

introducción a `<?xml?>`

alfredo.reino@frogdesign.de

introducción a <?xml?>

¿Qué es XML?

- XML es un subconjunto de SGML (*Standard Generalised Mark-up Language*), simplificado y adaptado a Internet.
- XML (*eXtensible Mark-up Language*) no es, como su nombre podría sugerir, un lenguaje de marcado.
- XML es un meta-lenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados.

introducción a <?xml?>

¿Qué no es XML?

- XML no es una “versión mejorada de HTML”.
- HTML es una aplicación de XML (más o menos)
- XML no es un lenguaje para hacer mejores páginas de web.
- XML no es difícil :-)

introducción a <?xml?>

¿Por qué XML?

- Es un estándar internacionalmente reconocido.
- No pertenece a ninguna compañía, y su utilización es libre.
- Permitirá la utilización efectiva de Internet en diferentes alfabetos, por gente con minusvalías físicas, y en diferente hardware (teléfonos celulares, PDAs, terminales Braille, etc.)

introducción a <?xml?>

Un ejemplo de HTML

```
<p><b>La insoportable levedad del ser</b>  
<br>Milan Kundera  
<br>Precio: <i>20 dólares</i>
```

introducción a <?xml?>

¿Qué problema tiene HTML?

- Define más la presentación que el contenido.
- No es fácilmente procesable por “máquinas”.
- Problemas de internacionalización.
- Su estructura es “caótica”.
- Su interpretación es ambigua según el software utilizado.
- Sólo tiene un uso: páginas de web.

introducción a <?xml?>

Una propuesta en XML

```
<libro>  
<titulo>La insoportable levedad del  
ser</titulo>  
<autor>Milan Kundera</autor>  
<precio moneda="USD">20</precio>  
</libro>
```

introducción a <?xml?>

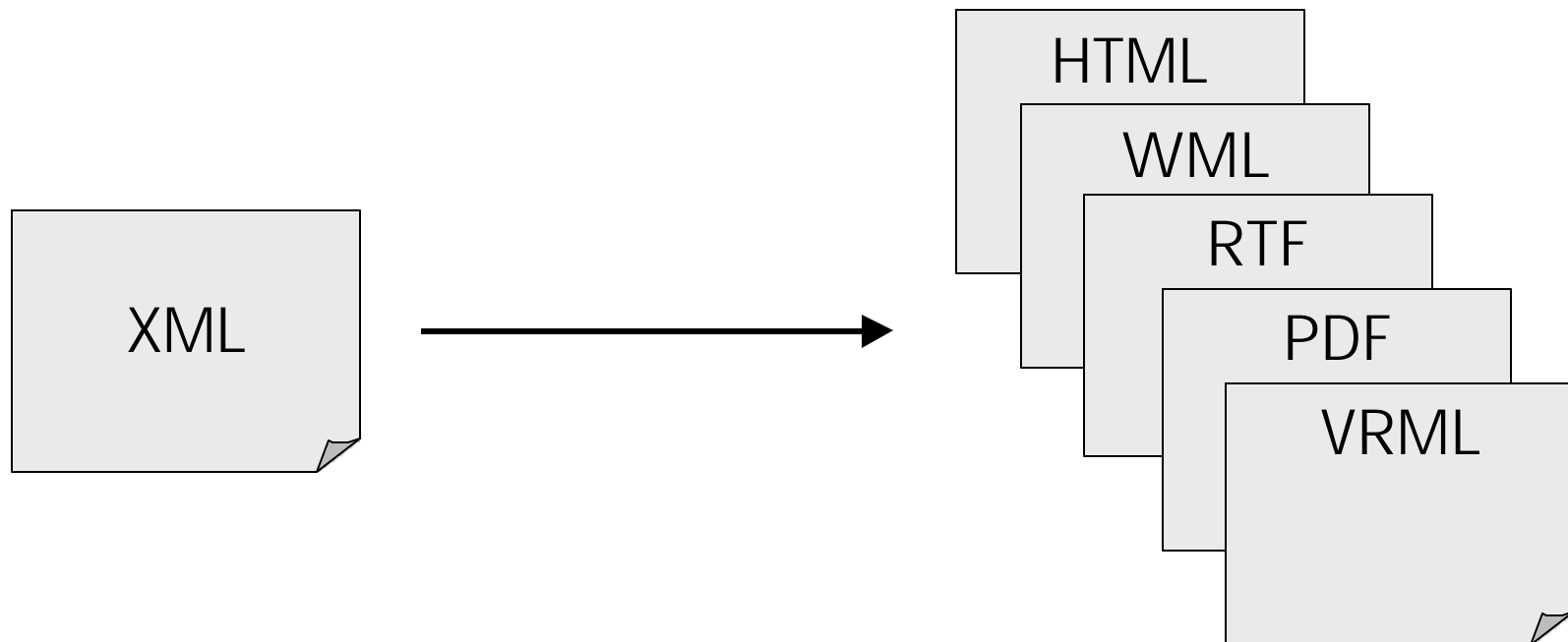
Ventajas de XML

- Fácilmente procesable tanto por humanos como por software.
- Separa radicalmente la información o el contenido de su presentación o formato.
- Diseñado para ser utilizado en cualquier lenguaje o alfabeto.

introducción a `<?xml?>`

Ventajas de XML (más...)

Un documento, muchas formas de presentación.



introducción a <?xml?>

Ventajas de XML (y más...)

- Formato ideal para transacciones B2B.
- Permite poderosas técnicas de extracción de información y *data-mining*.
- XML + validación = datos autodefinidos.
- Las estrictas reglas para la composición de un documento XML permiten su fácil análisis sintáctico.

introducción a <?xml?>

(Breve) historia de XML

- Desarrollado a partir de 1996, como subconjunto de SGML.
- Adoptado como estándar en Febrero 1998 por el World Wide Web Consortium (W3C)

introducción a <?xml?>

World Wide Web Consortium (W3C)

- Constituido en 1994 con el objetivo de desarrollar protocolos comunes para la evolución de Internet.
- Es un consorcio de industrias internacionales, y está participado por el MIT (EEUU), INRIA (Francia), y Keio University (Japón).
- Cuenta con el soporte oficial de DARPA (EEUU) y la Comisión Europea.

introducción a <?xml?>

La sopa de letras

XML XLL XPath XML-Sig
XSL XBase
DOM
DC RDF
XSLT
SAX XPointer
DTD
SGML Schema SML
namespace

introducción a <?xml?>

Un poco de terminología

XML	eXtended Mark-up Language
SGML	Standard Generalised Mark-up Language
XML "bien-formado"	
Validación	
DTD	Document Type Definition
Schema	
XSL	eXtended Stylesheet Language
Parser	Analizador sintáctico

introducción a <?xml?>

Un poco (más) de terminología

Elemento

Atributo

DOM

Document Object Model

RDF

Resource Description Framework

XLink, XPointer

SML

Simple Mark-up Language

SMIL

Synchronized Multimedia bla bla

XHTML

introducción a <?xml?>

Tecnologías XML (un resumen)

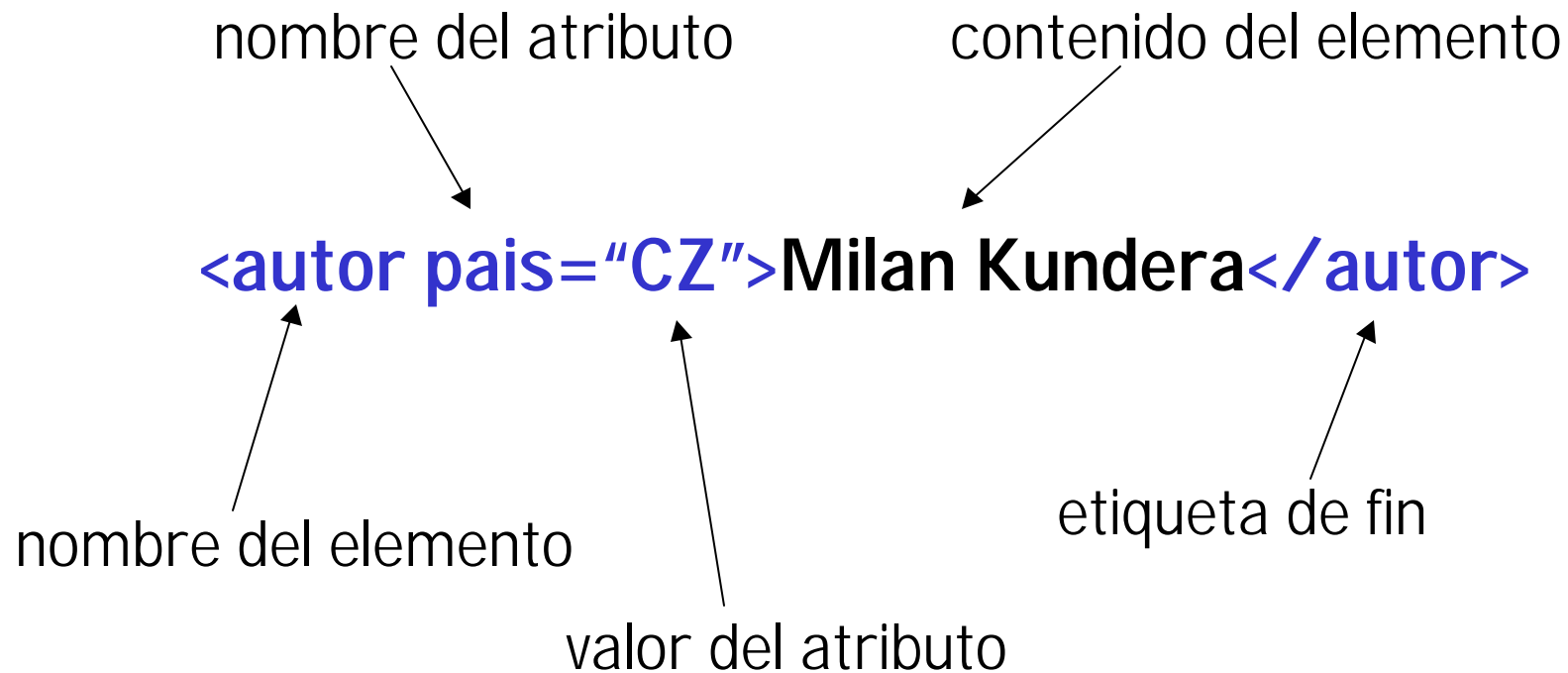
Especificación	XML v1.0
Definición de documentos	DTD or Schemas
Definición de estilos	XSL = XSLT + XPath
Enlazado de documentos	XLL = XLink + XPointer
Aplicaciones	RDF, SMIL, HTML, etc.

introducción a `<?xml?>`

Construyendo documentos XML

introducción a `<?xml?>`

El elemento y sus atributos



introducción a <?xml?>

XML “bien-formado” (*well-formed*)

- Se dice que un documento XML es “bien-formado” cuando cumple una serie de reglas descritas en la especificación oficial de XML v1.0

introducción a <?xml?>

Estructura jerárquica de elementos

- Los elementos deben seguir una estructura de “árbol”, es decir, estrictamente jerárquica.
- Los elementos deben estar correctamente anidados.
- Los elementos no se pueden superponer entre ellos.

introducción a <?xml?>

Ejemplo de XML erróneo

```
<nombre>Alfredo Reino<email>  
</nombre>alf@ibium.com</email>
```

introducción a <?xml?>

Estructura jerárquica de elementos

- Sólo puede haber un elemento raíz, en el que están contenidos todos los demás.

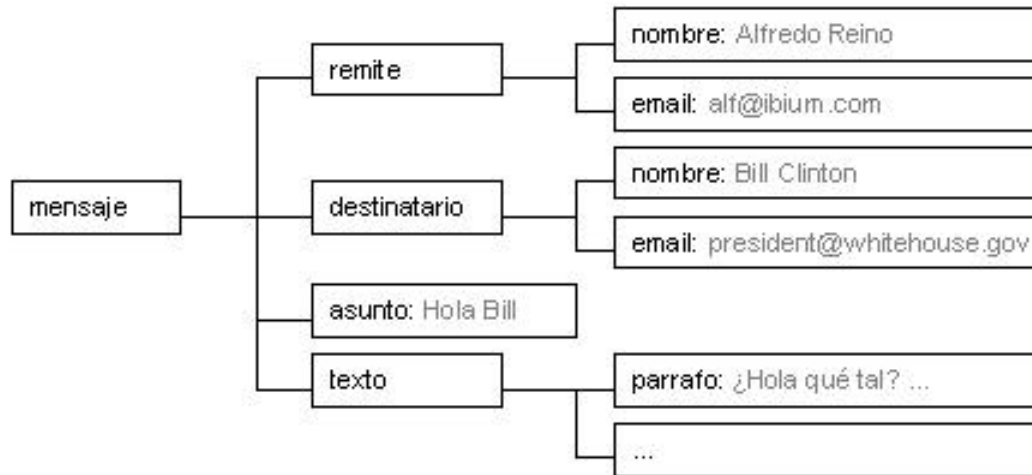
introducción a <?xml?>

Estructura jerárquica de elementos

```
<?xml version="1.0" encoding="UTF-7"?>
<!DOCTYPE mensaje SYSTEM "mensaje.dtd">
<mensaje>
  <remite>
    <nombre>Alfredo Reino</nombre>
    <email>alf@ibium.com</email>
  </remite>
  <destinatario>
    <nombre>Bill Clinton</nombre>
    <email>president@whitehouse.gov</email>
  </destinatario>
  <asunto>Hola Bill</asunto>
  <texto>
    <parrafo>¿Hola qué tal? ...</parrafo>
  </texto>
</mensaje>
```

introducción a <?xml?>

Estructura jerárquica de elementos



introducción a <?xml?>

Etiquetas

- Todas las etiquetas tienen que estar debidamente “cerradas”, es decir, con una etiqueta de cierre que se corresponda con la de apertura.
- Las etiquetas “vacías” (es decir, sin contenido) tienen una sintaxis especial.

introducción a <?xml?>

Ejemplos de XML incorrecto

```
<animal>Perro
    <raza tipo="Cocker Spaniel">
<animal>Vaca
    <raza tipo="Holstein">
```

introducción a <?xml?>

Una versión más correcta...

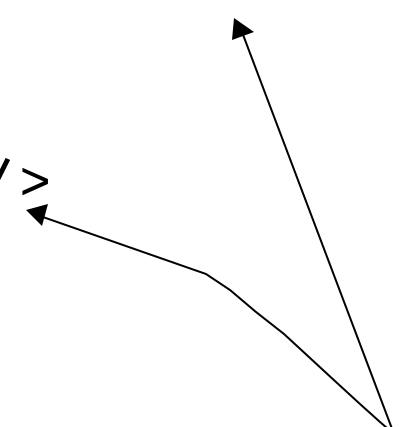
```
<animal>Perro
    <raza tipo="Cocker Spaniel">
</animal>
<animal>Vaca
    <raza tipo="Holstein">
</animal>
```

Las etiquetas tienen que estar SIEMPRE cerradas

introducción a <?xml?>

¡La versión más correcta!

```
<animal>Perro
    <raza tipo="Cocker Spaniel" />
</animal>
<animal>Vaca
    <raza tipo="Holstein" />
</animal>
```

Two arrows originate from a common point on the right side of the slide. One arrow points to the closing tag of the first XML snippet, `</animal>`. The other arrow points to the closing tag of the second XML snippet, `</animal>`.

Las etiquetas sin contenido se cierran de la forma mostrada

introducción a <?xml?>

Atributos

- Los valores de los atributos de los elementos siempre deben estar marcados con las comillas dobles (") o sencillas (')

`Esto es correcto`

`Esto es correcto`

`Esto NO es correcto`

introducción a <?xml?>

Nombrando cosas

- Un nombre de elemento, atributo, entidad, etc. empieza por una letra, y continua con letras, dígitos, guiones, rayas, punto o dos puntos.
- Las letras "XML" (o "xml" o "xMl", etc.) no pueden usarse como caracteres iniciales de un nombre de elemento, atributo, entidad, etc.

introducción a <?xml?>

Otras reglas

- XML es "*case-sensitive*", es decir, que no es lo mismo <autor> que <Autor>, por ejemplo.
- El uso del espacio blanco y los saltos de línea, en general funciona como en HTML (sólo se toma en cuenta cuando aparece en el valor de un atributo, o cuando se indica su significancia)

introducción a <?xml?>

Marcado y datos

- Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan “marcas” (mark-up). Estas son las partes del documento que el analizador sintáctico (parser) espera comprender.
- El resto del documento, que se encuentra entre las “marcas”, son los datos que resultan entendibles por las personas.

introducción a <?xml?>

Marcado y datos

- Marcas en un documento XML son aquellas que comienzan por el caracter "<" y terminan con ">".
- En el caso de las referencias de entidad, el caracter inicial es "&" y el final es ";"

introducción a <?xml?>

El prólogo

- El prólogo es opcional.
- La primera línea permite especificar la versión de XML (de momento sólo "1.0"), la codificación de carácter (US-ASCII, UTF-8, UTF-7, UCS-2, EUC-JP, Big5, ISO-8859-1, ISO-8859-7, etc.), y algunas otras cosas.
- La segunda línea define el tipo de documento, especificando que DTD valida y define los datos que contiene.

introducción a <?xml?>

Ejemplos de prólogos

```
<?xml version="1.0" encoding="UTF-7"?>  
<!DOCTYPE mensaje SYSTEM "mensaje.dtd">
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<!DOCTYPE HTML PUBLIC "-// /W3C/ /DTD HTML 3.2 Final/ /EN">
```

```
<?xml version="1.0" encoding="Big5"?>
```

introducción a <?xml?>

Elementos

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

Un elemento con contenido es, por ejemplo:

```
<nombre>Fulano Mengánez</nombre>
```

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

introducción a <?xml?>

Elementos

Siempre empieza con una <etiqueta> que puede contener atributos o no, y termina con una </etiqueta> que debe tener el mismo nombre. Al contrario que HTML, en XML siempre se debe "cerrar" un elemento.

Hay que tener en cuenta que el símbolo "<" siempre se interpreta como inicio de una etiqueta XML. Si no es el caso, el documento no estará bien-formado. Para usar ciertos símbolos se usan las entidades predefinidas, que se explican más adelante.

introducción a <?xml?>

Elementos vacíos

Un elemento vacío, es el que no tiene contenido (claro).

```
<identificador referencia="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. La sintaxis de HTML permite etiquetas vacías tipo <hr> o . En HTML reformulado para que sea un documento XML bien-formado, se debería usar <hr /> o

introducción a <?xml?>

Atributos

Como se ha mencionado antes, los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento.

```
<gato raza="Persa">Micifú</gato>
```

introducción a <?xml?>

Atributos

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

```
<verdura clase="zanahoria" longitud='15" y media'>
```

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


introducción a <?xml?>

Atributos

A veces, un elemento con contenido, puede modelarse como un elemento vacío con atributos. Un concepto se puede representar de muy diversas formas, pero una vez elegida una, es aconsejable fijarla en el DTD, y usar siempre la misma consistentemente dentro de un documento XML.

```
<gato><nombre>Micifú</nombre><raza>Persa</raza>  
</gato>
```

```
<gato raza="Persa">Micifú</gato>
```

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

introducción a <?xml?>

Entidades Predefinidas

En XML 1.0, se definen cinco entidades para representar caracteres especiales y que no se interpreten como marcado por el procesador XML. Es decir, que así podemos usar el carácter "<" sin que se interprete como el comienzo de una etiqueta XML, por ejemplo.

introducción a <?xml?>

Entidades Predefinidas

Entidad	Caracter
&	&
<	<
>	>
'	'
"	"

introducción a <?xml?>

Secciones CDATA

Existe otra construcción en XML que permite especificar datos, utilizando cualquier carácter, especial o no, sin que se interprete como marcado XML. La razón de esta construcción llamada CDATA (Character Data) es que a veces es necesario para los autores de documentos XML, poder leerlo facilmente sin tener que descifrar los códigos de entidades. Especialmente cuando son muchas.

introducción a <?xml?>

Secciones CDATA

```
<ejemplo>
&lt;HTML>
&lt;HEAD>&lt;TITLE>Rock & Roll&lt;/TITLE>
&lt;/HEAD>
</ejemplo>
```

introducción a <?xml?>

Secciones CDATA

```
<ejemplo>
```

```
<![CDATA[
```

```
<HTML>
```

```
<HEAD><TITLE>Rock & Roll</TITLE></HEAD>
```

```
]]>
```

```
</ejemplo>
```

introducción a <?xml?>

Secciones CDATA

Como hemos visto, dentro de una sección CDATA podemos poner cualquier cosa, que no será interpretada como algo que no es.

Existe empero una excepción, y es la cadena "]]>" con la que termina el bloque CDATA. Esta cadena no puede utilizarse dentro de una sección CDATA.

introducción a <?xml?>

Comentarios

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

Se pueden introducir comentarios en cualquier lugar de la instancia o del prólogo, pero nunca dentro de las declaraciones, etiquetas, u otros comentarios.

introducción a <?xml?>

Comentarios (ejemplo)

```
<?xml version="1.0"?>
<!-- Aquí va el tipo de documento -->
<!DOCTYPE EJEMPLO [
<!-- Esto es un comentario -->
<!ELEMENTO EJEMPLO (#PCDATA)>
<!-- ¡Eso es todo por ahora! -->
]>

<EJEMPLO>texto texto texto bla bla bla
<!-- Otro comentario -->
</EJEMPLO>
<!-- Ya acabamos -->
```

introducción a `<?xml?>`

Validación y definición de documentos

Document Type Definitions (DTD)

introducción a <?xml?>

DTD: Document Type Definition

- Crear una definición del tipo de documento (DTD) es como crear nuestro propio lenguaje de marcado, para una aplicación específica.
- La DTD define los tipos de elementos, atributos y entidades permitidas, y puede expresar algunas limitaciones para combinarlos.
- La DTD puede residir en un fichero externo, y quizá compartido por varios (puede que miles) de documentos. O bien, puede estar contenida en el propio documento XML, como parte de su declaración de tipo de documento.

introducción a <?xml?>

Document Type Definition

- Los documentos XML que se ajustan a su DTD, se denominan "válidos". El concepto de "validez" no tiene nada que ver con el de estar "bien-formado". Un documento "bien-formado" simplemente respeta la estructura y sintaxis definidas por la especificación de XML. Un documento "bien-formado" puede además ser "válido" si cumple las reglas de una DTD determinada. También existen documentos XML sin una DTD asociada, en ese caso no son "válidos", pero tampoco "inválidos"... simplemente "bien-formados"... o no.

introducción a <?xml?>

DTD (ejemplo)

```
<!DOCTYPE etiqueta[
<!ELEMENT etiqueta (nombre, calle, ciudad, pais, codigo)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT calle (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
<!ELEMENT pais (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
]>
<etiqueta>
<nombre>Fulano Mengánez</nombre>
<calle>c/ Mayor, 27</calle>
<ciudad>Valderredible</ciudad>
<pais>España</pais>
<codigo>39343</codigo>
</etiqueta>
```

introducción a <?xml?>

DTD

En el ejemplo anterior, todas las declaraciones DTD que definen "etiqueta" residen dentro del documento. Sin embargo, la DTD se puede definir parcial o completamente en otro lugar. Por ejemplo:

```
<?xml version="1.0"?>
<!DOCTYPE coche SYSTEM "http://sitio.com/coche.dtd">
<coche>
<modelo>...</modelo>
...
</coche>
```

introducción a <?xml?>

Declaraciones de tipo Elemento

Los elementos son la base de las marcas XML, y deben ajustarse a un tipo de documento declarado en una DTD para que el documento XML sea considerado válido.

Las declaraciones de tipo de elemento deben empezar con "<!ELEMENT" seguidas por el identificador genérico del elemento que se declara. A continuación tienen una especificación de contenido.

```
<!ELEMENT receta (titulo, ingredientes, procedimiento)>
```

introducción a <?xml?>

Declaraciones de tipo Elemento

Siguiendo la definición de elemento anterior, este ejemplo de documento XML sería válido:

```
<receta>  
<titulo>...</titulo>  
<ingredientes>...</ingredientes>  
<procedimiento>...</procedimiento>  
</receta>
```


introducción a <?xml?>

Declaraciones de tipo Elemento

Pero no este:

```
<receta>  
<parrafo>Esto es un párrafo</parrafo>  
<titulo>...</titulo>  
<ingredientes>...</ingredientes>  
<procedimiento>...</procedimiento>  
</receta>
```

introducción a <?xml?>

Declaraciones de tipo Elemento

La especificación de contenido puede ser de cuatro tipos:

EMPTY

Puede no tener contenido. Suele usarse para los atributos.

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

ANY

Puede tener cualquier contenido. No se suele utilizar, ya que es conveniente estructurar adecuadamente nuestros documentos XML.

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

introducción a <?xml?>

Declaraciones de tipo Elemento

Mixed

Puede tener caracteres de tipo datos o una mezcla de caracteres y sub-elementos especificados.

```
<!ELEMENT enfasis (#PCDATA)>
```

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

introducción a <?xml?>

Declaraciones de tipo Elemento

Element

Sólo puede contener sub-elementos especificados en la especificación de contenido.

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

Para declarar que un tipo de elemento tenga contenido de elementos se especifica un modelo de contenido en lugar de una especificación de contenido mixto o una de las claves ya descritas.

introducción a **<?xml?>**

Modelos de contenido

<!ELEMENT aviso (parrafo)>

Esto indica que <aviso> sólo puede contener un solo <parrafo>

<!ELEMENT aviso (titulo, parrafo)>

La coma, en este caso, denota una secuencia. Es decir, el elemento <aviso> debe contener un <titulo> seguido de un <parrafo>.

introducción a <?xml?>

Modelos de contenido

<!ELEMENT aviso (parrafo | grafico)>

La barra vertical "|" indica una opción. Es decir, <aviso> puede contener o bien un <parrafo> o bien un <grafico>. El número de opciones no está limitado a dos, y se pueden agrupar usando paréntesis.

<!ELEMENT aviso (titulo, (parrafo | grafico))>

En este último caso, el <aviso> debe contener un <titulo> seguido de un <parrafo> o de un <grafico>.

introducción a <?xml?>

Modelos de contenido

Además, cada partícula de contenido puede llevar un indicador de frecuencia, que siguen directamente a un identificador general, una secuencia o una opción, y no pueden ir precedidos por espacios en blanco.

Indicadores de frecuencia	
?	Opcional (0 o 1 vez)
*	Opcional y repetible (0 o más veces)
+	Necesario y repetible (1 o más veces)

introducción a <?xml?>

Modelos de contenido (ejemplo)

`<!ELEMENT aviso (titulo?, (parrafo+, grafico)*)>`

En este caso, <aviso> puede tener <titulo>, o no (pero sólo uno), y puede tener cero o más conjuntos <parrafo><grafico>, <parrafo><parrafo><grafico>, etc.

introducción a <?xml?>

Declaraciones de lista de Atributos

Los atributos permiten añadir información adicional a los elementos de un documento. La principal diferencia entre los elementos y los atributos, es que los atributos no pueden contener sub-atributos. Se usan para añadir información corta, sencilla y desestructurada.

```
<mensaje prioridad="urgente">
<de>Alfredo Reino</de>
<a>Hans van Parijs</a>
<texto idioma="holandés">
Hallo Hans, hoe gaat het?
...
</texto>
</mensaje>
```

alfredo.reino@frogdesign.de

introducción a <?xml?>

Declaraciones de lista de Atributos

Otra diferencia entre los atributos y los elementos, es que cada uno de los atributos sólo se puede especificar una vez, y en cualquier orden.

En el ejemplo anterior, para declarar la lista de atributos de los elementos <mensaje> y <texto> haríamos lo siguiente:

```
<!ELEMENT mensaje (de, a, texto)>  
<!ATTLIST mensaje prioridad (normal|urgente) normal>  
<!ELEMENT texto(#PCDATA)>  
<!ATTLIST texto idioma CDATA #REQUIRED>
```

introducción a <?xml?>

Declaraciones de lista de Atributos

Las declaraciones de los atributos empiezan con "<!ATTLIST", y a continuación del espacio en blanco viene el identificador del elemento al que se aplica el atributo. Después viene el nombre del atributo, su tipo y su valor por defecto.

En el ejemplo anterior, el atributo "prioridad" puede estar en el elemento <mensaje> y puede tener el valor "normal" o "urgente", siendo "normal" el valor por defecto si no especificamos el atributo.

introducción a <?xml?>

Declaraciones de lista de Atributos

El atributo "idioma", pertenece al elemento texto, y puede contener datos de carácter (**CDATA**). Es más, la palabra **#REQUIRED** significa que no tiene valor por defecto, ya que es obligatorio especificar este atributo.

A menudo interesa que se pueda omitir un atributo, sin que se adopte automáticamente un valor por defecto. Para esto se usa la condición "**#IMPLIED**". Por ejemplo, en una supuesta DTD que defina la etiqueta de HTML:

```
<!ATTLIST IMG URL CDATA #REQUIRED  
              ALT CDATA #IMPLIED>
```

introducción a <?xml?>

Tipos de Atributos

Atributos CDATA y NMTOKEN

Los atributos **CDATA** (*character data*) son los más sencillos, y pueden contener casi cualquier cosa. Los atributos **NMTOKEN** (*name token*) son parecidos, pero sólo aceptan los caracteres válidos para nombrar cosas (letras, números, puntos, guiones, subrayados y los dos puntos).

```
<!ATTLIST mensaje fecha CDATA #REQUIRED>  
<mensaje fecha="15 de Julio de 1999">  
<!ATTLIST mensaje fecha NMTOKEN #REQUIRED>  
<mensaje fecha="15-7-1999">
```

introducción a <?xml?>

Tipos de Atributos

Atributos Enumerados

Los atributos enumerados son aquellos que sólo pueden contener un valor de entre un número reducido de opciones.

```
<!ATTLIST mensaje prioridad (normal|urgente) normal>
```

introducción a <?xml?>

Tipos de Atributos

Atributos ID e IDREF

El tipo **ID** permite que un atributo determinado tenga un nombre único que podrá ser referenciado por un atributo de otro elemento que sea de tipo **IDREF**. Por ejemplo, para implementar un sencillo sistema de hipervínculos en un documento:

```
<!ELEMENT enlace EMPTY>  
<!ATTLIST enlace destino IDREF #REQUIRED>  
<!ELEMENT capitulo (parrafo)*>  
<!ATTLIST capitulo referencia ID #IMPLIED>
```

introducción a <?xml?>

Declaración de Entidades

- XML hace referencia a objetos (ficheros, páginas web, imágenes, cualquier cosa) que no deben ser analizados sintácticamente según las reglas de XML, mediante el uso de entidades. Se declaran en la DTD mediante el uso de "<!ENTITY"
- Una entidad puede no ser más que una abreviatura que se utiliza como una forma corta de algunos textos. Al usar una referencia a esta entidad, el analizador sintáctico reemplaza la referencia con su contenido. En otras ocasiones es una referencia a un objeto externo o local.

introducción a <?xml?>

Tipos de Entidades

Las entidades pueden ser:

Internas	o	Externas
Analizadas	o	No analizadas
Generales	o	Parámetro

introducción a <?xml?>

Tipos de Entidades

Entidades generales internas

Son básicamente abreviaturas definidas en la sección de la DTD del documento XML. Son siempre entidades analizadas, es decir, una vez reemplazada la referencia a la entidad por su contenido, pasa a ser parte del documento XML y como tal, es analizada por el procesador XML.

```
<!DOCTYPE texto[  
  <!ENTITY ovni "Objeto Volante No identificado">  
>  
<texto><titulo>Un día en la vida de un &ovni;  
</titulo></texto>
```

introducción a <?xml?>

Tipos de Entidades

Entidades generales externas analizadas

Las entidades externas obtienen su contenido en cualquier otro sitio del sistema, ya sea otro archivo del disco duro, una página web o un objeto de una base de datos. Se hace referencia al contenido de una entidad así mediante la palabra SYSTEM seguida de un URI (*Universal Resource Identifier*)

```
<!ENTITY intro SYSTEM "http://server.com/intro.xml">
```

introducción a `<?xml?>`

Tipos de Entidades

Entidades no analizadas

Evidentemente, si el contenido de la entidad es un archivo MPEG o una imagen GIF o un fichero ejecutable EXE, el procesador XML no debería intentar interpretarlo como si fuera texto XML. Este tipo de entidades siempre son generales y externas.

```
<!ENTITY logo SYSTEM "http://server.com/logo.gif">
```

introducción a <?xml?>

Tipos de Entidades

Entidades parámetro internas

Se denominan entidades parámetro a aquellas que sólo pueden usarse en la DTD, y no en el documento XML. Para hacer referencia a ellas, se usa el símbolo "%" en lugar de "&" tanto como para declararlas como para usarlas.

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

introducción a <?xml?>

Tipos de Entidades

Entidades parámetro externas

Igualmente, las entidades parámetro, pueden ser externas.

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

```

introducción a <?xml?>

Ejemplo de DTD

```
<?xml encoding="UTF-7"?>
```

```
<!ELEMENT lista (persona)+>
```

```
<!ELEMENT persona (nombre, email*, relacion?)>
```

```
<!ATTLIST persona id ID #REQUIRED>
```

```
<!ATTLIST persona sexo (hombre | mujer) #IMPLIED>
```

```
<!ELEMENT nombre (#PCDATA)>
```

```
<!ELEMENT email (#PCDATA)>
```

```
<!ELEMENT relacion EMPTY>
```

```
<!ATTLIST relacion amigo-de IDREFS #IMPLIED  
                    enemigo-de IDREFS #IMPLIED>
```

introducción a <?xml?>

XML basado en el DTD anterior

```
<?xml version="1.0"?>
<!DOCTYPE lista SYSTEM "LISTA.DTD">
<lista>
  <persona sexo="hombre" id="alvaro">
    <nombre>Álvaro Álvarez</nombre>
    <email>alvaroa@hotmail.com</email>
    <relacion amigo-de="beatriz">
  </persona>
  <persona sexo="mujer" id="beatriz">
    <nombre>Beatriz Bayo</nombre>
    <email>bea@terra.com</email>
  </persona>
</lista>
```


introducción a `<?xml?>`

Validación y definición de documentos

XML SCHEMAS

introducción a <?xml?>

XML Schemas

Un "schema XML" es algo similar a un DTD, es decir, que define qué elementos puede contener un documento XML, cómo están organizados, y que atributos y de qué tipo pueden tener sus elementos.

introducción a <?xml?>

XML Schemas

La ventaja de los schemas con respecto a los DTDs son:

- Usan sintaxis de XML, al contrario que los DTDs.
- Permiten especificar los tipos de datos.
- Son extensibles.

introducción a `<?xml?>`

XML Schemas

Veamos un ejemplo de un documento XML, y su schema correspondiente:

```
<documento xmlns="x-schema:personaSchema.xml">  
  <persona id="fulano">  
    <nombre>Fulano Menganez</nombre>  
  </persona>  
</documento>
```

introducción a <?xml?>

XML Schemas

```
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <AttributeType name='id' dt:type='string' required='yes' />
  <ElementType name='nombre' content='textOnly' />
  <ElementType name='persona' content='mixed'>
    <attribute type='id' />
    <element type='nombre' />
  </ElementType>
  <ElementType name='documento' content='eltOnly'>
    <element type='persona' />
  </ElementType>
</Schema>
```

introducción a `<?xml?>`

XML Schemas

ElementType

Define el tipo y contenido de un elemento, incluyendo los sub-elementos que pueda contener.

AttributeType

Asigna un tipo y condiciones a un atributo.

introducción a `<?xml?>`

XML Schemas

attribute

Declara que un atributo previamente definido por `AttributeType` puede aparecer como atributo de un elemento determinado.

element

Declara que un elemento previamente definido por `ElementType` puede aparecer como contenido de otro elemento.

introducción a `<?xml?>`

XML Schemas

Tal como hemos visto, es necesario empezar el schema definiendo los elementos más profundamente anidados dentro de la estructura jerárquica de elementos del documento XML. Es decir, tenemos que trabajar "de dentro hacia fuera".

Visto de otra manera, las declaraciones de tipo `ElementType` y `AttributeType` deben preceder a las declaraciones de contenido `element` y `attribute` correspondientes.

introducción a **<?xml?>**

XML Schemas

Para más información sobre XML Schemas:

<http://msdn.microsoft.com/xml/reference/schema/start.asp>

introducción a <?xml?>

Vocabularios y Languages de Marcado

- XHTML (eXtended HTML)
- WML (Wireless Mark-up Language)
- SVG (Scalable Vector Graphics)
- CDF (Channel Definition Format)
- RDF (Resource Definition Framework)
- XUL (eXtensible User Interface Language)
- GEML (Gene Expression Markup Language)
- VISA Invoice Specification
- VoxXML, VoiceXML, JSML
- ...

introducción a **<?xml?>**

XHTML v1.0

Una reformulación de HTML4 en XML

introducción a <?xml?>

XHTML

- XHTML, ya que es una aplicación XML, ha sido diseñado para ser ampliable.
- XHTML ha sido diseñado pensando en la portabilidad, y su visualización en diferente hardware (computadoras personales, PDAs, teléfonos celulares, etc.)
- XHTML es HTML4 escrito de forma que cumpla las normas sintácticas de XML.

introducción a <?xml?>

XHTML

- Los nombres de las etiquetas de elementos tienen que estar en minúscula.
- Los valores de los atributos deben estar rodeados de comillas (") o (')
- Todos los elementos tienen que estar cerrados, ya tengan contenido (<p> . . . </p>) o no (
)
- Los elementos deberán estar correctamente anidados.

introducción a <?xml?>

XHTML

- Los valores de atributos iguales sin variantes no pueden ser simplificados.
 - `<OPTION VALUE="valor" SELECTED>` (incorrecto)
 - `option value="valor" selected="selected">`
- Algunos elementos (html, body, head, ...) son obligatorios.

introducción a <?xml?>

XHTML

- Se debe incluir una Declaración de Tipo de Documento (DTD)

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

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

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

introducción a <?xml?>

XHTML

- El contenido de los elementos <script> y <style> tiene que ser una sección CDATA.

```
<script language="JavaScript">
<!--
<![CDATA[
document.write("<p>Texto de prueba</p>");
]]>
//-->
</script>
```


introducción a `<?xml?>`

Hojas de Estilo XSL

XSLT / XPath

introducción a <?xml?>

eXtended Stylesheet Language

- El XSL es un lenguaje que nos permite definir una presentación o formato para un documento XML. Un mismo documento XML puede tener varias hojas de estilo XSL que lo muestren en diferentes formatos (HTML, PDF, RTF, VRML, PostScript, sonido, etc.)
- La aplicación de una hoja de estilo XSL a un documento XML puede ocurrir tanto en el origen (por ejemplo, un servlet que convierta de XML a HTML para que sea mostrado a un navegador conectado a un servidor de web), o en el mismo navegador (como es el caso del MS IE5, y Netscape 6)

introducción a <?xml?>

eXtended Stylesheet Language

- Básicamente, XSL es un lenguaje que define una transformación entre un documento XML de entrada, y otro documento XML de salida.
- Una hoja de estilo XSL es una serie de reglas que determinan cómo va a ocurrir la transformación. Cada regla se compone de un patrón (*pattern*) y una acción o plantilla (*template*).

introducción a <?xml?>

XSL = XSLT + XPath

- XSLT (XSL Transformations)
 - Define las acciones o transformaciones a realizar.
 - W3C Recommendation 16 noviembre 1999
 - <http://www.w3.org/TR/xslt>
- XPath
 - Localización de elementos en el documento.
 - W3C Recommendation 16 noviembre 1999
 - <http://www.w3.org/TR/xpath>

introducción a <?xml?>

Un ejemplo sencillo de XSL

```
<?xml version="1.0" encoding="UTF-7"?>
<?xml-stylesheet href="links1.xsl" type="text/xsl"?>

<links>
  <item href="http://www.ibium.com/alf"
        title="Mi página personal"/>
  <item href="http://www.elpais.es"
        title="Diario El Pais"/>
  <item href="http://www.frogdesign.de"
        title="Mi empresa"/>
  <item href="http://www.terra.es"
        title="Mi antigua empresa"/>
</links>
```

introducción a <?xml?>

Un ejemplo sencillo de XSL

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

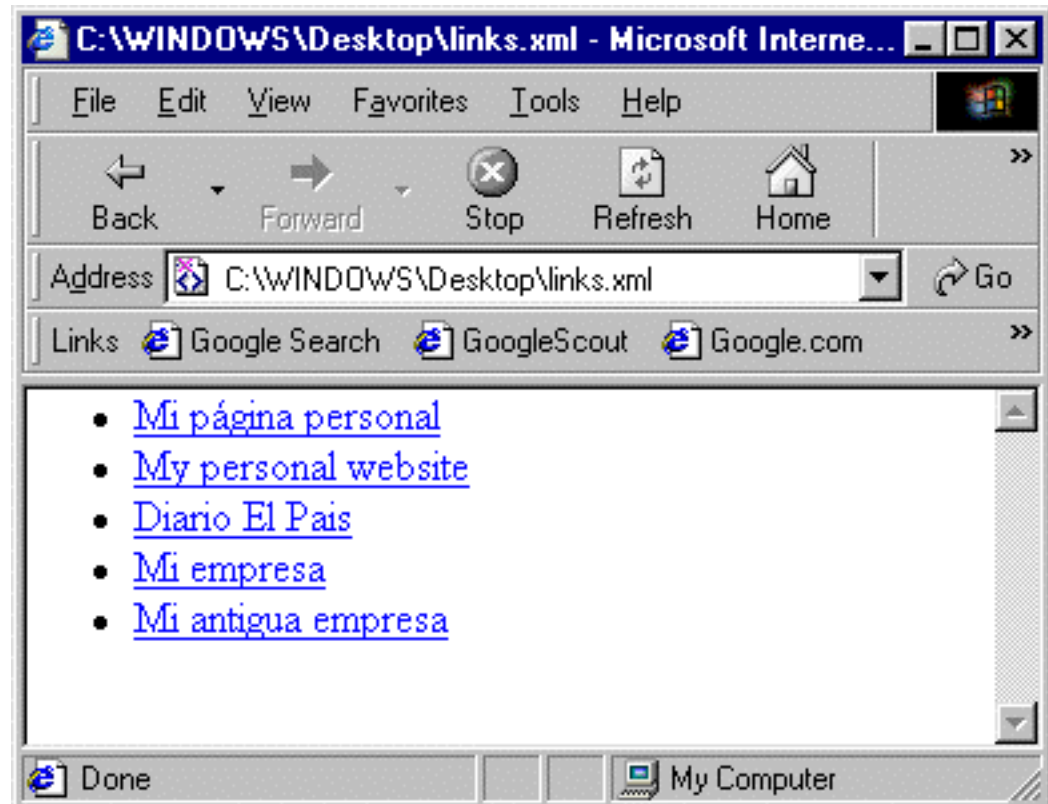
<xsl:template match="/">
<UL>
    <xsl:for-each select="links/item">
        <LI><A>
            <xsl:attribute name="href">
                <xsl:value-of select="./@href"/>
            </xsl:attribute>
            <xsl:value-of select="./@title"/>
        </A></LI>
    </xsl:for-each>
</UL>
</xsl:template>
</xsl:stylesheet>
```

alfredo.reino@frogdesign.de

introducción a <?xml?>

Un ejemplo sencillo de XSL

Al visualizar el XML con MSIE5, obtenemos este resultado



introducción a **<?xml?>**

Elementos de XSLT

<xsl:stylesheet> </xsl:stylesheet>

Es el elemento raíz de la hoja de estilo. Contiene elementos `xml:template` y `xml:script`

<xsl:template match="*pattern*"> </xsl:template>

Define un patrón para un determinado conjunto de nodos de un cierto tipo o en un contexto determinado.

introducción a <?xml?>

Elementos de XSLT

<xsl:apply-template select=" *pattern* " />

Indica al procesador que busque el template adecuado para el tipo y contexto de cada nodo seleccionado.

<xsl:value-of select=" *pattern* " />

Inserta el valor del nodo seleccionado como texto.

introducción a **<?xml?>**

Elementos de XSLT

<xsl:attribute name=" *nombre*"> *valor* </xsl:attribute>

Crea un atributo y lo añade al elemento de salida.

<xsl:element name=" *nombre*"> </xsl:element>

Crea un elemento con el nombre indicado. Esto es útil para evitar conflictos de nombres.

introducción a **<?xml?>**

Elementos de XSLT

<xsl:for-each order-by=" *criterio* " select=" *pattern* ">

Repite un template de forma iterativa.

<xsl:if test=" *expresion booleana* "> </xsl:if>

Realiza un test, y ejecuta un template condicionalmente.

introducción a <?xml?>

Elementos de XSLT

<xsl:choose>

<xsl:when test=" expresion1 "> ... </xsl:when>

<xsl:when test=" expresion2 "> ... </xsl:when>

...

<xsl:otherwise> ... </xsl:otherwise>

</xsl:choose>

Esta construcción permite realizar condiciones múltiples.

introducción a <?xml?>

Metodos en XMLT

XSLT provee una serie de métodos que pueden ser llamados desde los elementos <xsl:script> y <xsl:eval>

XSL Built-in Methods

absoluteChildNumber	Returns the number of the node relative to all siblings.
ancestorChildNumber	Returns the number of the nearest ancestor of a node with the requested node name.
childNumber	Returns the number of the node relative to siblings of the same node name.
depth	Returns the depth within the document tree at which the specified node appears.
formatDate	Formats the supplied date using the specified formatting options.
formatIndex	Formats the supplied integer using the specified numerical system.
formatNumber	Formats the supplied number using the specified format.
formatTime	Formats the supplied time using the specified formatting options.
uniqueID	Returns the unique identifier for the supplied node.

[© 1999 Microsoft Corporation. All rights reserved. Terms of use.](#)

introducción a <?xml?>

Especificación de patrones (XPath)

- / Especifica el "hijo" inmediato. Puede referirse al raíz.
- // Selecciona a "cualquier profundidad" en el árbol.
- . Selecciona el contexto actual.
- * Selecciona todos los elementos en el contexto actual.
- @ Selecciona un atributo.
- @* Selecciona todos los atributos en el contexto actual.

introducción a <?xml?>

Ejemplos de XPath

Selecciona todos los apellidos de clientes a partir del contexto actual.

```
cliente/apellido
```

Selecciona todos los elementos "empleado" que aparezcan a cualquier nivel de profundidad por debajo de elementos "tienda".

```
tienda//empleado
```

introducción a <?xml?>

Ejemplos de XPath

Selecciona todos los elementos "empleado" que aparezcan a un nivel de profundidad de distancia por debajo de elementos "tienda".

```
tienda/*/empleado
```

Selecciona todos los atributos "precio" de elementos "objeto" por debajo del contexto actual.

```
./objeto/@precio
```


introducción a <?xml?>

Ejemplos de XPath

Selecciona todos los elementos "empleado" que aparezcan a un nivel de profundidad de distancia por debajo de elementos "tienda" y cuyo "nombre" sea Alvaro.

```
tienda/*/empleado[nombre $eq$ "Alvaro"]
```

Selecciona todos los elementos "objeto" por debajo del contexto actual, cuyo atributo "precio" sea menor que 500, y que el contenido de su sub-elemento "tipo" sea 1.

```
./objeto[(@precio $le$ 500) $and$ (tipo $eq$ 1)]
```

introducción a <?xml?>

Aplicaciones de XSL

Con una serie de documentos en XML, podemos crear diferentes vistas en diferentes formatos (HTML estático, DHTML, WML, etc.) usando varias hojas de estilo XSL.

introducción a `<?xml?>`

EJEMPLO PRÁCTICO I

MENU DE ENLACES COLAPSABLE

introducción a <?xml?>

Menu de enlaces colapsable (XML)

```
<?xml version="1.0" encoding="UTF-7"?>
<?xml-stylesheet href="menu.xsl" type="text/xsl"?>
<enlaces>
  <item title="Mis enlaces personales">
    <item href="http://www.ibium.com/alf"
          title="Mi página personal"/>
    <item href="http://www.elpais.es"
          title="Mi diario favorito"/>
  </item>
  <item title="Empresas">
    <item href="http://www.frogdesign.de"
          title="Frogdesign"/>
    <item href="http://www.terra.es"
          title="Terra Networks"/>
  </item>
</enlaces>
```

alfredo.reino@frogdesign.de

introducción a <?xml?>

La hoja de estilo (1)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">

<HTML>
<HEAD>
<STYLE>
    BODY { font-family:verdana, arial, helvetica, sans-serif;
font-size:10pt; color: #000000; }
    UL.enlacesMuestra { }
    UL.enlacesOcultas { display:none; }
</STYLE>
</HEAD>

...
```

introducción a <?xml?>

La hoja de estilo (2)

...

```
<SCRIPT>
function Cambia() {
    var e = event.srcElement.nextSibling;
    if (e.className == "enlacesOcultas")
    {
        e.className = "enlacesMuestra";
    } else {
        e.className = "enlacesOcultas";
    }
}
</SCRIPT>
```

...

introducción a <?xml?>

La hoja de estilo (3)

...

```
<BODY>  
  <UL>  
    <xsl:apply-templates select="enlaces/item"/>  
  </UL>  
</BODY>  
</HTML>
```

```
</xsl:template>
```

...

introducción a <?xml?>

La hoja de estilo (4)

...

```
<xsl:template match="item">
  <LI><A TARGET="_BLANK">
    <xsl:attribute name="HREF">
      <xsl:value-of select="@href"/></xsl:attribute>
    <xsl:value-of select="@title"/></A>
  </LI>
</xsl:template>
```

...

introducción a <?xml?>

La hoja de estilo (y 5)

...

```
<xsl:template match="item[item]">
  <LI>
    <A ONCLICK="Cambia()" HREF="javascript:">
      <xsl:value-of select="@title"/>
    </A><UL CLASS="enlacesOcultas">
      <xsl:apply-templates select="item"/>
    </UL>
  </LI>
</xsl:template>

</xsl:stylesheet>
```

introducción a <?xml?>

¿Por qué no hacerlo crossbrowser?

```
<% @LANGUAGE="JScript" %>
<%
    var sXml = "menu.xml";
    var sXsl = "menu.xsl";

    var oXmlDoc = Server.CreateObject("Microsoft.XMLDOM");
    oXmlDoc.async = false;
    oXmlDoc.load(Server.MapPath(sXml));

    if (false != oXmlDoc.parseError)
    {
        Response.Write('XML parseError on line ' +
oXmlDoc.parseError.line);
        Response.End();
    }

    ...

```

introducción a <?xml?>

¿Por qué no hacerlo crossbrowser?

...

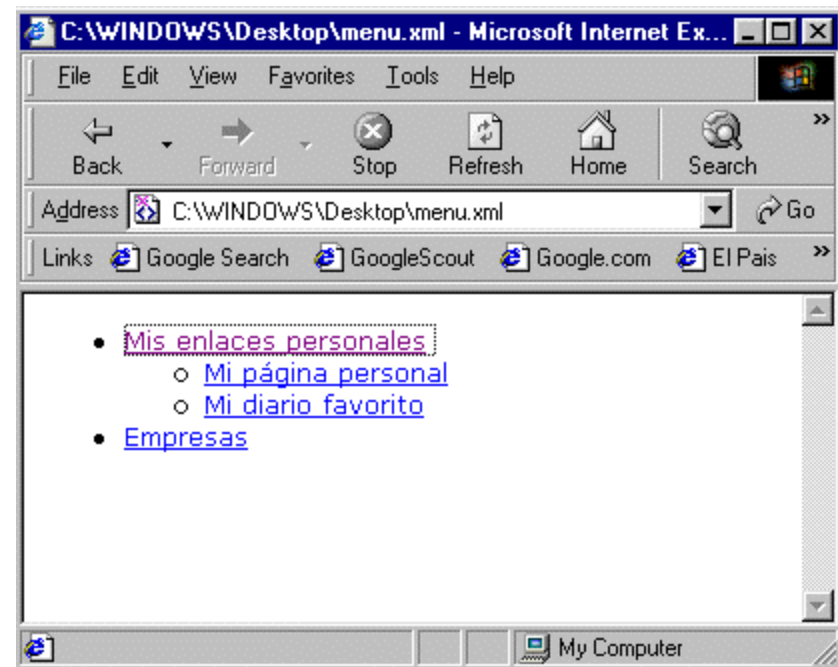
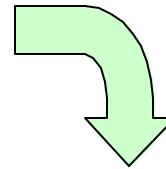
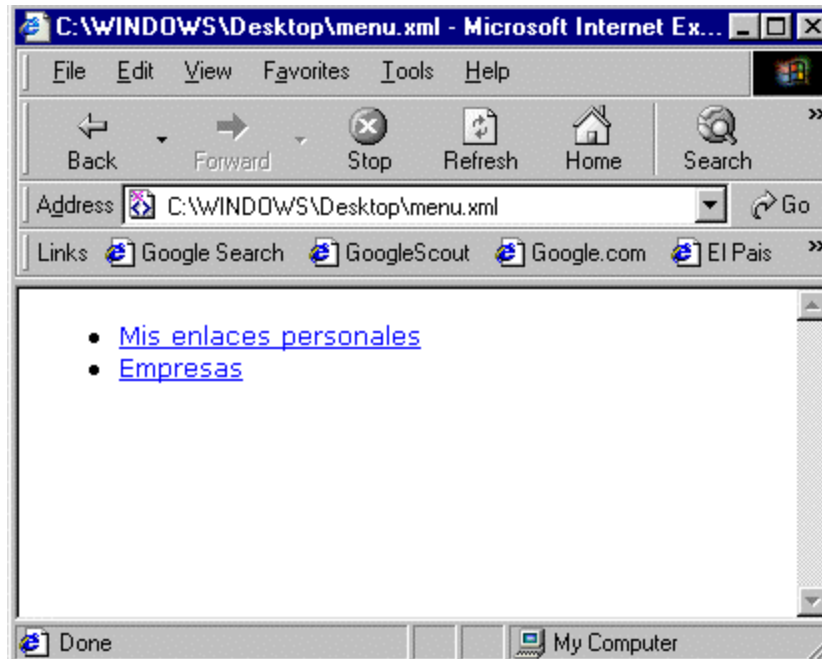
```
var oXslDoc = Server.CreateObject("Microsoft.XMLDOM");
oXslDoc.async = false;
oXslDoc.load(Server.MapPath(sXsl));

if (false != oXslDoc.parseError)
{
    Response.Write('XSL parseError on line ' +
oXslDoc.parseError.line);
    Response.End();
}

Response.Write(oXslDoc.transformNode(oXslDoc));
%>
```

introducción a `<?xml?>`

El resultado



introducción a `<?xml?>`

EJEMPLO PRÁCTICO II

FACTURAS EN XML

introducción a `<?xml?>`

Facturas en XML

Queremos desarrollar una hoja de estilo XSL que extraiga los datos de facturas y clientes de un fichero XML.

Además tiene que ser capaz de calcular totales y descuentos automáticos.

introducción a <?xml?>

El documento XML (ejemplo)

```
<?xml version="1.0" encoding="UTF-7"?>
<?xml-stylesheet type="text/xsl" href="factura.xsl"?>
<facturas xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <clientes>
    <cliente id="1001">Ramiro</cliente>
    <cliente id="1002">Rodrigo</cliente>
  </clientes>
  <factura id="5034">
    <cliente ref="1001"/>
    <articulos>
      <articulo>
        <numero-producto>213891723</numero-producto>
        <descripcion>Artículo útil</descripcion>
        <cantidad dt:dt="number">7</cantidad>
        <precio dt:dt="number">15.00</precio>
      </articulo>
    </articulos>
  </factura>
</facturas>
```

alfredo.reino@frogdesign.de

introducción a <?xml?>

La hoja de estilo (1)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <HTML>
      <STYLE>
        TD {font-size:9pt}
        BODY {font:10pt Arial}
      </STYLE>
      <BODY>
        <H3>FACTURAS</H3>
        <TABLE BORDER="1">
          <TR>
            <TD><B>Cantidad</B></TD><TD><B>Descripcion</B></TD>
            <TD><B>Precio</B></TD><TD><B>Descuento</B></TD>
            <TD><B>Total</B></TD>
          </TR>
          ...

```


introducción a <?xml?>

La hoja de estilo (2)

```
...
<xsl:for-each select="facturas/factura">
  <TR><TD COLSPAN="5" STYLE="border:none; background-color:#99ffcc">
    Factura #<xsl:value-of select="@id"/>
    (<xsl:value-of select="/facturas/clientes/cliente[@id=context()/cliente/@ref]"/>)
  </TD></TR>
  <xsl:for-each select="articulos/articulo">
    <TR>
      <TD><xsl:value-of select="cantidad"/></TD>
      <TD><xsl:value-of select="descripcion"/></TD>
      <TD>$<xsl:value-of select="precio"/></TD>
      <TD><xsl:if test="cantidad[. $ge$5]">
        <xsl:for-each select="precio">
          <xsl:eval>formatNumber(this.nodeTypeValue*.10, "$#,##0.00")</xsl:eval>
        </xsl:for-each>
      </xsl:if></TD>
      <TD STYLE="text-align:right">
        <xsl:eval>formatNumber(lineTotal(this), "$#,##0.00")</xsl:eval>
      </TD></TR>
    </xsl:for-each>
  </TD>
</TR>
...

```

introducción a <?xml?>

La hoja de estilo (3)

...

```
<TR>
  <TD COLSPAN="4"></TD>
  <TD STYLE="text-align:right; border:none;
              border-top:1px solid black">
    <xsl:eval>formatNumber(invoiceTotal(this), "$#,##0.00")</xsl:eval>
  </TD>
</TR>
<TR/>
</xsl:for-each>
</TABLE>
</BODY>
</HTML>
</xsl:template>
```

...

introducción a <?xml?>

La hoja de estilo (4)

...

```
<xsl:script><![CDATA[
```

```
function invoiceTotal(invoice)
{
    items = invoice.selectNodes("articulos/articulo");
    var sum = 0;
    for (var item = items.nextNode(); item; item = items.nextNode())
    {
        var price = item.selectSingleNode("precio").nodeTypedValue;
        var qty = item.selectSingleNode("cantidad").nodeTypedValue;
        if (qty >= 10)
            price = 0.9*price;
        sum += price * qty;
    }
    return sum;
}
```

...

alfredo.reino@frogdesign.de

introducción a <?xml?>

La hoja de estilo (y 5)

...

```
function lineTotal(item)
{
    var price = item.selectSingleNode("precio").nodeTypedValue;
    var qty = item.selectSingleNode("cantidad").nodeTypedValue;
    if (qty >= 10)
        price = 0.9*price;
    return qty*price;
}
```

```
]]></xsl:script>
```

```
</xsl:stylesheet>
```

introducción a <?xml?>

El resultado

FACTURAS

Cantidad	Descripcion	Precio	Descuento	Total
Factura #5034 (Ramiro)				
7	Artículo útil	\$15.00	\$1.50	\$105.00
				\$105.00
Factura #5035 (Rodrigo)				
2	Artículo muy útil	\$130.00		\$260.00
3	Manual de instrucciones	\$20.50		\$61.50
				\$321.50
Factura #5036 (Rodrigo)				
1	Artículo muy útil	\$130.00		\$130.00
1	Manual de XML	\$20.00		\$20.00
				\$150.00

introducción a **<?xml?>**

Analizadores sintácticos

(PARSERS)

introducción a `<?xml?>`

Analizadores sintácticos (*parsers*)

Parsers hay muchos, variados, y con funcionamientos muy diferentes.

Pueden incluir validación o no.

Pueden realizar transformaciones o no.

Pueden exponer la información de diferentes formas.

(DOM, SAX, etc.)

Existen para la mayoría de los lenguajes y plataformas de desarrollo (VB, ASP, C, VC++, Perl, Python, PHP, ...)

introducción a <?xml?>

El parser de Microsoft (MSXML.DLL)

- Realiza validación contra DTD.
- Realiza transformaciones.
- Soporte para XSL (versión anterior de XSL).
- Gratuito y redistribuible libremente.
- Esta basado en DOM Level 1.
- Puede ser utilizado en cualquier lenguaje o plataforma que soporte llamadas a objetos ActiveX (ASP, VB, VC++, ...)

introducción a <?xml?>

DOM Level 1

- DOM (Document Object Model) es un modelo que especifica la forma de acceder a los datos de un documento XML.
- DOM especifica un árbol jerárquico de nodos (elementos, comentarios, entidades, atributos, etc.).
- DOM especifica además un conjunto de “comandos” para movernos por el árbol.

<http://www.w3.org/TR/REC-DOM-Level-1>

introducción a <?xml?>

Usando MSXML.DLL

Para crear una instancia del parser en un ASP, hacemos lo siguiente:

```
Set objParser = Server.CreateObject("Microsoft.XMLDOM")
```

Para utilizarlo en Visual Basic, tenemos que añadir el objeto COM Microsoft XML, version 2.0, en las referencias del proyecto, y luego crear una instancia del objeto:

```
Dim objParser As MSXML.DOMDocument  
Set objParser = New MSXML.DOMDocument
```

introducción a <?xml?>

Usando MSXML.DLL

Para cargar un documento:

```
objParser.validateOnParse = False
If objParser.Load("c:\temp\xml\documento.xml") Then
    ' Ha funcionado
Else
    ' Ha ocurrido un error
End If

' Ahora destruimos el objeto parser
Set objParser = nothing
```

introducción a <?xml?>

Usando MSXML.DLL

Una vez cargado, analizado y validado el documento, podemos empezar a utilizar la información que contiene.

Para eso utilizamos la interface `IXMLDOMNode` del parser, que nos permite acceder a los diferentes nodos (elementos, atributos, texto, etc.) del documento.

Este interface nos provee de ciertos métodos y propiedades para acceder con comodidad a los contenidos del documento, tanto para lectura como para escritura.

introducción a <?xml?>

Métodos y Propiedades

nodeType

Nos da información sobre el tipo de un nodo. Este parser soporta 13 tipos diferentes de nodo.

Constantes de tipos de nodo
NODE_ATTRIBUTE
NODE_CDATA_SECTION
NODE_COMMENT
NODE_DOCUMENT
NODE_DOCUMENT_FRAGMENT
NODE_DOCUMENT_TYPE
NODE_ELEMENT
NODE_ENTITY
NODE_ENTITY_REFERENCE
NODE_INVALID
NODE_NOTATION
NODE_PROCESSING_INSTRUCTION
NODE_TEXT

introducción a `<?xml?>`

Métodos y Propiedades

nodeName

Si el tipo de nodo es adecuado, nodeName nos devuelve el nombre del nodo. Por ejemplo, un nodo tipo `NODE_ELEMENT` contendría en nodeName el nombre de un elemento de un documento XML.

introducción a <?xml?>

Métodos y Propiedades

nodeValue

Nos devuelve el valor que contiene ese nodo, si es aplicable.

childNodes

Contiene una colección de nodos "hijos" del nodo que estamos considerando. Esta colección puede ser iterada por una construcción for each de Visual Basic.

introducción a `<?xml?>`

Métodos y Propiedades

hasChildNodes

Propiedad booleana que nos dice si un nodo tiene "hijos".

firstChild / lastChild

Contienen referencias al primer y último "hijos" de un nodo.

introducción a `<?xml?>`

Métodos y Propiedades

parentNode

Nos devuelve una referencia al "padre" del nodo.

nextSibling / previousSibling

Nos devuelve una referencia al siguiente/anterior "hermano" del nodo, es decir, al siguiente/anterior nodo con el mismo "padre".

introducción a `<?xml?>`

Métodos y Propiedades

attributes

Nos devuelve una colección de los nodos tipo `NODE_ATTRIBUTE` del nodo.

introducción a <?xml?>

Ejemplo en Visual Basic

Vamos a ver un ejemplo de un sencillo programa que selecciona todos los nodos tipo texto (NODE_TEXT) de un documento XML de una forma recursiva.

El fichero test.xml que usa este programa es el siguiente:

```
<?xml version="1.0"?>
<documento>
  <titulo>Un documento cualquiera</titulo>
  <fecha><dia>1</dia><mes>12</mes><año>1999</año></fecha>
</documento>
```

introducción a <?xml?>

Ejemplo en Visual Basic

Primero cargamos el documento en el parser y comprobamos los posibles errores.

```
Public Sub CargaDoc()  
Dim oParser As MSXML.DOMDocument  
Set oParser = New MSXML.DOMDocument  
If oParser.Load("test.xml") Then  
    MuestraNodos oParser.childNodes  
Else  
    MsgBox "Ha ocurrido un error."  
End If  
End Sub
```

introducción a <?xml?>

Ejemplo en Visual Basic

A continuación analizamos de forma recursiva el documento.

```
Public Sub MuestraNodos _  
    (ByRef Nodos As MSXML.IXMLDOMNodeList)  
    Dim oNodo As MSXML.IXMLDOMNode  
    For Each oNodo In Nodos  
        If oNodo.nodeType = NODE_TEXT Then  
            Debug.Print oNodo.parentNode.nodeName & "=" & _  
                oNodo.nodeValue  
        End If  
        If oNodo.hasChildNodes Then  
            MuestraNodos oNodo.childNodes  
        End If  
    Next oNodo  
End Sub
```

introducción a <?xml?>

Ejemplo en Visual Basic

El resultado obtenido sería el siguiente:

```
titulo=Un documento cualquiera  
dia=1  
mes=12  
año=1999
```

introducción a <?xml?>

Convirtiendo XML en HTML

Una forma de transformar un documento en HTML usando una hoja de estilo XSL y el parser MSXML.DLL para ser utilizado en páginas ASP.

```
<%  
    Set xmlObj = CreateObject("Microsoft.XMLDOM")  
    xmlObj.Async = False  
    xmlObj.Load(Server.MapPath("documento.xml"))  
  
    Set xslObj = CreateObject("Microsoft.XMLDOM")  
    xslObj.Async = False  
    xslObj.Load(Server.MapPath("estilo.xsl"))  
  
    Response.Write(xmlObj.transformNode(xslObj))  
%>
```

introducción a `<?xml?>`

EJEMPLO PRÁCTICO III

LIBRO DE FIRMAS BASADO EN XML

introducción a <?xml?>

El libro de firmas XML

- El objetivo es implementar un libro de firmas para una página de web.
- Los datos del libro se almacenarán en un documento XML, y se mostrarán mediante una hoja de estilo XSL.
- La plataforma elegida es Microsoft ASP, y el lenguaje a usar es Visual Basic Script.
- El *parser* MSXML.DLL deberá estar instalado en el servidor de web (IIS o PWS)

introducción a <?xml?>

El documento de datos XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<guestbook>
  <item>
    <author>Alfredo</author>
    <email>alf@ibium.com</email>
    <datetime order="200005162319">
      16-5-2000 (23:19)
    </datetime>
    <ip>127.0.0.1</ip>
    <texto>Hola, esto es una prueba</texto>
  </item>
  <item>
    ...
  </item>
</guestbook>
```


introducción a <?xml?>

Aspecto final del libro de firmas

Ramiro Fernández

Esto es una prueba más, y aprovecho para comprobar que los acentos y las eñes aparecen correctamente.


 ramiro@yahoo.com

 16-5-2000 (23:44)

Alfredo

Hola, esto es una prueba

 alf@ibium.com

 16-5-2000 (23:19)

Nombre

Email

Texto:

Enviar

introducción a <?xml?>

Plantilla en HTML

```
<table width="500" border="0" align="center">

<tr bgcolor="#000000"><td colspan="2">
    <font size="2" face="Verdana" color="red">
        <b>Nombre del autor</b></font>
    </td></tr>
<tr><td colspan="2">
    <p><font face="Verdana" size="2">Texto</font></p>
</td></tr>
    <tr valign="middle" bgcolor="#CCCCCC"><td width="50%">
<a href="mailto:email@email">

    <font size="1" face="Verdana">email@email</font></a>
    </td>
    <td width="50%">
    <i><font size="1" face="Verdana">fecha y hora</font></i></td>
</tr>
</table>
```

introducción a <?xml?>

La hoja de estilo (I)

```
<?xml version='1.0' encoding='UTF-7'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">

<table width="500" border="0" align="center">

<xsl:for-each select="guestbook/item"
               order-by="-datetime/@order">
  <tr bgcolor="#000000"><td colspan="2">
    <font size="2" face="Verdana" color="red">
      <b><xsl:value-of select="author"/></b></font>
    </td></tr>
    <tr><td colspan="2">
      <p><font face="Verdana" size="2">
        <xsl:value-of select="texto"/></font></p>
      </td></tr>
  ...
```

introducción a <?xml?>

La hoja de estilo (II)

```
<tr valign="middle" bgcolor="#CCCCCC"><td width="50%">
  <xsl:if test="email"><a>
    <xsl:attribute name="href">mailto:<xsl:value-of select="email"/>
    </xsl:attribute>
    
    <font size="1" face="Verdana"><xsl:value-of select="email"/>
    </font></a>
  </xsl:if>
</td>
<td width="50%">
<i><font size="1" face="Verdana"><xsl:value-of select="datetime"/>
</font></i></td>
</tr>
</xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>
```

introducción a <?xml?>

Transformando XML en HTML

```
xmlDoc = "data.xml"  
xslDoc = "gbook.xsl"
```

```
Set xmlObj = CreateObject("Microsoft.XMLDOM")  
xmlObj.Async = False  
xmlObj.Load(Server.MapPath(xmlDoc))
```

```
Set xslObj = CreateObject("Microsoft.XMLDOM")  
xslObj.Async = False  
xslObj.Load(Server.MapPath(xslDoc))
```

```
Response.Write(xmlObj.transformNode(xslObj))
```

introducción a <?xml?>

Añadiendo datos al documento XML

```
stringNuevo = "<item>" & vbCrLf & _  
    "<author>" & nombre & "</author>" & vbCrLf & _  
    "<email>" & email & "</email>" & vbCrLf & _  
    ... & "</item>" & vbCrLf & vbCrLf  
Set xmlObj = CreateObject("Microsoft.XMLDOM")  
Set xmlObj2 = CreateObject("Microsoft.XMLDOM")  
  
xmlObj.Async = False  
xmlObj.Load(Server.MapPath("data.xml"))  
xmlObj2.Async = False  
xmlObj2.LoadXML(stringNuevo)  
Set root = xmlObj.documentElement  
Set root2 = xmlObj2.documentElement  
  
root.appendChild(root2)  
xmlObj.Save(Server.MapPath("data.xml"))
```

alfredo.reino@frogdesign.de

introducción a `<?xml?>`

XLL / XLink

Enlazando documentos

introducción a <?xml?>

XLink

- XLink es una aplicación XML que intenta superar las limitaciones que tienen los enlaces de hipertexto en HTML. Es una especificación que todavía está en desarrollo, y que todavía no tiene "rodaje" en el mundo real.
- Los enlaces en HTML tienen una serie de limitaciones, que los hacen bastante pobres. Por ejemplo, sólo tienen dos extremos, la terminación origen y la destino, y son unidireccionales.

introducción a <?xml?>

XLink

Si yo creo un enlace en mi página web personal que me lleve hasta la página principal de Coca-Cola, estoy creando un "vínculo" entre esta página y la mia.

```
<a href="http://www.cocacola.com/">Coca-Cola</a>
```

Este vínculo es unidireccional, porque un visitante cualquiera que entre en la página de Coca-Cola, no tiene forma de saber que mi página (así como otros cientos de miles) enlaza con ella.

introducción a <?xml?>

XLink

Con XLL/XLink un estudiante podría hacer anotaciones a los apuntes que un profesor tiene disponibles en la red, por el método de insertar un enlace desde los apuntes (a los que no tiene acceso de escritura o modificación) y su propia página de anotaciones y comentarios.

introducción a <?xml?>

XLink

O bien, podremos vincular dos páginas cualquiera, por ejemplo enlazando el texto de una noticia en un diario on-line, con el texto de la noticia equivalente en el diario competidor. Y no hay porqué quedarse en enlaces con dos extremos. Los enlaces extendidos permiten mucho más que eso.

En el momento actual, la tecnología para poder mantener una base de datos de enlaces mundial, no está desarrollada, pero se puede hacer a nivel local.

introducción a <?xml?>

XLink

Por ejemplo, nuestra empresa quiere comprar ciertos productos de un suministrador. Los trabajadores de la empresa podrán visitar la página web del proveedor y hacer enlaces externos a comentarios sobre las especificaciones de tal producto. Cuando otro compañero de la empresa visite la página, el servidor de enlaces de nuestra empresa le mostrará la página junto con los enlaces externos creados, y mostrar nuestros comentarios como si fueran parte del documento original.

introducción a `<?xml?>`

Aplicaciones de XML

Desarrollo de portales

introducción a <?xml?>

Portales

- Los portales son websites que integran información proveniente de muchas fuentes diferentes. Además permiten buscar información no estructurada.
- Una forma útil de gestionar la información en un portal de este tipo sería el desarrollar un sistema basado en XML para describir recursos (artículos, fotos, enlaces, vídeos, bases de datos, etc.)

introducción a <?xml?>

Portales

- Asimismo, cada elemento de que están compuestas las páginas (texto, módulos, formularios, etc.) se podría describir en XML.
- Un sistema de gestión del contenido (CMS, Content Management System) debería integrar y gestionar los documentos XML de las fuentes de información así como el lay-out de cada una de las páginas.

introducción a <?xml?>

Estrategias servir documentos XML

- Los documentos XML pueden llevar implícita la hoja de estilo XSL adecuada, y dejar que el browser haga la transformación.
 - Sólo una hoja de estilo por documento.
 - Problema de incompatibilidad de browsers.
 - Si cambia el fichero XSL, hay que cambiar todas las referencias en todos los documentos XML.

introducción a <?xml?>

Estrategias servir documentos XML

- La transformación de XML al formato aceptado por el browser (HTML/WML/XHTML/VRML...) se realiza del lado del servidor.
 - Mejor solución que la anterior.
 - Evitamos problemas de incompatibilidad.
 - Se puede hacer por medio de Java Servlets, CGI scripts, ASP, PHP3, etc...
 - Problema de sobrecarga del servidor cuando hay mucha actividad.

introducción a <?xml?>

Estrategias servir documentos XML

- La transformación de XML al formato aceptado por el browser se realiza del lado del servidor, pero off-line, generando ficheros destino estáticos.
 - Mismas ventajas que el método anterior.
 - El servidor trabaja menos.
 - No se puede hacer para ficheros que cambien cada vez que son cargados (resultados de una búsqueda, hora y fecha, etc.)

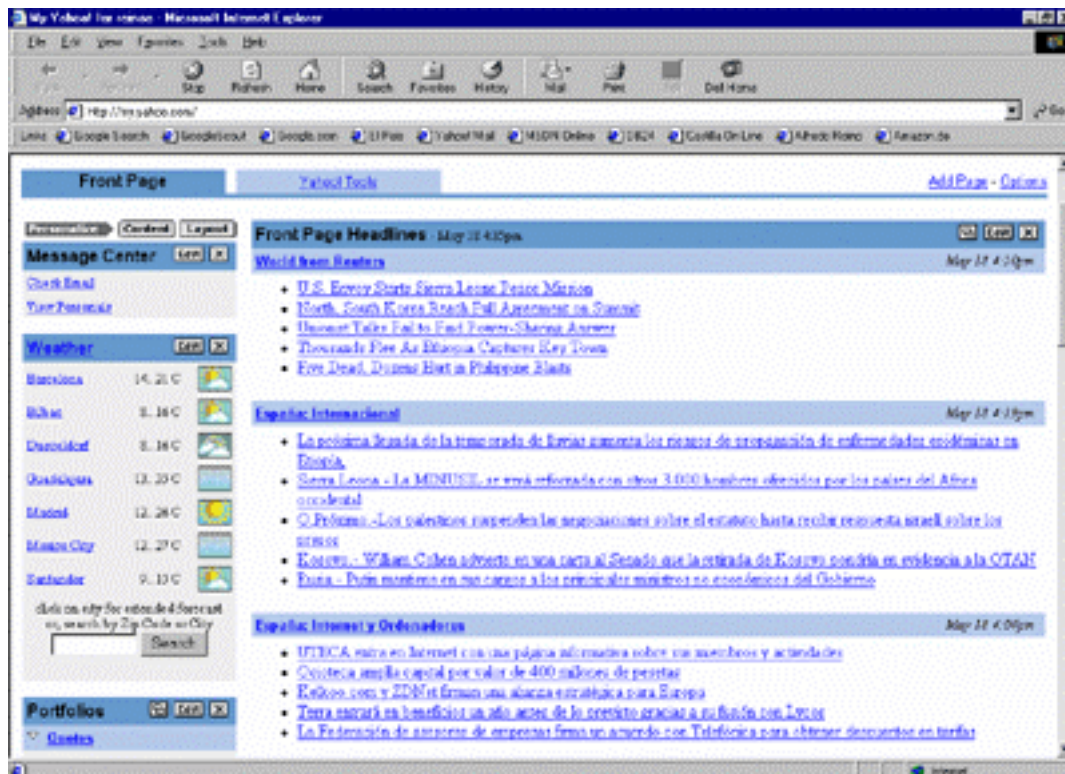
introducción a <?xml?>

Portales HTML/WML

- Con la aparición de teléfonos celulares que incorporan browsers WML (Wireless Mark-up Language) se impone la migración a sistemas basados en XML.
- Un sólo documento puede tener varias vistas en HTML y varias vistas en WML, simplemente usando diferentes hojas de estilo.
- Esto elimina problemas de sincronía de actualizaciones.

introducción a <?xml?>

Portales HTML/WML



Portal HTML

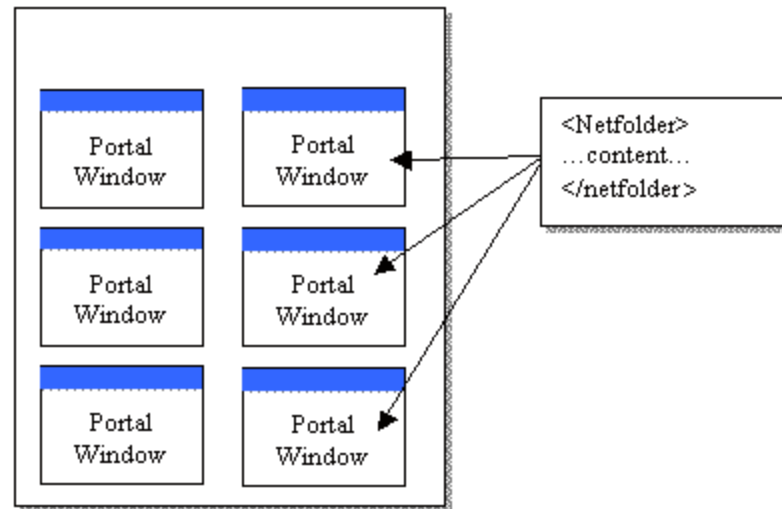
Portal WML



introducción a `<?xml?>`

Portales HTML

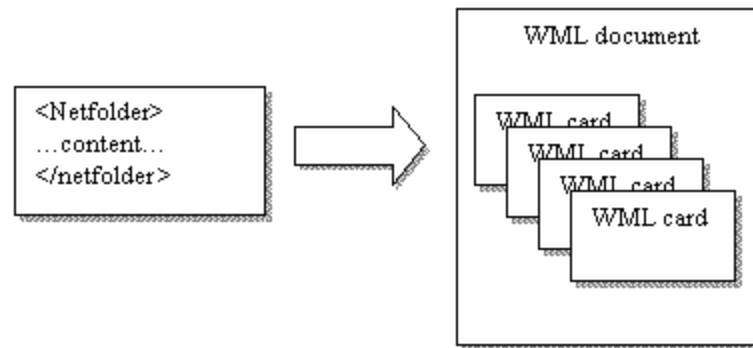
- Los portales HTML presentan la información en "módulos", que se pueden mapear a código XML de definición del lay-out.



introducción a <?xml?>

Portales WML

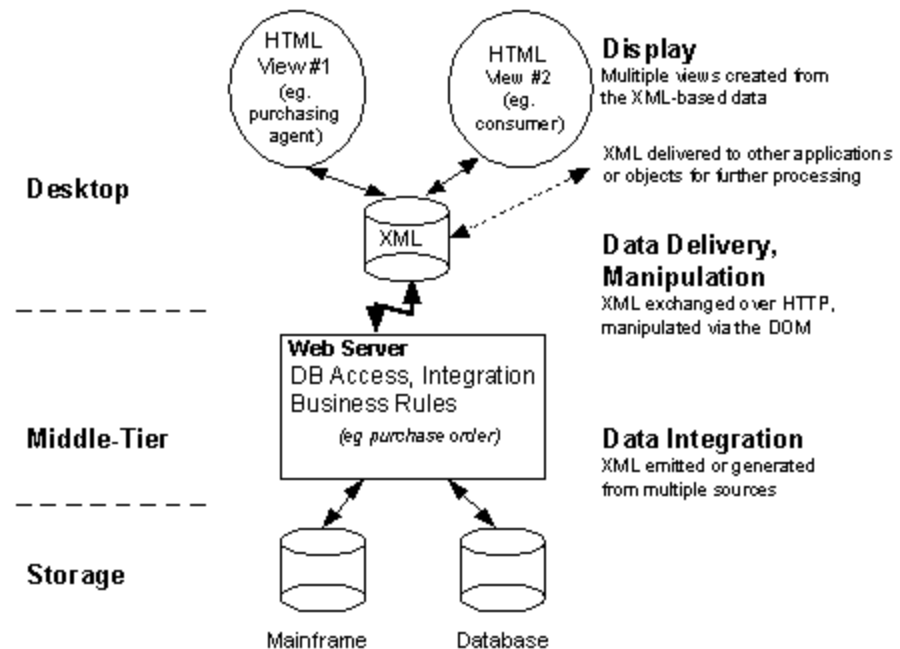
- Los portales WML presentan la información en "cards", que se pueden mapear al mismo código XML de definición del lay-out que en el caso anterior.



introducción a <?xml?>

Aplicaciones que impulsarán XML

Aplicaciones que exijan que el cliente web medie entre dos o más bases de datos. Se hará posible la integración de bases de datos distribuidas en los navegadores que admitan XML, pudiéndose modificar el contenido y la estructura de esta.



introducción a <?xml?>

Aplicaciones que impulsarán XML

Aplicaciones que intentan transferir una parte significativa de la carga del proceso del servidor al cliente web.

Funcionará con un subprograma Java que se insertará en el PC del cliente. Esta carga hará que muchas de las funciones de modificación puedan desarrollarse desde el mismo navegador web del cliente.

introducción a <?xml?>

Aplicaciones que impulsarán XML

Aplicaciones que precisen que el cliente web presente diferentes versiones de los mismos datos a diferentes usuarios. Se podría aplicar un manual de diferentes grados (iniciación, intermedio y avanzado) con sus diferentes idiomas, etcétera. Esto hará que este manual se pueda personalizar por los usuarios y extraer la información requerida de un capítulo determinado, con una ordenación y formatos concretos.

introducción a <?xml?>

Aplicaciones que impulsarán XML

Aplicaciones en las que agentes web inteligentes intentan adaptar la búsqueda de información a las necesidades de los usuarios individuales. Habrá una interacción entre la información requerida y las preferencias del usuario de la aplicación. Con el XML vendrá una segunda generación de robots que permitirá una mayor precisión de la búsqueda requerida. Actualmente podemos encontrar aplicaciones de medios de comunicación como los periódicos personalizados.

introducción a `<?xml?>`

Una llamada de advertencia...

El XML y sus tecnologías nos pueden parecer muy útiles y potentes, pero...

usemos la herramienta más adecuada a cada caso.

¡No por tener un martillo, todo se convierte en clavo!



introducción a <?xml?>

<http://www.ibium.com/alf/xml/>

Pueden encontrar esta presentación en la página indicada.