Introducción a XML

XML

Amparo López Gaona

México, D.F. Octubre de 2004

Aplicaciones de la WWW

La WWW ofrece servicios a una gran cantidad de gente, que en lo general no tiene conocimientos de computación, por lo que deben existir mecanismos sencillos de utilizar.

Actualmente encontramos gran variedad de páginas:

- Consulta de información.
- Servicios financieros.
- Compra y venta de artículos, etc.

WWW fue diseñada para presentar información estática.

Las páginas estáticas son una fracción (pequeña) de las páginas que se transfieren.

Las páginas dinámicas son el resultado de algún programa.

Modelo para las apliciones

Las aplicaciones que funcionan en la Web generalmente constan de tres entidades:

- La interfaz gráfica del usuario. En la mayoría de los casos es el navegador.
- El programa (o programas) de aplicación. Se ejecutan en el servidor Web y son los encargados de procesar los datos.
- El sistema manejador de bases de datos. Se encarga de almacenar y recuperar los datos que requiere el programa de aplicación.

Ventajas:

- Los navegadores funcionan en cualquier plataforma, lo que facilita el acceso a las aplicaciones.
- Las aplicaciones comparten el modelo de trabajo, tienen el mismo aspecto.
- La modularidad implícita facilita la modificación o el reemplazo de cualquier componente.

Una aplicación

Una entidad financiera emite las cotizaciones de los tipos de cambio cada hora. La información se puede consultar a través de una página disponible a todo público (`http://www.cnienlinea.com.mx/DivisasYMetales.asp`).

> Problema: Desarrollar una aplicación que emita una señal de alarma cuando el cambio peso/dólar tenga tendencia al alza (tres valores crecientes a lo largo del día).

Para desarrollar esta aplicación se cuenta con un servicio disponible en la Web. El servicio ofrece los tipos de cambio vigentes y se actualiza cada hora. La información está disponible en una página escrita con HTML.

¿Cómo se desarrollaría una aplicación que emita la señal de alarma en base a la consulta de la página que ofrece los tipos de cambio?

Página tipos de cambio

Tipos de cambio

Moneda	Compra	Venta
Dólar	8.87	9.12
Marco Alemán	3.8251	3.9149
Franco Francés	1.1405	1.1673
Franco Suizo	4.9203	5.0357
Libra Esterlina	12.5390	12.7580
Yen Japonés	0.0711	0.0728
Peseta Española	0.0450	0.0460

Código de la página tipos de cambio

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 TRANSITIONAL//EN">
<HTML>
<HEAD>
 <TITLE>Tipos de cambio</TITLE>
 <META NAME="GENERATOR" CONTENT="Emacs/20.4.1 (X11; I;
       Linux 2.2.12-20 i586) [Emacs]">
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
 <meta name="Author" content="Salvador Lopez Mendoza">
</HEAD>
<BODY>
<center><h1>Tipos de cambio</h1></center>
<center><table BORDER="1" WIDTH="50%">
<tr>
  <th><center>Moneda</center></th>
  <th><center>Compra</center></th>
  <th><center>Venta</center></th>
</tr>
<tr>
  <td>Dólar</td>
```

```
  <td><center>8.98</center></td>
  <td><center>9.18</center></td>
</tr>
<tr>
  <td>Marco Alemán</td>
  <td><center>3.8251</center></td>
  <td><center>3.9149</center></td>
</tr>
<tr>
  <td>Franco Francés</td>
  <td><center>1.1405</center></td>
  <td><center>1.1673</center></td>
</tr>
<tr>
  <td>Franco Suizo</td>
  <td><center>4.9203</center></td>
  <td><center>5.0357</center></td>
</tr>
...
</table></center>
</BODY>
</HTML>
```

Propuestas para la aplicación

1. Contratar a una persona que se encargue de consultar la página de los tipos de cambio cada hora y que emita la señal de alarma en caso necesario.

2. Desarrollar un sistema que consulte la página de tipos de cambio y vaya a los renglones correspondientes (20 y 21) y tome los siguientes cuatro caracteres a partir de la columna 15. En base a los valores obtenidos puede realizar el procesamiento para determinar si emite la alarma o no.

3. Desarrollar un sistema que consulte la página de tipos de cambio y vaya a la primer etiqueta `<table>`, después a la segunda etiqueta `<tr>` dentro de la tabla, a continuación a la segunda etiqueta `<td>`, que pase por la etiqueta `<center>` y tome el valor que se encuentra a continuación. Repetir el proceso para tomar el segundo valor (tercer etiqueta `<tr>`). En base a los valores obtenidos puede realizar el procesamiento para determinar si emite la alarma o no.

Son soluciones muy débiles.

¿Cuál es el problema?

HTML está diseñado para definir una *estructura de representación* de un documento, cuyos elementos estructurales son encabezados, títulos, párrafos, listas, etcétera; las etiquetas se diseñaron para cumplir ese propósito.

HTML no incluye etiquetas para representar *datos lógicos* (como el tipo de cambio de una moneda).

Para extraer la información se necesitan información sobre la *semántica* de la información.

Problemas adicionales:

- La mayoría de las páginas existentes no cumplen con las especificaciones de HTML.
- La mayoría de las páginas están rodeadas de gran cantidad de elementos superfluos (comerciales, animaciones, etc.).

Otra solución

¿Qué pasaría si la información estuviera escrita de la siguiente forma?

```
<TiposDeCambio>
   <Pais>México</Pais>
   <Fecha>6 de Julio de 2001</Fecha>
   <Hora>9:00 a.m.</Hora>
   <Divisa>
      <Nombre>Dólar</Nombre>
      <Compra>8.98</Compra>
      <Venta>9.18</Venta>
   </Divisa>
   <Divisa>
      <Nombre>Marco Alemán</Nombre>
      <Compra>3.8251</Compra>
      <Venta>3.9149</Venta>
   </Divisa>
</TiposDeCambio>
```

El documento tiene una *representación lógica* de los datos financieros, es independiente de la estructura de la presentación.

Nueva estrategia de solución: Desarrollar un programa que se encargue de buscar la etiqueta `<Divisa>` cuyo `<Nombre>` sea Dólar y tomar el valor que se encuentra entre las etiquetas `<Compra></Compra>`.

Es necesario que exista un acuerdo entre los que generan la información (los servidores de las cotizaciones) y la aplicaciones que la usan.

La representación lógica se debe convertir a la representación adecuada para el cliente, dependiendo de sus características:

- Para el programa que emite la señal de alarma el formato de los datos puede quedar como está.
- Para usuarios casuales (que visualizan la información a través de un navegador) se debe convertir a HTML.
- Para quien consulta el servicio a través de un PDA se podría convertir a algo más compacto (WML).

Extensión del servicio

Dado que es posible extraer la información se facilita la extensión de los servicios que ofrece el sistema.

Se podría ofrecer el sistema a otros clientes:

- Se conectarían a nuestras páginas en donde se identificarían como usuarios autorizados.
- Indicarían la moneda a monitorear.
- Determinarían la condición que dispara la señal de alarma.
- El método de notificación.

XML

El lenguaje XML (eXtensible Markup Language) es un lenguaje de marcado diseñado para definir contenido más que presentación.

Marcado significa un mecanismo para agregar meta-conocimiento e información de la estructura al documento.

Estándar para representación e intercambio de datos, principalmente en Internet.

EN tanto HTML usa las marcas para presentación (''itálicas''), XML las utiliza para definir la semántica (''esto es una dirección'').

La idea principal es crear un conjunto de marcas para un dominio (química, economía, etc.) y traducir todos los datos en documentos XML con las marcas apropiadas.

Características generales

- Usuarios definen sus propias marcas/etiquetas.
- Las marcas sobre elementos identifican su semántica en lugar de especificar el formato.
- La estructura de los documentos puede anidarse a cualquier nivel.
- Las relaciones entre los datos se dan vía anidamientos y referencias.
- Los documentos XML pueden contener la descripción de su gramática.
- El contenido de la información está separado de su traducción (formato).
- XML es una familia de tecnologías.
- XML no requiere de licencia y es independiente de plataformas.

Documentos XML

Un documento XML puede ser visto como una secuencia de caracteres que se sujetan a ciertas reglas y en ocasiones con cierto sentido.

Una aplicación XML es un vocabulario específico que contiene elementos y atributos.

Por ejemplo DocBook es una aplicación XML para producir manuscritos técnicos como un libro. Los elementos que define incluyen `book`, `chapter`, `para`, `sect1`, `sect2`, `programlisting`, etc.

Al escribir un documento DocBook, se utilizan esos elementos y en la forma establecida. Todos los documentos DocBook son documentos XML, pero al contrario no es cierto.

La sintaxis de la aplicación define cuáles son los documentos válidos y existen diferentes lenguajes para ello: DTD, esquemas.

Ejemplo

```
<TiposDeCambio>
   <Pais>México</Pais>
   <Fecha>6 de Julio de 2001</Fecha>
   <Hora>9:00 a.m.</Hora>
   <Divisa>
      <Nombre>Dólar</Nombre>
      <Compra>8.98</Compra>
      <Venta>9.18</Venta>
   </Divisa>
   <Divisa>
      <Nombre>Marco Alemán</Nombre>
      <Compra>3.8251</Compra>
      <Venta>3.9149</Venta>
   </Divisa>
</TiposDeCambio>
```

Ingredientes de XML

Un documento XML consta de tres tipos de cosas:

- Elementos. Todo elemento está delimitado por una marca de inicio y una de final.

  `<divisa>.... </divisa>`

- Atributos sobre los elementos. `<divisa moneda= "dolar" >.... </divisa>`

- Texto. Todo aquello que está entre las marcas y no es otro elemento.

  `<...>6 de Julio de 2001</...>`

Elementos

La unidad fundamental de XML es el elemento.

Cada documento tiene al menos un elemento.

Los elementos tienen 4 piezas:

- Un nombre.
- Atributos.
- El espacio de nombre en el alcance del elemento.
- Contenido.

Cada elemento está delimitado por un par de marcas:

- Una de inicio: <*nombre del elemento*>
- Una de fin </*nombre del elemento*>

Lo que está entre las marcas es el contenido del elemento. Este puede:

- Ser texto: `<país>México</país>`
- Estar formado de otros elementos.

```
<envio>
  <calleNum>Tlalcoligia 98</calleNum>
  <ciudad> México D.F.</ciudad>
  <cp>14440</cp>
  <país>México</país>
</envio>
```

- Ser mixto:

```
<envio>
  Srita. Andrea López
  <calleNum>Tlalcoligia 98</calleNum>
  <ciudad> México D.F.</ciudad>  <cp>14440</cp>
  <país>México</país>
</envio>
```

```
??

<mensaje>
    <para>ti@tuDirección.com</para>
    <de>mi@miDirección.com</de>
    <tema>Felicitación</tema>
   <texto>
       Este premio es la recompensa justa a tu esfuerzo. !Felicidades!
   </texto>
</mensaje>
```

Muy útil para documentos que contienen narrativa: libros, historias, etc.

No tan útil en aplicaciones orientadas a los datos.

- Estar vacío

```
<país></país>          !=  <país>  </país>

<país/>
```

En el contenido de un elemento no pueden ir los caracteres <, >, &, ;, ’.

Atributos

Los atributos son valores, con nombre, asociados con los elementos. La sintaxis es: *nombre = valor*

El nombre de un atributo es cualquier nombre válido y el valor es una cadena de texto, entre apostrofes o comillas.

```
<subtotal moneda='euro'>393.85</subtotal>
<subtotal moneda="euro">393.85</subtotal>
<subtotal moneda = "euro">393.85</subtotal>
```

```
<mensaje para="ti@tuDirección.com" de="mi@miDirección.com"
         tema="Felicitación">
    <texto>
       Este premio es la recompensa justa a tu esfuerzo. !Felicidades!
    </texto>
</mensaje>
```

El orden de los atributos es irrelevante.

```
<mensaje para="ti@tuDirección.com" de="mi@miDirección.com" tema="Felicitación">
<mensaje para="ti@tuDirección.com" tema="Felicitación" de="mi@miDirección.com">
<mensaje tema="Felicitación" de="mi@miDirección.com" para="ti@tuDirección.com">
```

Instrucciones de proceso

Estas instrucciones se utilizan por la aplicación que procesa un documento XML.

Especifican software particular para manejar el documento XML una vez que se ha realizado la revisión sintáctica.

Sintaxis: <? y ?>

Estas pueden aparecer antes, dentro o después del elemento raíz. No hay regla para ello. Generalmente aparecen en el prólogo.

En general no se asocian con una aplicación XML.

Se utilizan para información que no es relativa a XML.

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

<?xml-stylesheet type= "text/xml" href="limited.xsl"?>
```

Prólogo

Los documentos XML pueden empezar con un prólogo.

El prólogo contiene una declaración y la definición de las reglas sintácticas del mismo.

Atributos del prólogo:

- `version` Identifca la versión XML usada en tal documento. Este atributo es obligatorio y de momento su valor `1.0`.
- `encoding` Identifica el conjunto de caracteres usados. `ISO-8859-1` es para español. El valor por omisión es el Unicode comprimido `UTF-8`.
- `standalone` Especifica cuando un documento se refiera a una entidad externa o a una definición de tipos de dato externo. Si no hay tales referencias se debe escribir `yes`.

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

Todo lo que está después del prólogo constituye el contenido del documento.

Comentarios

Los comentarios permiten incluir en el documento información que no será procesada.

Se escriben entre los símbolos <!-- y -->

Se pueden colocar en cualquier sitio, excepto en las marcas, dentro de otros comentarios ni en las DTDs.

```
<!-- Verificar que este documento tenga código ISO-8859-1 -->
<mensaje para="ti@tuDirección.com" de="mi@miDirección.com"
         tema="Felicitación">
    <!-- Este es un comentario inútil -->
    <texto>
       Este premio es la recompensa justa a tu esfuerzo. !Felicidades!
    </texto>
</mensaje>
```

Entidades

Existen entidades predefinidas que son marcas para representar caracteres especiales. Se ponen entre los símbolos & y ;

Se tienen cinco entidades predefinidas:

1. & (&).
2. < (<).
3. > (>).
4. ' (').
5. " (").

Espacios de Nombre

No es fácil diseñar un buen esquema que sea utilizado por varias personas a lo largo del tiempo.

Los espacios de nombre permiten reutilizar esquemas existentes para diseñar documentos XML más complejos.

No son parte de XML 1.0, se crearon después.

La idea es asociar cada elemento a un *uniform resource identifier (URI)*.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/199/XSL/Transform">

  <xsl:template match="/">
...
```

Documentos bien formados

Un documento bien formado es el que cumple con la especificación XML, que dice que:

- Empieza especificando que se trata de un documento XML.
- Tiene un elemento raíz.
- Tiene nombres únicos de atributos.
- El valor de cada atributo se especifica entre comillas.
- No tiene marcas sin cerrar o mal cerradas.
- Sólo tiene caracteres UNICODE válidos.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<departamento>
  <empleado id="J.P">
    <nombre>Juan Pardo</nombre>
    <email>juan.pardo@yahoo.com</email>
  </empleado>

  <empleado id="B.S">
    <nombre>Benito Salas</nombre>
    <email>Benito.Salas@yahoo.com</email>
  </empleado>

  <empleado id="A.M">
    <nombre>Alicia Mata</nombre>
    <url href="http://www.yahoo.com/~amata/"/>
  </empleado>
</departamento>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<departamento>
  <empleado id="J.P">
    <nombre>Juan Pardo</nombre>
    <email>juan.pardo@yahoo.com</email1>
  </empleado>

  <empleado id="B.S">
    <nombre>Benito Salas</nombre>
    <email>Benito.Salas@yahoo.com</email>
  </empleado>

  <empleado id="A.M">
    <nombre>Alicia Mata</nombre>
    <url href="http://www.yahoo.com/~amata/"/>
  </empleado>
</departamento>
```

Documentos válidos

Una de las innovaciones de XML es que permite especificar de manera declarativa la estructura o esquema del documento.

Verificar que un documento cumpla con las especificaciones se denomina *validación*.

La forma de definir esquemas es por medio de:

- DTD (Document Type Descriptor). El único reconocido por la mayoría de los analizadores y especificado como parte estándar de XML.

- Esquemas.

DTDs

La DTD (Document Type Descriptor) es una gramática que especifica cuáles elementos pueden estar en un documento; para cada uno qué elementos puede y debe contener y en cuál orden. Además describe los atributos válidos para cada elemento.

Es una gramática libre de contexto extendida. Pues no tiene símbolos terminales.

Sea Σ un alfabeto finito de etiquetas. Una DTD consta de un conjunto de reglas de la forma $e \rightarrow r$ donde $e \in \Sigma$ y r es una expresión regular sobre Σ.

De aquí que pueda usarse como estándar para intercambio de documentos si las partes se ponen de acuerdo.

Por ejemplo: estructuras moleculares, facturas, etc.

Declaración de elementos en una DTD

Todo elemento permitido en el documento debe estar declarado en la DTD.

Sintaxis:

<!ELEMENT *nombre contenido*>

El nombre puede ser cualquier cadena, sólo que no debe empezar con & ni con `XML`. Contenido:

- Sin contenido.
- Sólo texto.
- Sólo elementos.
- Mixto.
- Sin restricciones.

Elementos vacíos

Como su nombre lo indica, no tienen contenido. Si tiene alguna información lo hará en sus atributos.

Sintaxis:

<!ELEMENT *nombre* EMPTY>

Ejemplo de declaración en la DTD:

```
<!ELEMENT url EMPTY>
```

Ejemplo de uso de elementos vacíos:

```
<url href="http://www.yahoo.com/~amata/"></url>
<url href="http://www.yahoo.com/~amata/"/>
```

Contenido sólo texto

Sintaxis:

<!ELEMENT *nombre* (#PCDATA)>

#PCDATA = (*Parser Character Data*)

Ejemplo de declaración en la DTD:

```
<!ELEMENT país (#PCDATA)>
```

Ejemplo de uso:

```
<país>México</país>
```

Contenido sólo elementos

Sintaxis:

<!ELEMENT *nombre modeloDeContenido*>

El *modeloDeContenido* se forma escribiendo entre paréntesis la lista de los subelementos separados por comas.

- Sin símbolo, especifica que se debe usar una vez.
- + especifica el al menos una vez este elemento.
- * especifica el uso de varias veces inclusive ninguna.
- ? especifica el uso o no del elemento.
- | da la opción de usar uno u otro elemento.
- () para tratar como unidad un grupo de elementos.

Ejemplo de declaración en la DTD:

```
<!ELEMENT NombreP (Nombre, ApellidoP, ApellidoM)>
<!ELEMENT NombreP (Nombre+, ApellidoP, ApellidoM?)>
<!ELEMENT resumen (educacion | experiencia)+, pasatiempos?, referencias*)>
```

Ejemplo de uso:

```
<nombreP>
   <Nombre> Amparo </Nombre>
   <ApellidoP> López </ApellidoP>
   <ApellidoM> Gaona </ApellidoM>
</nombreP>
```

Contenido mixto

Los elementos que contienen texto y elementos se denominan de contenido mixto.
Sintaxis:

<!ELEMENT *nombre* (#PCDATA | *listaDeElementos*)*>

Ejemplo de declaración en la DTD:

```
<!ELEMENT envio (#PCDATA calleNum ciudad cp país)>
```

Ejemplo de uso:

```
<envio>
  Srita. Andrea López
  <calleNum>Tlalcoligia 98</calleNum>
  <ciudad> México D.F.</ciudad>  <cp>14440</cp>
  <país>México</país>
</envio>
```

Contenido sin restricción

Sintaxis:

<!ELEMENT *nombre* ANY>

Son como elementos de contenido mixto pero sin ningún control.

De preferencia no usarlos.

Declaración de atributos en una DTD

La declaración de todo atributo empieza con `<!ATTLIST` seguida de la especificación de:

- El nombre del elemento al cual pertenece el atributo.
- El nombre del atributo.
- El tipo del atributo.
- El valor por omisión del atributo.

>

Tipos de atributo

- `CDATA`. Cadena de texto.
- `ID`. Nombre único, en el documento, entre los atributos.
- `IDREF`. Nombre de atributo usado como valor de un atributo `ID`.
- `IDREFS`. Lista de `IDREF` separados por espacios en blanco.
- `ENTITY`. Nombre de una entidad.
- `ENTITIES`. Lista de entidades separadas por espacios en blanco.
- `NOTATION`. Nombre de una notación.
- *enumeración*. Lista con los valores permitidos para ese atributo, cada uno separado por una barra vertical.

Valor para el atributo

- `#REQUIRED`. Se debe especificar valor para este atributo.
- `#IMPLIED`. El valor para este atributo es opcional.
- `#FIXED "`*valor*`"`. El atributo siempre tiene el valor especificado.
- "*valor*" es el valor por omisión de este atributo.

Ejemplo

```
<!ATTLIST jugador equipo CDATA #REQUIRED>
<!ATTLIST jugador posicion (1B | 2B | 3B | SS | pitcher | catcher | jardinero)
          "pitcher">
```

DTD para bibliografías

```
<!ELEMENT Bibliography (Book | Paper)*>
<!ELEMENT Book (Title, Authors, Publisher?, Remark?)>
<!ATTLIST Book ISBN CDATA #REQUIRED Edition CDATA #IMPLIED>
<!ELEMENT Paper (Title, Authors, Year?, Pages?, URL?)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Authors (Author+)>
<!ELEMENT Publisher (#PCDATA)>
<!ELEMENT Remark (#PCDATA)>
<!ELEMENT Author (#PCDATA | (FirstName, LastName))>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT LastName (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT Pages (First, Last)>
<!ELEMENT First (#PCDATA)>
<!ELEMENT Last (#PCDATA)>
```


Existen dos formas de especificar en el documento XML que se va a usar una DTD:

- Incluir la DTD en el documento:

  <!DOCTYPE elementoRaiz [...]>

- Hacer referencia a la DTD desde el documento:

  <!DOCTYPE elementoRaiz SYSTEM "URI de referencia">

  Ejemplos:

```
<?xml version = "1.0" standalone = "no"?>
<!doctype Bibliografia system "Bibliografia.dtd">

<!doctype Bibliografia system
                        "http://www.fciencias.unam.mx/dtds/Bibliografia.dtd">
```

Un documento bien formado que no incluye una DTD no puede ser válido.

Ejemplo de un documento XML

```
<?xml version="1.0" standalone="yes"?>
<Bibliography>
   <Book ISBN="ISBN-0-13-035300-0" Edition="2nd">
      <Title>A First Course in Database Systems</Title>
      <Authors>
         <Author>
            <FirstName>Jeffrey</FirstName>
            <LastName>Ullman</LastName>
         </Author>
         <Author>
            <FirstName>Jennifer</FirstName>
            <LastName>Widom</LastName>
         </Author>
      </Authors>
      <Publisher> Addison-Wesley </Publisher>
   </Book>
   <Book ISBN="ISBN-0-13-031995-3" >
      <Title>Database Systems: The Complete Book</Title>
      <Authors>
```

```
        <Author> Hector Garcia-Molina </Author>
        <Author>
           <FirstName>Jeffrey</FirstName>
           <LastName>Ullman</LastName>
        </Author>
        <Author>
           <FirstName>Jennifer</FirstName>
           <LastName>Widom</LastName>
        </Author>
      </Authors>
      <Remark>
      Amazon.com says: Buy this book bundled with "A First Course,"
      it's a great deal!
      </Remark>
   </Book>
</Bibliography>
```

Referencias

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE Bibliografia SYSTEM "Bibliografia.dtd">
<Bibliography>
   <Book ISBN="ISBN-0-13-035300-0" Edition="2nd">  ... </Book>
   <Book ISBN="ISBN-0-13-031995-3">
      <Title>Database Systems: The Complete Book</Title>
      <Remark>
         Amazon.com says: Buy this book bundled with
         <BookRef book="ISBN-0-13-035300-0"></BookRef>, It's a great deal!
       </Remark>
   </Book>
  <Paper Ident = "Abi99" Referencias="ISBN-0-13-035300-0">
    <Author>
      <Firstnombre> Serge </Firstnombre> <Lastnombre> Abiteboul </Lastnombre>
    </Author>
    <Title> Quering semistructured data </Title>
  </Paper>
  <Paper Ident = "Via01" Referencias="Abi99,ISBN-0-13-035300-0">  ...  </Paper>
</Bibliography>
```

La DTD debe cambiar.

```
<!DOCTYPE Bibliografia
<!ELEMENT Bibliography (Book | Paper)*>
<!ELEMENT Book (Title, Authors, Publisher?, Remark?)>
<!ATTLIST Book ISBN ID #REQUIRED Edition CDATA #IMPLIED>
<!ELEMENT Paper (Title, Authors, Year?, Pages?, URL?)>
<!ATTLIST Paper Ident ID #REQUIRED Referencias IDREFS #REQUIRED>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Authors (Author+)>
<!ELEMENT Remark (#PCDATA | BookRef)*>
<!ELEMENT BookRef EMPTY>
<!ATTLIST BookRef book IDREF #REQUIRED>
<!ELEMENT Author ( (FirstName, LastName) | #PCDATA)>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT LastName (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT Pages (First, Last)>
<!ELEMENT First (#PCDATA)>
<!ELEMENT Last (#PCDATA)>
 >
```

Entidades

Sintaxis:

<!ENTITY nombre definición>

Ejemplo de declaración:

```
<!ENTITY alg "&#169;2004 Amparo López Gaona ">
```

Ejemplo de uso:

```
.... &alg; ...
```

Atributos vs Elementos

Ventajas de los atributos:

- Pueden limitar su valor a una lista predefinida.
- Pueden tener un valor predefinido.
- Son concisos.
- Son más fáciles de usar que los elementos.

Desventajas de los atributos:

- No son convenientes para cadenas grandes de texto.
- No pueden contener información anidada.
- No pueden omitirse los espacios en blanco en su valor.

Ejercicio

Crear la DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE departamento SYSTEM "departamento.dtd">
<departamento>
  <empleado id="J.P">
    <nombre>Juan Pardo</nombre>
    <email>Juan.Pardo@yahoo.com</email>
  </empleado>

  <empleado id="B.S">
    <nombre>Benito Salas</nombre>
    <email>Benito.Salas@yahoo.com</email>
  </empleado>

  <empleado id="A.M">
    <nombre>Alicia Mata</nombre>
    <url href="http://www.yahoo.com/~amata/"/>
  </empleado>
</departamento>
```

Otro Ejercicio

```
<ciudades>
<ciudad>
<nombre>Acapulco</nombre>
<min>34</min>
<max>19</max>
<tiempo valor ="Soleado"/>
<mar>poco oleaje</mar>
<sol>
  <salida>6:08</salida>
  <puesta>7:05</puesta>
</sol>
</ciudad>

<ciudad>
<nombre>Mazatlán</nombre>
<min>32</min>
<max>20</max>
<tiempo valor ="Soleado"/>
<mar>Poco oleaje</mar>
<sol>
  <salida>5:23</salida>
  <puesta>6:41</puesta>
</sol>
</ciudad>

<ciudad>
<nombre>México</nombre>
<min>25</min>
<max>14</max>
<tiempo valor="Lluvia ligera"/>
<sol>
  <salida>7:01</salida>
  <puesta>8:06</puesta>
</sol>
</ciudad>
    ...
</ciudades>
```