #3

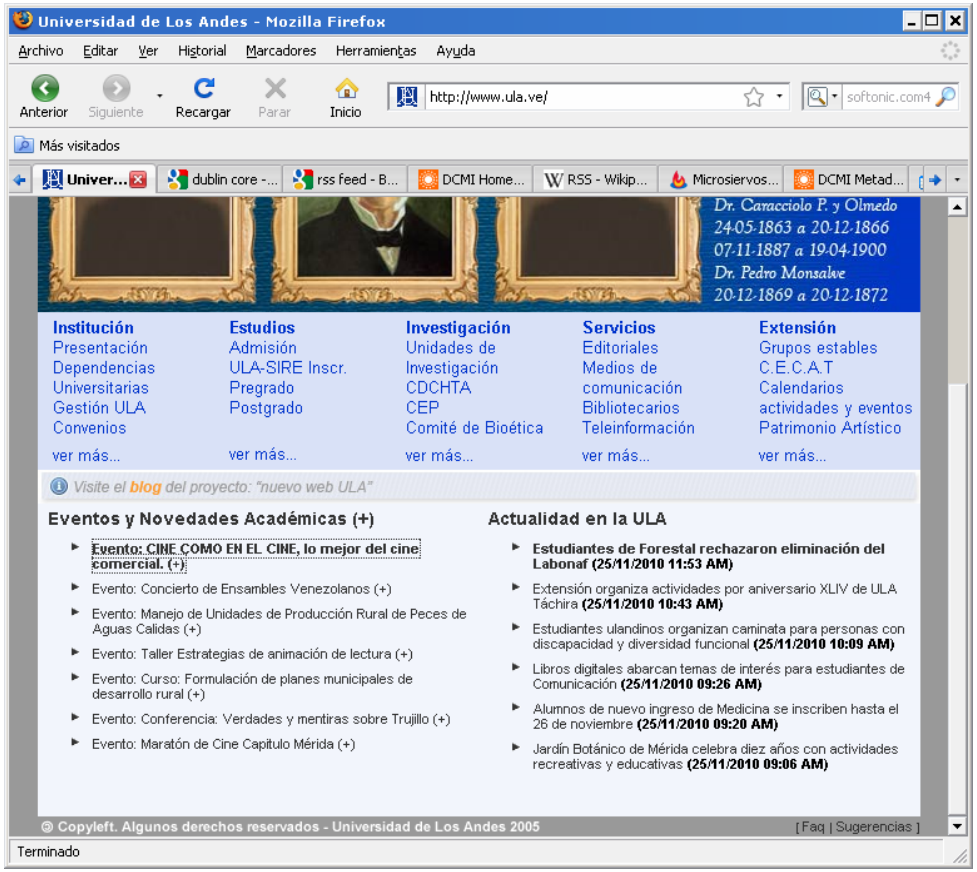
Usar PHP para validar el archivo libro2.xml con el DTD libro2.dtd y el XML Schema libro2.xsd agregando la modificación de:

- Se permite agregar un título traducido al libro
- Se permite sólo un autor
- El precio puede ser en BS o en BSF

5 - Uso de XML en Aplicaciones

1. Integrando XML en aplicaciones
2. Generando salidas XML desde una Base de Informacion (BI)
3. Transformando salidas XML de BI
4. Uso de XML como formato de intercambio (RSS, MARC-XML, OAI, DC, etc.)
5. Definición de servicios (Web Services)

RSS



RSS (Really Simple Syndication): XML para sindicar o compartir contenido en la web (RSS 2.0)

El formato OPML ("Outline Processor Markup Language", estándar para almacenar listas de subscripciones a canales RSS.

RSS

```
<?xml version="1.0" ?>
    <rss version="2.0">
     <channel>
      <title>Ajax and XUL</title>
      <link>http://www.xul.fr/en/</link>
      <description>XML graphical interface etc...</description>
      <image>
         <url>http://www.xul.fr/xul-icon.gif</url>
         <link>http://www.xul.fr/en/index.php</link>
      </image>
      <item>
        <title>News of today</title>
        <link>http://www.xul.fr/en-xml-rss.html</link>
        <description>All you need to know about RSS</description>
      </item>
      <item>
        <title>News of tomorrows</title>
        <link>http://www.xul.fr/en-xml-rdf.html</link>
        <description>And now, all about RDF</description>
      </item>
     </channel>
    </rss>
```

Dublin Core

Esquema de metadatos más utilizado a nivel mundial.
Ventajas:

- Su simplicidad
- La independencia sintáctica (que ha permitido que se integre en la estructuración de datos en XML/RDF).
- Alto nivel de normalización formal: ANSI/NISOZ39.85-2001, ISO 15836-2003.
- Crecimiento y evolución del estándar a través de una institución formal: la DCMI (consorcio).
- El conjunto de elementos DC se ha convertido en una infraestructura operacional del desarrollo de la Web Semántica

Dublin Core DC

Contenido:

- DC.Title **Título:** el nombre dado a un recurso, habitualmente por el autor.
- DC.Subject **Claves:** los tópicos del recurso.
- DC.Description **Descripción:** una descripción textual del recurso.
- DC.Source **Fuente** del cual proviene el recurso actual.
- DC.Language **Lengua:** del contenido intelectual del recurso.
- DC.Relation **Relación:** es un identificador de un segundo recurso y su relación con el recurso actual.
- DC.Coverage **Cobertura:** es la característica de cobertura espacial y/o temporal del contenido intelectual del recurso.

Dublin Core DC

Propiedad Intelectual:

- DC.Creator **Autor o Creador:** la persona o organización responsable de la creación del contenido intelectual del recurso.
- DC.Publisher **Editor:** la entidad responsable de hacer que el recurso se encuentre disponible en la red en su formato actual.
- DC.Contributor **Otros Colaboradores:** una persona u organización que haya tenido una contribución intelectual significativa
- DC.Rights **Derechos:** son una referencia sobre derechos de autor

Instanciación:

- DC.Date **Fecha:** una fecha en la cual el recurso se puso a disposición
- DC.Type **Tipo del Recurso:** la categoría del recurso. Por ejemplo, página personal, romance, poema, diccionario, etc.
- DC.Format **Formato:** es el formato de datos de un recurso, identificar el software y, posiblemente, el hardware que se necesitaría para mostrar el recurso.
- DC.Identifier **Identificador del Recurso:** secuencia de caracteres utilizados para identificar unívocamente un recurso. URL, URN, ISBN ("International Standard Book Number"), etc.

Dublin Core DC

Ejemplo de código en HTML con Metadatos basados en Dublín Core:

```
<HEAD>
<TITLE>Resource:
http://cvc.cervantes.es/aula/didactiteca/</TITLE>
<META NAME="DC.Creator" CONTENT="Centro Virtual
Cervantes, varios autores">
<META NAME="DC.Title" CONTENT="Didactiteca">
<META NAME="DC.Date.Created" CONTENT="2000--">
<META NAME="DC.Date.X-MetadataCreated"
CONTENT="2001-03-28">
<META NAME="DC.Publisher" CONTENT="Centro Virtual
Cervantes">
<META NAME="DC.Publisher.X-Email"
CONTENT="cvc@cervantes.es">
</HEAD>
```

Dublin Core DC

```
<?xml version="1.0"?>
<metadata
  xmlns="http://example.org/myapp/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://example.org/myapp/
     http://example.org/myapp/schema.xsd"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:creator>Deepak Chopra</dc:creator>
  <dc:title>El sendero del mago</dc:title>
  <dc:description><![CDATA[En esta obra, Deepak Chopra, autor de varios
     libros que han ocupado los primeros puestos en las listas de ventas,
     nos muestra cómo debemos ... Por medio de historias como Gail &
     Jarret... ]]></dc:description>
  <dc:publisher>Harmany Book</dc:publisher>
  <dc:identifier>950-15-1727</dc:identifier>
</metadata>
```