

Curso: “*Tecnología Web*”

Profesores:

Jose Emilio Labra Gayo (Universidad de Oviedo, España)
Raúl Monge (UTFSM, Chile)

***** Curso sobre Tecnología Web ***** Versión 2005

Contenido

1.- Tecnologías XML

Definición y validación: DTDs, Espacios de nombres, XML Schema
Transformación y consulta de XML: XSLT, Xpath, Xquery
Programación XML: SAX, DOM

2.-Servicios Web

SOAP, WSDL, UDDI

3.-Web Semántica

Fundamentos, RDF, OWL

4.-Nuevas aplicaciones

***** Curso sobre Tecnología Web ***** Versión 2005

Viernes		
Clase 1	14:30h – 16h	Lenguajes de marcado XML, DTDs NameSpaces
once		
Clase 2	16:30h – 18h	XML Schema Diseño Vocabularios XML
break		
Clase 3	18:15h – 20:15h	Acceso XML (XPath) Transformación XML (XSLT) Consultas XML (XQuery)
break		
Clase 4	20:30h – 22h	Programación XML: SAX, DOM, JAXB Aplicaciones XML: XHTML, SVG, SMIL, WML, X3D, etc

WWW = Mayor almacén de información jamás recopilado por la humanidad

Características

Grandes cantidades de información sobre cualquier asunto

Acceso *casi instantáneo* desde cualquier lugar con conexión a Internet

Sistema no centralizado ⇒ Cualquier persona puede añadir más información

Plataforma Multimedia (Texto, Imágenes, Vídeo, etc.)

Identificación de recursos unificada (URIs)

Grandes Retos

Integración de aplicaciones

Búsqueda de lenguajes comunes: Estandarización

Interoperabilidad: computación ubicua

Modelos Orientados a Servicios

Automatización de tareas

Representaciones *comprensibles* por las máquinas

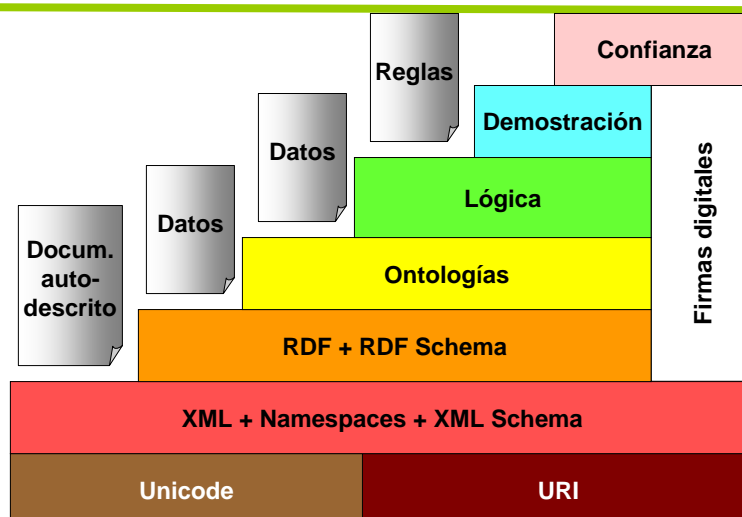
Creación de agentes autónomos

Accesibilidad

Acceso universal

Atención a todos los usuarios: discapacidades, entornos diferentes, etc.

Arquitectura de la Web



Representación de Información Bits, bytes, números, caracteres, ...

Los ordenadores manejan código binario: 0s y 1s

Bytes: Grupos de 8 bits

Números: Sistemas de codificación binaria, octal, hexadecimal...

Caracteres: Código que asocia a cada carácter un nº

ASCII: 7 bits \Rightarrow (0 – 127)

(A)merican (S)tandard (C)ode for (I)nformation (I)nterchange

Extensiones de ASCII

ISO-8859-1 (iso-latin-1)

(8 bits) ASCII (0-127) + otros caracteres típicos de Europa occidental

Familia ISO-8859-X = Otros alfabetos europeos

ISO-8859-15 (iso-latin-9) Igual que iso-8859-1 + símbolo de €

¡CUIDADO! ...hay muchos idiomas y muchos caracteres...
⌘ € き せ 𐀀 𐀁 𐀂 𐀃 𐀄 𐀅 𐀆 𐀇 𐀈 𐀉

***** Curso sobre Tecnología Web ***** Versión 2005

Unicode

ISO-10646 (31 bits) Define un repertorio universal de caracteres (UCS)

En continua revisión: ISO-10646-2:2001 contiene más de 70.000 caracteres

UNICODE = Consorcio de empresas que define restricciones sobre la implementación de ISO-10646

Varias codificaciones (UTF = Unicode Transformation Format)

- UTF-8: Los primeros 127 códigos se presentan igual (compatible con ASCII)

El resto se codifican en longitud variable

Relativamente Eficiente

- UTF-16: Usa 16bits para los caracteres más comunes, el resto con pares de 16 bits

- UTF-32: Codificación directa en 32 bits (desperdicio de espacio)

NOTA: Conviene distinguir:

Carácter: Entidad abstracta (Letra A)

Glifo (Glyph): Representación del carácter A А А А А А

Fuente (Font): Conjunto de glyphs, ejemplo: Times Roman, Arial, etc.

***** Curso sobre Tecnología Web ***** Versión 2005

Información multimedia

Imágenes: En pantalla = matriz de puntos de colores (pixels)

Formatos Raster (Raw): Se enumeran todos los puntos con sus colores

Ejemplo: Bitmap, TIFF

Compresión: diversos algoritmos de compresión

GIF: Utiliza 8 bits (hasta 256 colores)

Byte de color = Índice en la paleta de colores

JPEG: utiliza 24 bits (hasta 16 millones de colores)

Vectorial: Se enumeran las instrucciones de dibujo

Ejemplos: DXF, [SVG](#) (estándar de Internet)

Sonido: Formatos raster (WAV) y comprimidos (MP3)

Vídeo: Formatos comprimidos (MPEG)

Realidad Virtual: Lenguaje de Modelado (VRML, [X3D](#))

Identificación Recursos

URI: (Uniform resource Identifier) Identifica un recurso de forma global

Puede sub-clasificarse en:

URL (Uniform resource locator)

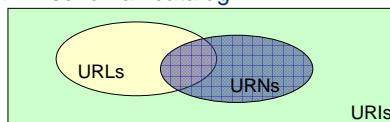
Además de identificar el recurso, indica cómo llegar hasta él

esquema: //servidor:puerto /ruta?datosGET

<http://www.uniovi.es:8080/prueba/carrito?action=print>

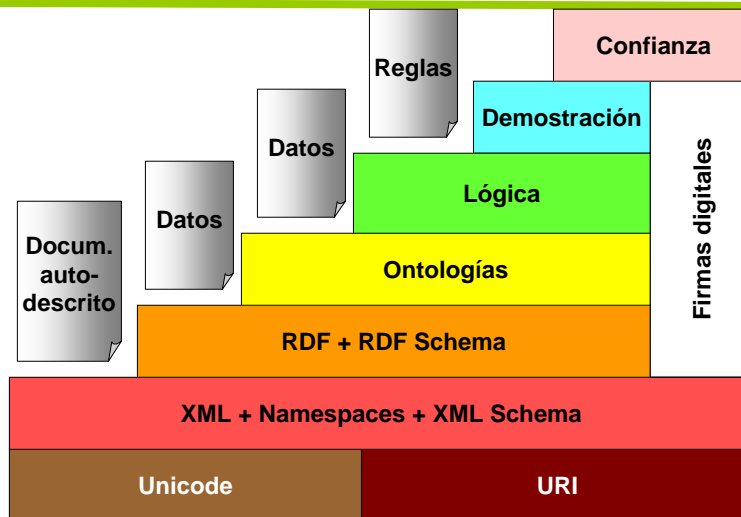
URN (Uniform resource name): Nombre de recurso

Ejemplo: [urn:xmlog:objects:schema:xmlschema:xcatalog](#)



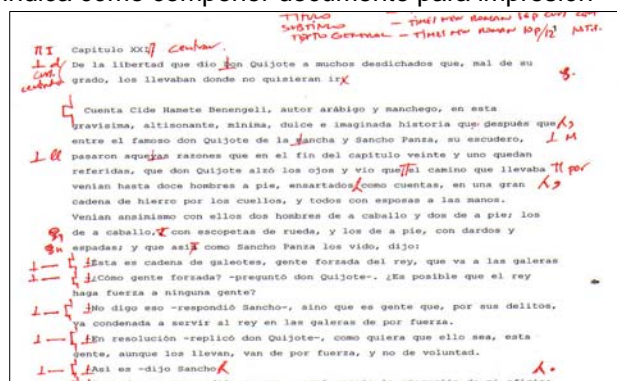
IRIs (Internationalized Resource Identifiers)

permiten utilizar caracteres Unicode en los identificadores



Orígenes: Industria de la Publicación

Uso de marcas = indica cómo componer documento para impresión.



Ejemplo de corrección tipográfica

Fuente: <http://recursos.cnice.mec.es/bancoimagenes>

Mercado de documentos

Sólo texto

ALBA Abril de 1915 Granada Mi corazón oprimido
siente junto a la alborada el dolor de sus
amores y el sueño de las distancias.

Texto marcado

```
[ALBA[ ← Título, negrita, centrado, 14pt
Abril de 1915[← SubTítulo, negrita, centrado
Granada[← SubTítulo cursiva, centrado
Mi corazón oprimido [← Verso, 10pt
siente junto a la alborada [← Verso
el dolor de sus amores [← Verso
y el sueño de las distancias. [← Verso
```

Resultado

ALBA
Abril de 1915
Granada

Mi corazón oprimido
 siente junto a la alborada
 el dolor de sus amores
 y el sueño de las distancias.

Mercado Descriptivo

El marcado **no** es la información que contiene el documento

Marcado = información acerca del documento = meta-información

Lenguajes de Marcado descriptivo: Incluyen marcas que describen cómo procesar el documento

Ejemplo: HTML

```
<html>
<head><title>Poema</title></head>
<body lang="es">
<h1>Alba</h1>
<h2>Abril de 1915 </h2>
<h2><i>Granada</i></h2>
<p>Mi corazón oprimido</p>
<p>siente junto a la alborada</p>
<p>el dolor de sus amores</p>
<p>y el sueño de las distancias. </p>
</body>
</html>
```



Marcado Generalizado

Marcado generalizado = Sintaxis común que facilita la creación de lenguajes descriptivos

HTML

```

<html>
<head><title>Poema</title></head>
<body lang="es">
<h1>Alba</h1>
<h2>Abril de 1915 </h2>
<h2><i>Granada</i></h2>
<p>Mi corazón oprimido</p>
<p>siente junto a la alborada</p>
<p>el dolor de sus amores</p>
<p>y el sueño de las distancias.</p>
</body>
</html>
    
```

Otras marcas...(misma sintaxis)

```

<poema fecha="Abril de 1915"
      lugar="Granada">

<titulo>Alba</titulo>

<verso>Mi corazón oprimido</verso>
<verso>siente junto a la alborada</verso>
<verso>el dolor de sus amores</verso>
<verso>y el sueño de las distancias. </verso>
</poema>
    
```

Sintaxis común

```

<etiqueta atrib="valor">contenido
</etiqueta>
    
```

.....

Curso sobre Tecnología Web

..... Versión 2005

Marcado Generalizado SGML

(70-) GML desarrollado en IBM – Generalized Markup Language
(Goldfarb, Mosher, Lorie)

(86) SGML Standard Generalized Markup Language (Estándar ISO)

Utilizado para el intercambio de documentos

Principio: Separar contenido de la forma de representarlo

Muy flexible (permite definir vocabularios específicos para cada aplicación)

HTML era un vocabulario de SGML

.....

Curso sobre Tecnología Web

..... Versión 2005

Marcado Generalizado XML

Desarrollado por T. Bray, J. Paoli, C. M. Sperberg-McQueen (1995)

T. Bray: Consultor de Textuality para Netscape (ahora en Sun)

C. M. Sperberg-McQueen (TEI, SGML)

J. Paoli (Microsoft)

Objetivos:

Crear una versión simplificada de SGML para la Web

20% de características de SGML ⇔ 80% de funcionalidad de SGML

Detalle (Especificación de XML = 26 páginas, de SGML > 500)

XML

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

Declaración de
XML

```
<!DOCTYPE raíz[  
...  

```

Declaración de
Tipo DTD
Opcional

```
<raíz>  
  <elemento>  
  ...  
  </elemento>  
</raíz>
```

Documento

Declaración de XML

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

version: Actual = 1.0

Borrador de versión 1.1

Mayor compatibilidad con Unicode

Identificadores: Permite cualquier carácter Unicode

encoding: UTF-8, UTF-16, iso-8859-1, etc.

standalone: Indica si el documento no hace referencias a entidades externas

Documentos XML

Los documentos consisten en una serie de datos marcados mediante etiquetas

Las etiquetas describen la estructura del documento

Un elemento = grupo formado por etiqueta inicial, etiqueta final y contenido entre ambas. La etiqueta inicial puede incluir atributos.

```
<etiqueta atributo="valor">.....</etiqueta>
```

Distinción
minúsculas/mayúsculas

Elemento vacío: Entre la etiqueta inicial y final no hay información:

```
<etiqueta atributo="valor"></etiqueta>
```

↓

```
<etiqueta atributo="valor"/>
```

Anidamiento

Se pueden anidar elementos

```

<externo>
  <interno>texto</interno>
</externo>
  
```



...pero no se pueden entrelazar:

```

<externo>
  <interno>texto</externo>
</interno>
  
```



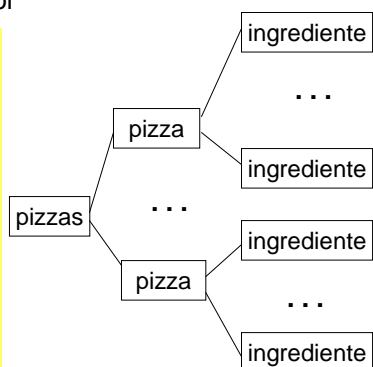
Estructura General

Sólo puede haber un único elemento raíz

 Cada documento XML equivale a un árbol

```

<pizzas>
  <pizza nombre="Barbacoa" precio="8">
    <ingrediente nombre="Salsa Barbacoa" />
    <ingrediente nombre="Mozzarella" />
    <ingrediente nombre="Pollo" />
    <ingrediente nombre="Bacon" />
    <ingrediente nombre="Ternera" />
  </pizza>
  ...
  <pizza nombre="Margarita" precio="6">
    <ingrediente nombre="Tomate" />
    <ingrediente nombre="Jamón" />
    <ingrediente nombre="Queso" />
  </pizza>
</pizzas>
  
```



Cada elemento puede contener atributos en la etiqueta inicial

```

<pizza nombre="Margarita" precio="6">
  . . .
</pizza>
  
```

El orden de los atributos no es significativo

No puede haber 2 atributos con el mismo nombre

Atributos predefinidos:

xml:lang: Especifica el idioma.

Por ejemplo: **en** (inglés), **sp** (español)

xml:space: Especifica cómo tratar el espacio en blanco.

Valores: **preserve** = Mantenerlo

default = Permitir a la aplicación que lo trate como quiera.

Comentarios

**<!-- el texto de un comentario
no es analizado -->**

Caracteres especiales: No pueden incluirse directamente

```

<código>
  if x &lt; 4 then x:=x + 1;
</código>
  
```

```

&lt; <
&gt; >
&quot; "
&apos; '
&amp; &
  
```

Secciones CDATA

Si se desea introducir código sin analizar

```

<código>
  if x < 3 && x > 4 then
    print "Hola"
</código>
  
```

```

<código>
  if x &lt; 3
    &amp;&amp; x &gt; 4 then
    print "Hola"
</código>
  
```



```

<código>
<![CDATA[
  if x < 3 && x > 4 then
    print "Hola"
]]>
</código>
  
```

Instrucciones de Procesamiento

Es posible incluir instrucciones que indican al procesador alguna acción a realizar

Sintaxis: `<?aplicación datos ?>`

Pueden utilizarse para asociar una hoja de estilos al documento:

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

...o para otros propósitos especiales

En realidad la declaración de documento es una instrucción de procesamiento

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

Documento bien formado

Documento bien formado

Sigue las reglas sintácticas

Importante:

Contiene un único elemento raíz

Todas las etiquetas están correctamente anidadas

```
<pizzas>
  <pizza nombre="Margarita" precio="6">
    <ingrediente nombre="Tomate" />
    <ingrediente nombre="Queso" />
  </pizza>
</pizzas>
```



```
<pizzas>
  <pizza nombre="Margarita" precio="6">
    <ingrediente nombre="Tomate" >
  </pizzas>
```

Documento válido

Se puede incluir una declaración del tipo de documento

```

<?xml version="1.0"?>
<!DOCTYPE pizzas SYSTEM "pizzas.dtd">
<pizzas>
  <pizza nombre="Margarita" precio="6">
    <ingrediente nombre="Tomate" />
  </pizza>
</pizzas>
    
```

pizzas.dtd

```

<!ELEMENT pizzas (pizza*)>
<!ELEMENT pizza (ingrediente*)>
<!ELEMENT ingrediente (#PCDATA)>
<!ATTLIST pizza nombre CDATA #REQUIRED>
<!ATTLIST pizza precio CDATA #REQUIRED>
<!ATTLIST ingrediente nombre CDATA #REQUIRED>
    
```

Documento válido

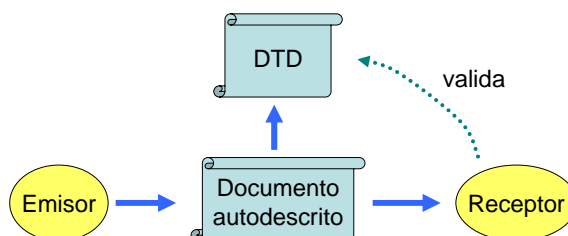
Está bien formado y

La estructura encaja con la declaración del tipo de documento

Declaración Tipo Documento DTD

La DTD permite especificar la estructura del documento

La DTD puede estar separada del documento



Declaración de tipo de documento (DTD)

DTD interno

```

<?xml version="1.0"?>
<!DOCTYPE pizzas [
  <!ELEMENT pizzas (pizza*)>
  ...
]>
<pizzas>... </pizzas>
    
```

También es posible especificar un DTD externo y añadir definiciones locales

DTD externo

SYSTEM (DTDs de ámbito local)

```

<?xml version="1.0"?>
<!DOCTYPE pizzas SYSTEM "http://www.mafia.it/pizzas.dtd" >
<pizzas>
  ...
</pizzas>
    
```

PUBLIC (DTDs compartidos por diversas organizaciones)

```

<?xml version="1.0"?>
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.0//EN"
  "http://www.w3c.org/TR/REC-html/strict.dtd">
    
```

DTD Tipos de declaraciones

ELEMENT

Elementos del documento XML

ATTLIST

Lista de atributos de un elemento

ENTITY

Entidades (≈variables o macros)

NOTATION

Definen tipos de contenidos

Facilitan la inclusión de formatos binarios (imágenes, vídeos, sonidos, ...)

DTD Elementos

(?) = 0, 1 elemento
 (*) = 0 ó más elementos
 (+) = 1 ó más elementos
 (|) = alternativa
 (,) = secuencia
 EMPTY = vacío
 ANY = cualquier estructura de subelementos
 #PCDATA = cadena de caracteres analizados

```

<ELEMENT pizza (ingrediente*, inventor?)>
<ELEMENT servicio (domicilio | restaurante) >
<ELEMENT ingrediente EMPTY>
<ELEMENT inventor (#PCDATA)>
<ELEMENT sección (título, (contenido | sección+))>
<ELEMENT p (#PCDATA | a | ul | em)* >
    
```

Rekursividad

PCDATA = Parsed Character Data
 Indica que los datos son analizados buscando etiquetas

5

DTD Atributos

Tipos de datos

CDATA = Cadena de caracteres
 NMTOKEN = Palabra (sin espacios)
 NMTOKENS = Lista de palabras
 Enumeración separada por |
 ID = Nombre único (sin duplicados)
 IDREF = Su valor debe apuntar a un ID

Valor de los Atributos

#REQUIRED Obligatorio
 #IMPLIED Opcional
 #FIXED Constante
 Valor Valor por defecto

```

<!ATTLIST pizza nombre CDATA #REQUIRED>
<!ATTLIST ingrediente nombre CDATA #REQUIRED
                    calorías CDATA #IMPLIED>
<!ATTLIST precio moneda (euros|dólares) #REQUIRED
                    valor CDATA #REQUIRED>
<!ATTLIST persona código ID #REQUIRED>
<!ATTLIST dueño código IDREF #REQUIRED>
<!ATTLIST conOrégano (sí|no) "sí">
<!ATTLIST impuesto tipo CDATA #FIXED "IVA">
    
```

```

<pizza nombre="4 estaciones" >
  <ingrediente nombre="Jamón" />
  <precio moneda="euros" valor="7" />
</pizza>
    
```

```

<persona código="23" nombre="Juan" />
<persona código="35" nombre="Pepe" />
<persona código="37" nombre="Luis" />
    
```

```

<dueño código="35" />
<impuesto tipo="IVA" />
    
```


DTD Entidades Generales

Entidades: Asignan nombres a ciertos elementos (similar a variables)

Se denotan por &entidad;

No se admite recursividad

```
<!ENTITY marg "Pizza Margarita">
<!ENTITY queso "<ingrediente nombre='queso' />" >
```

```
<pizza nombre="&marg;" precio="7">
&queso;
</pizza>
```

```
<pizza nombre="Pizza Margarita" precio="7">
<ingrediente nombre='queso' />
</pizza>
```

Entidades numéricas: Código numérico del carácter

```
&#x2200; ∅      &#8707; ∃
```

Entidades predefinidas: Permiten incluir etiquetas sin analizar

```
&lt;    <      &quot;    "      &apos;    '
&gt;    >      &amp;    &
```

DTD Entidades externas

Permiten usar archivos externos (Incluir otros documentos XML)

pizzas.xml

```
<pizzas>
<pizza nombre="4 Quesos" precio="7">
<ingrediente nombre="Jamón" />
<ingrediente nombre="Queso" />
</pizza>
...
</pizzas>
```

personal.xml

```
<personal>
<trabajador
  nombre=" Benito Alcaparra" >
...
</trabajador>
...
</personal>
```

establecimiento.dtd

```
<!ELEMENT establecimiento ANY>
<!ENTITY personal SYSTEM "personal.xml">
<!ENTITY pizzas SYSTEM "pizzas.xml">
```

establecimiento.xml

```
<establecimiento
  nombre="Pizzería Al Capone">
&personal;
&pizzas;
</establecimiento>
```

DTD Entidades externas

Las entidades externas combinadas con notaciones permiten incluir archivos externos de datos binarios

```
<!NOTATION gif SYSTEM "gifEditor.exe">  
<!ENTITY dibujo SYSTEM "logotipo.gif" NDATA gif>
```

```
<información>  
<logotipo>&dibujo;</logotipo>  
</información>
```

DTD Entidades Parámetro

Permiten dar nombres a partes de un DTD

Se denotan por %entidad;

```
<!ENTITY establecimiento (nombre, dueño?, {calle, número?, ciudad, país, códigoPostal}) >  
<!ENTITY persona (dni, nombre {calle, número?, ciudad, país, códigoPostal}) >
```

```
<!ENTITY %localización "calle, número?, ciudad, país, códigoPostal" >  
<!ENTITY establecimiento (nombre, dueño?, %localización;)>  
<!ENTITY persona (dni, nombre, %localización;)>
```

Entidades externas: Permiten incluir elementos externos en una DTD

Aplicación: Dividir la definición de una DTD en varios documentos

```
<!ENTITY %persona SYSTEM "persona.dtd">  
<!ENTITY %establecimiento SYSTEM "establecimiento.dtd">  
%persona;  
%establecimiento;
```

Discusión sobre XML: Ventajas

- Es un formato estructurado
 - Contiene información y meta-información
- Ha sido diseñado específicamente para Internet
 - Soportado por visualizadores y servidores
- Numerosas herramientas de procesamiento
- Legible por personas humanas
- Admite la definición de vocabularios específicos
- Separa contenido del procesamiento y visualización
- Aumenta la seguridad mediante la validación de documentos
- Formato abierto, respaldado por numerosas organizaciones
- Una vez definido un DTD común, facilita intercambio de información



Discusión sobre XML: Inconvenientes

- Puede requerir demasiado espacio, ancho de banda y tiempo de procesamiento
 - Documentos largos con mucha información redundante
 - Problemas para Bases de Datos
- Es una sintaxis de documentos, no un lenguaje de programación
- Es posible crear formatos y vocabularios propietarios
- Puede fomentar la proliferación de vocabularios específicos
- Bueno para texto, malo para datos binarios



```
<?xml version="1.0">
<imagen formato="base64">
DS34JSCDF029876D76523981DFNDF3F2134F5FD019A
FGF23DAND345CD2135911943DCBKAPFGDAJJK32A10
....
</imagen>
```

Espacios de Nombres

Problema de la Homonimia

Homonimia: Mismo nombre con diferentes propósitos

```
<país nombre="Francia">  
<capital>París</capital>  
</país>
```

```
<inversión>  
  <capital>7000€</capital>  
</inversión>
```

¿Cómo combinar en el mismo documento estos vocabularios?

```
<inversiones>  
  <país nombre="Francia">  
    <capital>París</capital>  
    <capital>1200€</capital>  
  </país>  
  . . .  
</inversiones>
```

Ambigüedad


Posibles Soluciones

Asignar un nombre único a cada etiqueta...

Posibles soluciones:

Crear una autoridad mundial que asigne nombres...

... o usar un mecanismo ya existente: URIs

Una URI es un identificador global único

Ejemplo: <http://www.aulanet.uniovi.es>

SOLUCIÓN:

Asociar a cada etiqueta una URI que indica a qué espacio de nombres pertenece...

`[http://www.bolsa.com]:capital`

`[http://www.geog.es]:capital`

Posibles soluciones

Solución fácil...

Asociar a cada etiqueta una URI

```
<[http://www.bolsa.com]:inversiones>
  <[http://www.geog.es]:país
    [http://www.geog.es]:nombre="Francia">
  <[http://www.geog.es]:capital>París
</[http://www.geog.es]:capital>
<[http://www.bolsa.com]:capital>1200€
</[http://www.bolsa.com]:capital>
</[http://www.bolsa.com]:país>
. . .
</[http://www.bolsa.com]:inversiones>
```

Legibilidad...



Solución: Asociar un alias a los elementos de un espacio de nombres dentro de un ámbito

`xmlns:alias` define *alias* en el ámbito de un elemento

```

<b:inversiones
  xmlns:b="http://www.bolsa.com"
  xmlns:g="http://www.geog.es">
  <g:país g:nombre="Francia">
    <g:capital>París</g:capital>
    <b:capital>1200€</b:capital>
  </g:país>
  . . .
</b:inversiones>
    
```

NOTA: Las URIs sólo se utilizan para que el nombre sea único, no son enlaces, ni tienen que contener información

Es posible ir asociando espacios de nombres a los elementos según van apareciendo

```

<b:inversiones
  xmlns:b="http://www.bolsa.com">
  <g:país
    xmlns:g="http://www.geog.es"
    g:nombre="Francia">
    <g:capital>París</g:capital>
    <b:capital>1200€</b:capital>
  </g:país>
  . . .
</b:inversiones>
    
```

Espacio de nombres por defecto

Mediante `xmlns="..."` se define un espacio de nombres por defecto (sin alias)

```

<inversiones
  xmlns="http://www.bolsa.com">
  <g:país
    xmlns:g="http://www.geog.es"
    g:nombre="Francia">
    <g:capital>París</g:capital>
    <capital>1200€</capital>
  </g:país>
  . . .
</inversiones>

```

Se refiere a
<http://www.bolsa.com>

Validación con DTDs

Posteriores a los DTDs, por tanto, los DTDs no dan soporte a Espacios de Nombres

Hay que definir los espacios de nombre usados

```

<!DOCTYPE inversiones [
<!ELEMENT inversiones (g:país*)>
<!ELEMENT g:país (g:capital,capital) >
<!ELEMENT g:capital (#PCDATA)>
<!ELEMENT capital (#PCDATA)>
<!ATTLIST inversiones
  xmlns CDATA #FIXED "http://www.bolsa.com">
<!ATTLIST g:país
  g:nombre CDATA #REQUIRED
  xmlns:g CDATA #FIXED "http://www.geog.es">
]>

```

Ampliamente utilizados para combinar vocabularios
Facilitan la incorporación de elementos no previstos inicialmente
Sintaxis *extraña* al principio
 Uso de prefijos
 URIs como elemento diferenciador...pero las URLs también sirven para acceder a recursos
Difícil combinación con DTDs

XML Schema

Lenguajes de Esquemas

Esquema = definición de estructura de un conjunto de documentos XML

Validar = Chequear que un documento sigue un esquema

Principal Ventaja: Protección de errores

Otras aplicaciones: Edición, compresión, etc.

DTDs = un ejemplo de esquemas (con varias limitaciones)

XML Schema = desarrollo posterior del W3c

Existen Otros:

RELAX-NG, Schematron, etc.

Características de DTD's

Especifican estructura del documento:

Elementos, atributos, anidamientos, etc.

Integridad referencial mínima (ID, IDREF)

Mecanismo sencillo de abstracción

Entidades ≈ Macros

Inclusión de documentos externos

Integrados en XML (Parte de la especificación)

Sencillos de comprender (≈ Expresiones regulares)



Limitaciones de DTD's



La Sintaxis **no es XML** (difíciles de manipular)

No soportan **Espacios de nombres**

No permiten especificar **tipos de datos** (por ejemplo: enteros, flotantes, fechas, etc.

No permiten especificar **secuencias no ordenadas**

((e1,e2,e3)|(e1,e3,e2)|(e2,e1,e3)|...(e3,e2,e1))

No hay soporte para declaraciones **sensibles al contexto**: Los elementos se definen todos a nivel de documento, ejemplo, contenido con el mismo nombre cuya estructura cambia en diferentes contextos

Soporte limitado para **Referencias cruzadas**, no es posible formar claves a partir de varios atributos o de elementos

No son extensibles (una vez definido, no es posible añadir nuevos vocabularios a un DTD)

XML Schema Objetivos de Diseño

Sintaxis XML

Soporte para Espacios de Nombres

Mayor expresividad

Restricciones numéricas

Integridad dependientes del contexto

Tipos de datos

Gran cantidad de tipos de datos predefinidos

Creación de tipos de datos por el usuario

Extensibilidad

Inclusión/Redefinición de esquemas

Herencia de tipos de datos

Soporte a Documentación

Ejemplo

alumnos.xsd

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.uniovi.es/alumnos"
  xmlns="http://www.uniovi.es/alumnos">
  <xs:element name="alumnos">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="alumno" minOccurs="1" maxOccurs="200"
          type="TipoAlumno"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="TipoAlumno">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="apellidos" type="xs:string"/>
      <xs:element name="nacim" type="xs:gYear"/>
    </xs:sequence>
    <xs:attribute name="dni" type="xs:string"/>
  </xs:complexType>
</xs:schema>

```

Elemento raíz **schema** y
espacio de nombres
determinado

Permite especificar
rangos de inclusión

Permite especificar
tipos

05

Validación

alumnos.xsd

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.uniovi.es/alumnos"
  xmlns="http://www.uniovi.es/alumnos">
  <xs:element name="alumnos">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="alumno" minOccurs="1" maxOccurs="200"
          type="TipoAlumno"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="TipoAlumno">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="apellidos" type="xs:string"/>
      <xs:element name="nacim" type="xs:gYear"/>
    </xs:sequence>
    <xs:attribute name="dni" type="xs:string"/>
  </xs:complexType>
</xs:schema>

```

Los espacios de nombres
deben coincidir.
También puede usarse:
xsi:noNameSpaceLocation

alumnos.xml

```

<alumnos
  xmlns="http://www.uniovi.es/alumnos"
  xsi:SchemaLocation="http://www.uniovi.es/alumnos
    alumnos.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  ...
</alumnos>

```

05

Tipos Anónimos vs. con nombre

```
<xs:element name="alumno">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:element>
```

+ legible

```
...
<xs:element name="alumno" type="TipoAlumno"/>
...
<xs:complexType name="TipoAlumno">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

+ Reutilizable

Otra posibilidad: Referencias

```
<xs:element name="alumno">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:element>
```

```
<xs:element name="alumnos">
  <xs:sequence>
    <xs:element ref="alumno" />
  </xs:sequence>
</xs:element>
```

Agrupaciones

Es posible nombrar agrupaciones de elementos y de atributos para hacer referencias a ellas

```

<xs:group name="nombApell">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:group>

```

```

<xs:complexType name="TipoAlumno">
  <xs:group ref="nombApell" />
  <xs:element name="carrera" type="xs:string"/>
</xs:complexType>

```

Tipos Complejos: Secuencia

Tipos Complejos: Son tipos que pueden contener elementos o atributos
Construcción básica mediante enumeración de elementos

```

<xs:complexType name="TipoAlumno">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
    <xs:element name="nacim" type="xs:gYear"
      minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="dni" type="xs:integer"/>
</xs:complexType>

```

```

<alumno dni="9399390">
  <nombre>Juan</nombre>
  <apellidos>García García</apellidos>
  <nacim>1985</nacim>
</alumno>

```

Tipos Complejos: Alternativa

choice: Representa alternativas

OJO: Es una o-exclusiva

```

<xs:complexType name="Transporte">
  <xs:choice>
    <xs:element name="coche" type="xs:string"/>
    <xs:element name="tren" type="xs:string"/>
    <xs:element name="avión" type="xs:string"/>
  </xs:choice>
</xs:complexType>

```

```

<transporte>
  <coche>Renault R23</coche>
</transporte>

```

Tipos Complejos: Contenido Mixto

El contenido Mixto permite mezclar texto con elementos

```

<xs:complexType name="TCom" mixed="true">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="emph" type="xs:string"/>
  </xs:choice>
</xs:complexType>

<xs:element name="comentarios" type="TCom" />

```

```

<comentarios>
  Es un poco <emph>listillo</emph>
</comentarios>

```

Secuencias no ordenadas

all = Todos los elementos en cualquier orden

En DTDs requería enumerar las combinaciones:

$(A,B,C)|(A,C,B)|\dots|(C,B,A)$

```
<xs:complexType name="TipoLibro">
  <xs:all>
    <xs:element name="autor" type="xs:string"/>
    <xs:element name="título" type="xs:string"/>
  </xs:all>
</xs:complexType>
<xs:element name="libro" type="TipoLibro" />
```

```
<libro>
  <autor>Juanita la Loca</autor>
  <título>No estoy loca</título>
</libro>
```

```
<libro>
  <título>El kigote</título>
  <autor>Cerbantes</autor>
</libro>
```

Tipos Simples

Los tipos simples son lo que no pueden contener elementos o atributos

Son los valores de atributos o el contenido de elementos básicos

Pueden ser:

- Predefinidos o *built-in* (Definidos en la especificación)

 - Primitivos

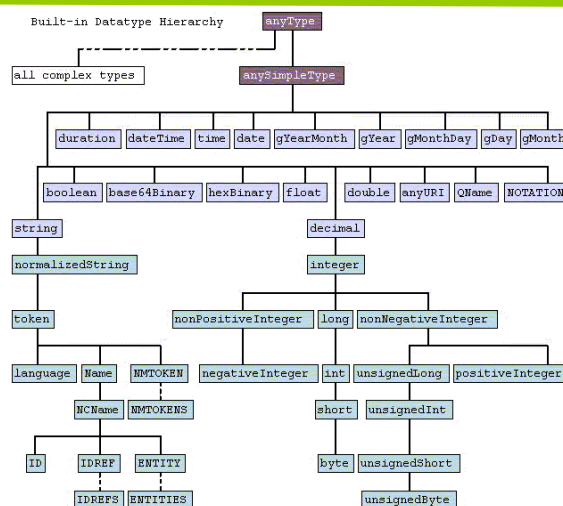
 - Derivados

- Definidos por el usuario (a partir de tipos predefinidos)

Tipos Primitivos

string
 boolean
 number, float, double
 duration, dateTime, time, date, gYearMonth, gYear, gMonthDay,
 gDay, gMonth
 hexBinary, base64Binary
 anyURI
 QName = Nombre cualificado con espacio de nombres
 NOTATION = Notación binaria (similar a DTD)

Jerarquía de tipos



Facetas de Tipos

Facetas fundamentales:

- equal*: Igualdad entre valores de un tipo de datos
- ordered*: Relaciones de orden entre valores
- bounded*: Límites inferiores y superiores para valores
- cardinality*: Define si es finito o infinito (contable, no contable)
- numeric*: Define si es numérico o no

Facetas de restricción

- length, minlength, maxlength*: Longitud del tipo de datos
- pattern*: Restricciones sobre valores mediante expresiones regulares
- enumeration*: Restringe a una determinada enumeración de valores
- whitespace*: Define política de tratamiento de espacios (preserve/replace, collapse)
- (max/min)(in/ex)clusive*: Límites superiores/inferiores del tipo de datos
- totalDigits, fractionDigits*: número de dígitos totales y decimales

Enumeraciones y Restricciones

Enumeración

```
<xs:simpleType name="TipoCarrera">
  <xs:restriction base="xs:token">
    <xs:enumeration value="Gestión"/>
    <xs:enumeration value="Sistemas"/>
  </xs:restriction>
</xs:simpleType>
```

Restricciones sobre valores

```
<xs:simpleType name="mes">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1" />
    <xs:maxInclusive value="31" />
  </xs:restriction>
</xs:simpleType>
```

```

<xs:simpleType name="ComponentesRGB">
  <xs:list itemType="ComponenteRGB"/>
</xs:simpleType>

<xs:simpleType name="ComponenteRGB">
  <xs:restriction base="xs:nonNegativeInteger">
    <xs:maxInclusive value="255" />
  </xs:restriction>
</xs:simpleType>

```

Se pueden aplicar las facetas: length, maxLength, minLength, enumeration

```

<xs:simpleType name="ColorRGB">
  <xs:restriction base="ComponentesRGB">
    <xs:length value="3" />
  </xs:restriction>
</xs:simpleType>

```

```
<color>255 255 0</color>
```

```

<xs:simpleType name="TipoNota">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:float">
        <xs:maxInclusive value="10" />
        <xs:minInclusive value="0" />
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="No presentado" />
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>

```

```
<nota> 5.75 </nota>
```

```
<nota> No presentado </nota>
```

```
<xs:element name="nota" type="TipoNota" />
```

Expresiones regulares

```

<xs:simpleType name="NIF">
  <xs:restriction base="xs:token">
    <xs:pattern value="\d{7,8}[A-Z]" />
  </xs:restriction>
</xs:simpleType>

```

```
<nif>9394173J</nif>
```

```
<xs:element name="nif" type="NIF" />
```

```
<nif>11079845M</nif>
```

Ejemplos de expresiones regulares

Expresión	Posibles valores
Elemento \d	Elemento 2
a*b	b, ab, aab, aaab, ...
[xyz]b	xb, yb, zb
a?b	b, ab
a+b	ab, aab, aaab, ...
[a-c]x	ax, bx, cx

Expresiones Regulares

[a-c]x	ax, bx, cx
[^0-9]x	Carácter ≠ dígito seguido de x
\Dx	Carácter ≠ dígito seguido de x
(pa){2}rucha	paparucha
.abc	Cualquier carácter (1) seguido de abc
(a b)+x	ax, bx, aax, bbx, abx, bax,...
a{1,3}x	ax, aax, aaax
\n	Salto de línea
\p{Lu}	Letra mayúscula
\p{Sc}	Símbolo de moneda

Tipos Derivados por Extensión

Similar a las subclases de POO: Añadir elementos a un tipo base

```

<xs:complexType name="Figura" >
  <xs:attribute name="color" type="Color"/>
</xs:complexType>

<xs:complexType name="Rectángulo">
  <xs:complexContent>
    <xs:extension base="Figura">
      <xs:attribute name="base" type="xs:float" />
      <xs:attribute name="altura" type="xs:float" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="Círculo">
  ...similar pero incluyendo el radio
</xs:complexType>
  
```

***** Curso sobre Tecnología Web ***** versión 2005

Tipos Derivados por Extensión

Los tipos derivados pueden utilizarse en los mismos sitios que la clase base

```

<xs:element name="figuras">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="figura" type="Figura"
        maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
  
```

```

<figuras>
<figura base="23" altura="3" xsi:type="Rectángulo" />
<figura radio="3" xsi:type="Círculo" />
</figuras>
  
```

Es necesario especificar el tipo mediante **xsi:type**

***** Curso sobre Tecnología Web ***** Versión 2005

Tipos Abstractos

Mediante `abstract="true"` se declara un tipo como abstracto.

Ese tipo no puede usarse directamente

```
<xs:complexType name="Figura" abstract="true">
  <xs:attribute name="color" type="Color"/>
</xs:complexType>
```

También es posible limitar la derivación de tipos

`final="restriction"`

Declaración de Atributos

```
<xs:complexType name="Círculo">
  <xs:attribute name="radio"
    type="xs:float"
    use="required" />

  <xs:attribute name="color"
    type="Color"
    default="255 0 0"/>

  <xs:attribute name="tipo"
    type="xs:string"
    fixed="jpeg" />
</xs:complexType>
```

Por defecto los atributos son opcionales. Indicar que son obligatorios: `use="required"`

Valor por defecto de un atributo. Podría definirse otro valor.

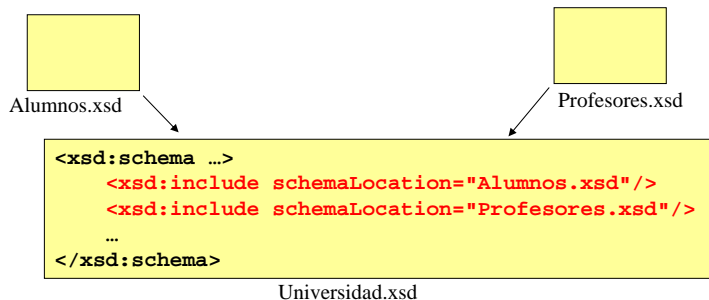
Valor fijo de un atributo. Si no se define, se utiliza ése. Si se define, debe coincidir.

Inclusión de Esquemas

include permite incluir elementos de otros esquemas

Los elementos deben estar en el mismo espacio de nombres

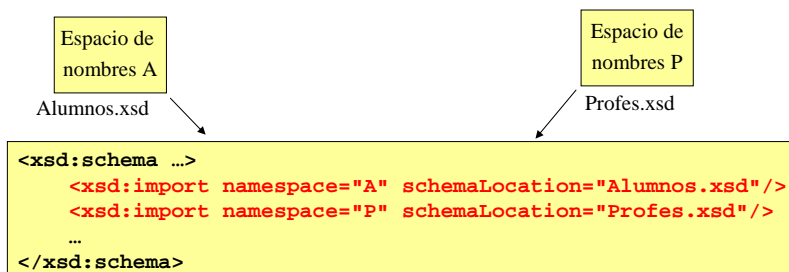
Es como si se hubiesen tecleado todos en un mismo fichero



Universidad.xsd

Importación de Esquemas

import permite incluir elementos de otros esquemas con distintos espacios de nombres



Universidad.xsd

Redefinición de Esquemas

redefine es similar a *include* pero permite modificar los elementos incluidos.

Alumnos.xsd



Añade el elemento nota

```

<xs:redefine schemaLocation="Alumnos.xsd">
  <xs:complexType name="TipoAlumno">
    <xs:complexContent>
      <xs:extension base="TipoAlumno">
        <xs:sequence>
          <xs:element name="nota" type="Nota" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:redefine>
  
```

Claves y Unicidad

Los DTDs proporcionaban el atributo ID para marcar la unicidad (un valor ID era único en todo el documento)

XML Schema tiene más posibilidades:

- Indicar que un elemento es único (**unique**)
- Definir atributos únicos
- Definir combinaciones de elementos y atributos como únicos
- Distinción entre unicidad y claves (**key**)
 - Clave = además de ser único, debe existir y no puede ser nulo.
- Declarar el rango de un documento en el que algo es único

Claves y Unicidad

```

<xs:complexType name="Alumnos">
  <xs:sequence>
    <xs:element name="Alumno" type="TipoAlumno"/>
  </xs:sequence>
  <xs:key name="DNI">
    <xs:selector xpath="a:alumno"/>
    <xs:field xpath="a:dni"/>
  </xs:key>
</xs:complexType>

```

Es necesario incluir el espacio de nombres (XPath)

La clave puede formarse para atributos y elementos

```

<xs:key name="DNI">
  <xs:selector xpath="a:alumno"/>
  <xs:field xpath="a:nombre"/>
  <xs:field xpath="a:nombre"/>
</xs:key>

```

Una clave puede estar formada por varios elementos

Claves y Unicidad

```

<xs:complexType name="Alumnos">
  <xs:sequence>
    <xs:element name="Alumno" type="TipoAlumno"/>
  </xs:sequence>
  <xs:unique name="DNI">
    <xs:selector xpath="a:alumno"/>
    <xs:field xpath="a:dni"/>
  </xs:unique>
</xs:complexType>

```

Unique especifica que debe ser único, pero podría no existir

Referencias a Claves

keyref especifica que debe hacer referencia a una clave (Claves Externas)

```

<xs:element name="clase">
  <xs:sequence>
    <xs:element name="alumnos" ...
    <xs:element name="delegado" ...
  </xs:sequence>

  <xs:key name="DNI">
    <xs:selector xpath="a:alumnos/a:alumno"/>
    <xs:field xpath="a:dni"/>
  </xs:key>

  <xs:keyref name="Delegado" refer="DNI">
    <xs:selector xpath="a:delegado"/>
    <xs:field xpath="a:dni"/>
  </xs:keyref>
    
```

Valores Nulos

Indicar que un elemento puede ser nulo sin estar vacío.

Vacío (Empty): Un elemento sin contenido

Nulo (Nil): Un elemento que indica que no hay valor

```

<xsd:element name="Persona">
  <xsd:complexType>
    <xsd:element name="nombre" type="xsd:NMTOKEN"/>
    <xsd:element name="primerApellido" type="xsd:NMTOKEN"/>
    <xsd:element name="segundoApellido" type="xsd:NMTOKEN"
      nillable="true"/>
  </xsd:complexType>
</xsd:element>
    
```

```

<persona>
  <nombre>John</nombre>
  <primerApellido>Smith</primerApellido>
  <segundoApellido xsi:nil="true"/>
</persona>
    
```

El segundo apellido puede ser un NMTOKEN o estar indefinido

Incluir cualquier contenido...

any indica cualquier contenido de un determinado espacio de nombres
anyAttribute cualquier atributo de un espacio de nombres

```

<xs:complexType name="Comentario">
  <xs:sequence>
    <xs:any namespace="http://www.w3.org/1999/xhtml"
      minOccurs="1"
      processContents="skip" />
    </xs:sequence>
    <xs:anyAttribute
      namespace="http://www.w3.org/1999/xhtml" />
  </xs:complexType>

```

También puede declararse
##any, ##local, ##other

```

<comentarios>
  <html:p>Es un
    <html:emph>Listillo</html:emph>
  </html:p>
</comentarios>

```

Otros valores
strict = obliga a validar
lax = valida si es posible

XML Schema Limitaciones

No soporta entidades. Mecanismo para crear macros

<ENTITY &texto; "Este texto se repite muchas veces" >

Es necesario seguir usando los DTDs ☹

Lenguaje de Restricciones limitado

Ejemplo: ¿Verificar valor total = suma de valores parciales?

Sensibilidad al contexto limitada

Por ejemplo: Especificar que el contenido depende del valor de un atributo

<transporte tipo="coche"> ...</transporte>

<transporte tipo="avión"> ...</transporte>

Tamaño de archivos XML Schema puede ser excesivo

Legibilidad de las especificaciones...XML no siempre es legible

Complejidad de la especificación:

Muchas situaciones/combinaciones excepcionales

Esquemas XML Otras técnicas

Relax NG. Desarrollado por OASIS a partir de TREX y RELAX

- Soporta mayor número de restricciones y gramáticas ambiguas

- Incluye una sintaxis abreviada (no XML)

Schematron

- Utiliza un modelo basado en reglas (en vez de gramáticas)

- Asocia reglas de validación a expresiones XPath

- Puede expresar restricciones arbitrarias

Lenguajes para XML (Encaje de patrones con expresiones Regulares)

- XDuce, CDuce, HydroJ

Diseño Vocabularios XML

Diseño Vocabularios XML

Separación tradicional de dos mundos

Sistemas orientados a Datos

Información uniforme y fuertemente estructurada (ej. Tablas)

Mucha cantidad de información repetida

Objetivo: Procesamiento eficiente (Almacenes de datos)

Sistemas orientados a Documentación

Información poco uniforme y entrelazada (ej. Libros)

No existe un patrón uniforme

Objetivo: Comunicación, Presentación (Mensajes)

Se podría añadir un tercer mundo:

Programación Orientada a Objetos

Propuestas para añadir capacidad de programación a documentos

XML: Información **semi-estructurada** (Lugar intermedio)

Estructuras jerárquicas entrelazadas

Diseño Vocabularios XML

Características a tener en cuenta

Tamaño de documentos

Facilidad de escritura

Facilidad de procesamiento

Flexibilidad (ej. HTML es muy flexible, Bases de Datos = menos)

Consistencia: Evitar características incoherentes

Nivel de abstracción: Buscar término medio en nivel de detalle

`<fecha>10 Marzo 2003</fecha>`

`<fecha><día>10</día><mes>Marzo</mes><año>2003</año></fecha>`

Patrones de diseño:

www.xmlpatterns.com

Ejemplo de Discusión

Representación de propiedades

```
<pizza
  nombre="Margarita"
  precio="6" />
```

```
<pizza>
  <nombre>Margarita </nombre>
  <precio>6</precio>
</pizza>
```

¿Atributos o Elementos?

Razones filosóficas:

Atributos: valores asociados con objetos sin identidad propia (edad)
 Subelementos: valores con identidad propia (fecha-nacimiento)



Orígenes (SGML):

Atributos: meta-información (información sobre el contenido)
 Subelementos: Contenido



Curso sobre Tecnología Web

Versión 2005

Ejemplo de Discusión

Representación de propiedades

```
<pizza
  nombre="Margarita"
  precio="6" />
```

```
<pizza>
  <nombre>Margarita </nombre>
  <precio>6</precio>
</pizza>
```

¿Atributos o Elementos?

En los DTDs

Pueden incluirse restricciones sobre su valor
 Ej. valor "si" o "no"
 Pueden definirse valores por defecto
 Pueden validarse los valores ID e IDREF
 Pueden definirse restricciones sobre espacios en blanco (NMTOKENS)
 Ocupan menos espacio
 Más fáciles de procesar (SAX y DOM)
 Acceso a entidades externas (datos binarios)

Soportan valores arbitrariamente complejos y repetidos
 Establecen un orden
 Soportan *atributos de atributos*
 Mayor flexibilidad ante modificaciones



Curso sobre Tecnología Web

Versión 2005

Diseño Vocabularios En resumen...

...Aparición de una nueva torre de Babel...

Algunos Consejos:

- Estudiar dominio de la Aplicación (ver estándares ya definidos!!!)
- Considerar futuras ampliaciones (extensibilidad)
- Validar antes de que sea tarde
- Usar espacios de nombres
- etc. etc.

Acceso a documentos XML XPath

Introducción a XPath

XPath define acceder a partes de un documento XML

Surgió dentro de XSL pero se generalizó su uso en otros contextos,
por ejemplo: XQuery

Se basa en relaciones de “parentesco” entre nodos

Su estilo de notación es similar a las rutas de los ficheros, pero se
refiere a nodos en un documento XML

Ejemplo: `/alumno/nombre`

Xpath: Términos básicos

Nodo actual (*current node*)

Es un nodo que *está seleccionado* cuando se va a evaluar una expresión
XPath

Constituye el punto de partida al evaluar la expresión

Nodo contexto (*context node*)

Para evaluar una expresión, se van evaluando *subexpresiones* parciales

Cada vez que se evalúa una subexpresión se obtiene un nuevo conjunto
de nodos (*node-set*) que es el nuevo *contexto* para evaluar la siguiente
subexpresión

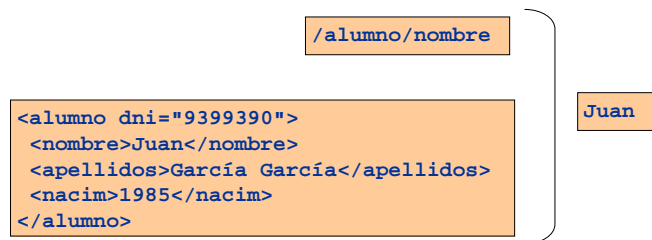
Tamaño del contexto (*context size*)

El número de nodos que se están evaluando en un momento dado en la
expresión XPath

Expresiones XPath

Una expresión XPath arroja (tras ser evaluada) una expresión de 4 tipos posibles: **conjunto de nodos** (**node-set**), **booleano**, **número**, **cadena**

Hay que considerar una expresión XPath como un “predicado”, que devuelve todo lo que encaja con dicho predicado



Expresiones XPath

Tokens válidos en una expresión XPath

Paréntesis y similares: () { } []

Elemento actual (.) elemento padre (..)

Atributo @, elemento *, separador ::, coma (,)

El nombre de un elemento

Tipo de nodo (comment, text, processing instruction, node)

Operadores: `and`, `or`, `mod`, `div`, `*`, `/`, `//`, `|`, `+`, `-`, `=`, `!=`, `<`, `<=`, `>`, `>=`

Nombres de función

Nombre de eje (*axis*): ancestor, ancestor-or-self, attribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, self

Literales, entre comillas dobles o simples (se pueden anidar alternadas)

Números

Referencias a variables (\$nombreVariable)

Node-set

Grupo de nodos (no ordenado) resultado de evaluar una expresión XPath

Los nodos pueden ser de 7 tipos

- Elemento
- Atributo
- Texto
- Espacio de nombres
- Instrucción de procesamiento
- Comentario
- Raíz

Los elementos de un node-set son siempre hermanos

Sus hijos originales no están incluidos, pero se puede acceder a ellos

Se corresponde con la idea intuitiva de “ruta de directorio”

Un *location path* siempre devuelve un **node-set**

Tipos de rutas de localización

Patrones (patterns): sólo permiten el uso de los ejes **child** y **attribute** (se verá después)

Absolutas: parten de la raíz

Relativas: no parten de la raíz (depende del nodo de contexto, **context node**).
Este cambia con cada /, que actúa como separador de los **pasos de localización**. En cada paso se selecciona un nuevo node-set que pasa a ser el nodo de contexto

Pasos de localización

Paso de localización: cada paso de una ruta de localización (separados por /)

Un paso de localización consta de:

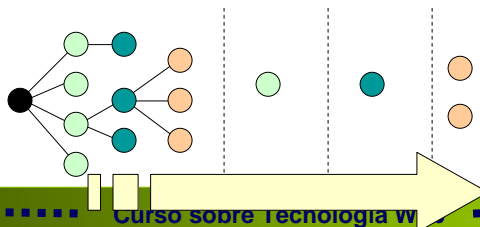
Eje (axis). Es la relación entre el nodo de contexto y el paso

Prueba de nodo (node test). Es el “nombre de directorio”

Predicado (predicate). Expresión XPath entre corchetes.

El eje a veces está implícito (no se pone). El predicado es opcional

eje::pruebanodo[predicado]
/ alumno / nombre / Juan



node test

La forma más simple es escribir simplemente el nombre del nodo (su etiqueta)

También se puede utilizar el asterisco * que simboliza cualquier nombre

Ejemplos:

/universidad/euitio/alumnos/alumno

Encaja con cualquier nodo “alumno” que sea hijo de un nodo “alumnos” que sea hijo de un nodo “euitio” que sea hijo del nodo “universidad” que será el nodo raíz

/universidad/*

Encaja con cualquier nodo que sea hijo del nodo “universidad” que será el nodo raíz

universidad/*

Encaja con cualquier nodo que sea hijo de un nodo “universidad” que sea hijo del nodo de contexto

IMPORTANTE: // indica “que sea hijo de cualquiera”

Ejes (axis)

El eje denota la relación de un paso de localización con su nodo de contexto

Hay una serie de ejes posibles: `ancestor`, `ancestor-or-self`, `attribute`, `child`, `descendant`, `descendant-or-self`, `following`, `following-sibling`, `namespace`, `parent`, `preceding`, `preceding-sibling`, `self`

El eje y la prueba de nodo se separan mediante el operador `::`

Equivale a “**que es un**”, pero sus argumentos se leen de derecha a izquierda

`child` está implícito y casi nunca se pone. Para el nodo raíz, está implícito `self` (self denota al nodo de contexto)

Ejemplos:

`/universidad/eutio`

Equivale de manera implícita a `/self::universidad/child::eutio`

`/universidad/eutio/following-sibling::*`

Todos los nodos que son “hermanos después de” eutio (en el orden del documento) que es hijo de universidad

Predicado

Añade un nivel de verificación al paso de localización

Expresión booleana

Dada la prueba de nodo, y dado el eje, del conjunto de nodos resultante quedan sólo los que cumplan el predicado

En el predicado pueden intervenir funciones XPath
(cuidado con las expresiones, `>` \rightarrow `>`;))

Funciones XPath

Gran variedad de funciones

`boolean()`: convierte a booleano. Aplicada a un conjunto de nodos, devuelve true si no es vacío. `not()`, `true()`

`count()`: Devuelve el número de nodos en un conjunto de nodos

`name()`: Devuelve el nombre de un nodo (su etiqueta). `local-name()`, `namespace-uri()`

`position()`: Devuelve la posición de un nodo en su contexto (empieza en 1) `last()`

Biblioteca de strings. `normalize-space()`, `string()`, `concat()`, `string-length()` `sum()`

Acceso a atributos

Se puede acceder a un elemento atributo gracias al eje attribute::

Contiene todos los nodos atributo del nodo contexto

Una abreviatura de esto es la arroba @

Ejemplo:

`alumno/@dni`

Transformación de XML XSLT y XSL-FO

Hojas de estilos para XML

SGML tenía DSSSL (Document Style Semantics and Specification Language)

Para XML se optó por crear XSL (XML Stylesheet Language)

Posteriormente se dividió en 3 partes:

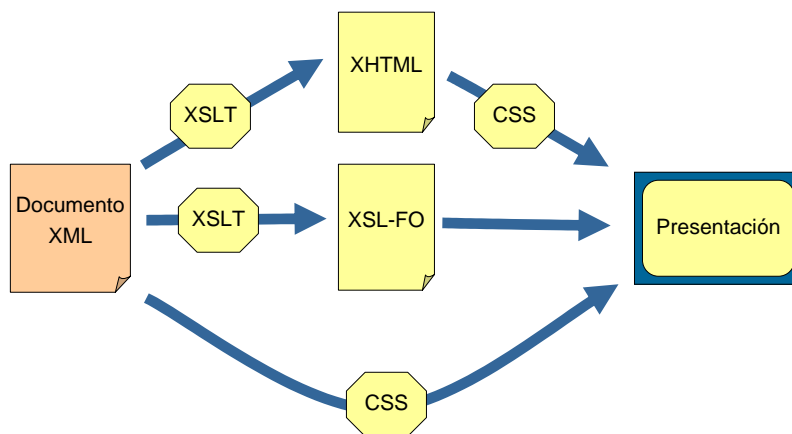
XSLT: Transformación de documentos XML

XPath: Especificar caminos y expresiones XML

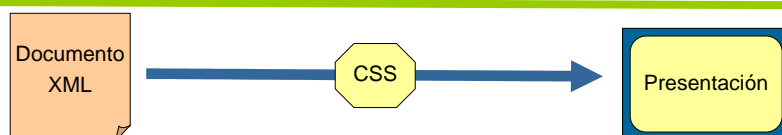
XSL-FO: Objetos de formateo

Además, CSS también puede usarse con XML

Hojas de estilos para XML Posibilidades



Hojas de estilo para XML CSS con XML



mensaje.xml

```

<?xml version="1.0" encoding="iso-8859-1" ?>
<?xml-stylesheet type="text/css"
href="mensaje.css" ?>

<mensaje>
<destino>Juanito el Loco</destino>
<remitente>María la Impaciente</remitente>
<texto>
Recuerda que nos tenemos que ver en la alacena...
</texto>
</mensaje>
  
```

mensaje.css

```

mensaje { border: 5px solid; ... }
destino { color: blue; ... }
remitente { color: red; ... }
texto { margin: 5em; ... }
  
```

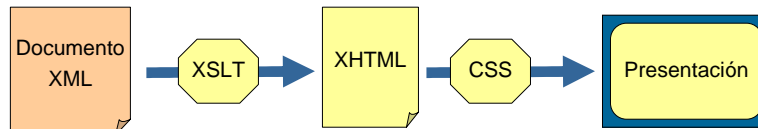


Facilidad
de uso



Posibilidades
limitadas

Hojas de estilo para XML Convirtiendo a HTML



Es la técnica más popular

Permite añadir características de hipertexto e interactivas

Menor calidad para medios impresos

Hojas de estilo para XML XSL

Originalmente, XSL = XSLT + XPath + XSL-FO

Posteriormente, XPath y XSLT toman identidad propia

XSL-FO = Objetos de formateo con propiedades

Muchas propiedades, son comunes con CSS

(2001) XSL-1.0 (Incluía XPath y XSLT)

Incluye modelos de páginas, soporte para internacionalización e hiper-enlaces.

(2003) XSL-1.1

Mayor soporte para marcadores, índices y múltiples flujos

En desarrollo:

XPath 2.0: Añade soporte para tipos de datos y Schemas

XSLT 2.0: Tipos de datos de XPath 2.0 y gestión de errores

Transformación de XML XSLT

XSLT es un lenguaje declarativo (transforma un árbol en otro árbol)
 El programador incluye una serie de reglas de transformación
 El procesador es el que se encarga de obtener el árbol y de escribir el resultado
 Las reglas se basan en la definición de plantillas (*templates*)
 Las plantillas utilizan sintaxis de XPath

```

<xsl:template match="valor a encajar">
  código de salida
</xsl:template>
  
```

Transformación de XML XSLT

```

<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" />

  <xsl:template match="/">
    <html><body><h1>Pizzas del Restaurante Al Capone</h1>
    <xsl:apply-templates />
    </body></html>
  </xsl:template>

  <xsl:template match="pizzas">
    <table><caption>Tipos de Pizzas</caption><tr>
    <xsl:apply-templates />
    </table>
  </xsl:template>

  <xsl:template match="pizza">
    <tr><td><xsl:value-of select="@nombre"/></td>
    <td><xsl:apply-templates /></td>
    <td><xsl:value-of select="@precio" /></td></tr>
  </xsl:template>

  <xsl:template match="ingrediente"><xsl:value-of select="@nombre" />
  </xsl:template>
</xsl:stylesheet>
  
```

Valores que se incluyen en resultado

Patrón de encaje

Referencia a valor de atributo

XSL I Lenguaje de programación

```
<ns>
<num>5</num>
<num>6</num>
<num>7</num>
<num>8</num>
<num>9</num>
<num>10</num>
<num>11</num>
<num>12</num>
<num>100</num>
</ns>
```



Factorial

- 5! = 120
- 6! = 720
- 7! = 5040
- 8! = 40320
- 9! = 362880
- 10! = 3628800
- 11! = 39916800
- 12! = 479001600
- 100! = NaN

```
<xsl:template match="num">
  <li>
    <xsl:value-of select="."/> !=
    <xsl:call-template name="fact">
      <xsl:with-param name="x"><xsl:value-of select="."/>
    </xsl:with-param>
    </xsl:call-template>
  </li>
</xsl:template>

fact x = if x = 0 then 1
         else x * fact (x - 1)

<xsl:template name="fact">
  <xsl:param name="x" />
  <xsl:choose> <xsl:when test="$x = 0">1</xsl:when>
  <xsl:otherwise>
    <xsl:variable name="llamada">
      <xsl:call-template name="fact">
        <xsl:with-param name="x"><xsl:value-of select="$x - 1" />
      </xsl:with-param>
      </xsl:call-template>
    </xsl:variable>
    <xsl:value-of select="$llamada * $x"/>
  </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

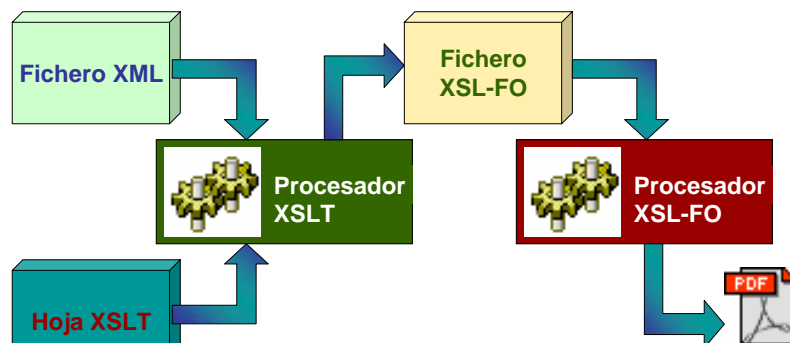
Curso sobre Tecnología Web

Versión 2005

Hojas de estilo para XML XSL-FO

XSL-Formatting Objects

Describe documento en un formato "imprimible" (presentación)



...o un visualizador.

Curso sobre Tecnología Web

Versión 2005

XSL-FO Características

Sintaxis XML

Muchas propiedades compatibles con CSS

Otras posibilidades no contempladas en CSS

- Texto de derecha a izquierda o de arriba a abajo

- Notas al pie

- Notas al margen

- Números de página y referencias cruzadas

...

XSL-FO Ejemplo

```
<?xml version="1.0" encoding="iso-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="PáginaEjemplo">
      <fo:region-body margin="2cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence
    master-reference="PáginaEjemplo">
    <fo:flow flow-name="xsl-region-body">
      <fo:block>Hola en FO</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```



XSL-FO

Estructura de documento

Espacio de nombres: <http://www.w3.org/1999/XSL/Format>

Elemento raíz: **fo:root**

Organización visual del documento:

Uso de modelos de página: "*páginas maestras*"

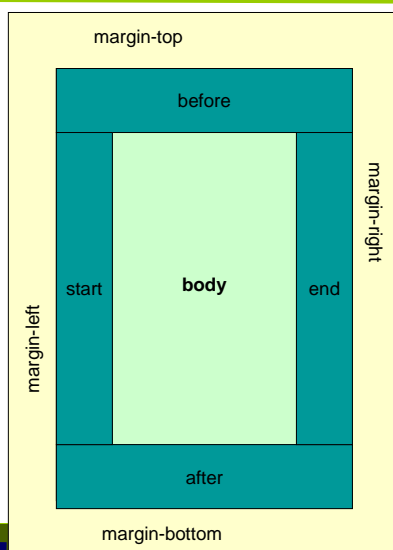
Los modelos se definen dentro de **fo:layout-master-set**

Sólo un tipo de página maestra, **fo:simple-page-master**, que es una página rectangular

Las páginas del documento harán referencia a las páginas maestras que se hayan definido, y heredarán sus propiedades

XSL-FO

fo:simple-page-master



Atributos:

master-name
 page-height
 page-width
 margin-left, margin-top, margin-right, margin-bottom

Ejemplo:

```

<fo:layout-master-set>
  <fo:simple-page-master
    master-name="pagina-normal"
    page-height="29.7cm"
    page-width="21cm"
    margin-left="3cm"
    margin-right="3cm"
    margin-top="2cm"
    margin-bottom="2cm">
  </fo:simple-page-master>
</fo:layout-master-set>
  
```

XSL-FO fo:page-sequence (I)

Crear una secuencia de páginas

Referencia a la página modelo

Atributo **master-reference**

Debe haberse definido en **fo:layout-master-set**

Elementos que incluye como hijos:

fo:title (opcional): Contiene el texto del "título" de la página (puede ser usado por navegadores de Internet como <TITLE>)

fo:static-content: Puede haber muchos o ninguno. Contienen texto que se repite en todas las páginas. También puede contener elementos que se repiten aunque se calculen en cada una (como el número de página)

fo:flow: Un elemento que contiene los contenidos de las páginas, que se irán incluyendo en secuencia

XSL-FO fo:page-sequence (II)

Tiene 8 atributos opcionales para indicar cómo se pagina la secuencia:

initial-page-number

force-page-count

format

letter-value

country

language

grouping-separator

grouping-size

XSL-FO Regiones (I)

Regiones de la página: hijas de `fo:simple-page-master`

Describir tamaños de las regiones

Pueden incluir márgenes

`fo:region-before`

`fo:region-after`

Atributo `extent`

`fo:region-body`

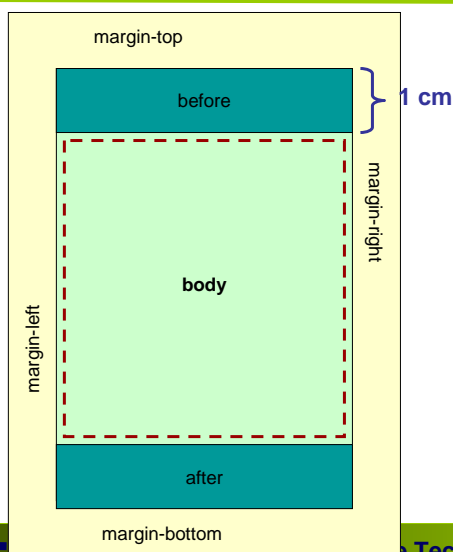
No tiene atributo `extent`

Es todo entre los márgenes de página

Su contenido "pisa" al de las regiones *before*, *after*, *start* y *end*

Pero puede incluir márgenes

XSL-FO Regiones (II)



```

<fo:layout-master-set>
  <fo:simple-page-master
    master-name="pagina-normal"
    page-height="29.7cm"
    page-width="21cm"
    margin-left="3cm"
    margin-right="3cm"
    margin-top="2cm"
    margin-bottom="2cm">
    <fo:region-before extent="1cm"/>
    <fo:region-body margin-top="1cm"
      margin-bottom="1cm"/>
    <fo:region-after extent="1cm"/>
  </fo:simple-page-master>
</fo:layout-master-set>
  
```

XSL-FO

Modelo de formato

Cajas rectangulares, llamadas áreas, que pueden contener:

- Texto
- Espacio vacío
- Imágenes
- Otros objetos de formato

Las cajas tienen bordes y márgenes

Algunos objetos de formato generan más de un área (ej.: en rupturas de páginas o similares)

Elementos a manejar:

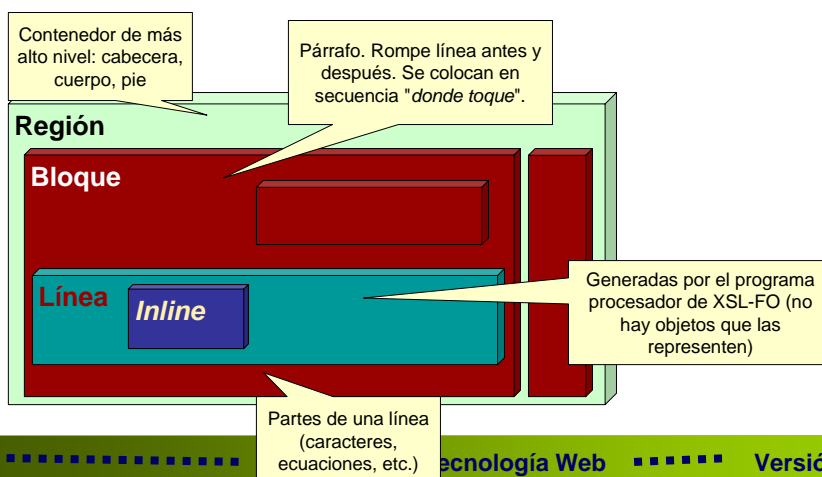
Objetos de formato: Orden y posición del contenido

Propiedades de formato: atributos de los objetos (tipo de letra, tamaño, color, etc.). Se especifican de manera similar a como se hace en CSS.

XSL-FO

Estructura de las áreas

Hay 4 tipos de áreas, representados en el siguiente diagrama



Contiene el "contenido real" de las páginas

Es una secuencia de elementos `fo:block`, `fo:block-container`, `fo:table-and-caption`, `fo:table`, y `fo:list-block`.

Atributo `flow-name`

Indica en qué región de la página va el contenido

Valores posibles:

- `xsl-region-body`
- `xsl-region-before`
- `xsl-region-after`
- `xsl-region-start`
- `xsl-region-end`

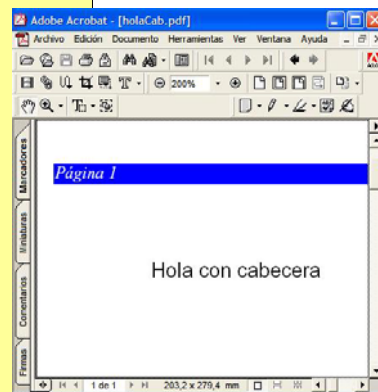
El contenido aparecerá en cada secuencia de página

```

<fo:layout-master-set>
  <fo:simple-page-master master-name="miPágina"
                        margin="2cm">
    <fo:region-body margin="2cm" />
    <fo:region-before extent="2cm" />
  </fo:simple-page-master>
</fo:layout-master-set>

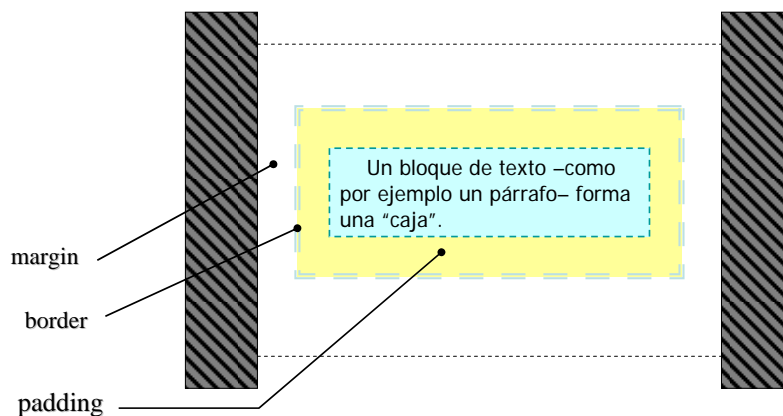
<fo:page-sequence master-reference="miPágina">
  <fo:static-content flow-name="xsl-region-before"
                    font-style="italic" font-size="10pt" font-family="Times">
    <fo:block text-align="start"
              background-color="blue" color="white">
      Página <fo:page-number/>
    </fo:block>
  </fo:static-content>

  <fo:flow flow-name="xsl-region-body">
    <fo:block>Hola con cabecera</fo:block>
  </fo:flow>
</fo:page-sequence>
</fo:root>
  
```



XSL-FO

Modelo de cajas



Cada elemento puede especificarse por separado: top, bottom, left, right
También puede especificarse la altura (height) y la anchura (width)

XSL-FO

Texto

Atributos (Muchos coinciden con los de CSS)

font: **font-style** (italic, normal, backslant, oblique), **font-variant** (normal, small-caps), **font-size** (tamaño absoluto o relativo), **font-family** (serif, sans,...), **font-stretch** (narrow, wide,...), **font-weight** (bold, normal)

text-indent: Sangrado (número), **last-line-end-indent**: Sangrado de última línea

text-align: Justificación (start, end, center, justify, etc.), **text-align-last**: Justificación de la última línea

line-height: Altura de la línea (puede ser porcentaje)

hyphenate (Guionado), language, country, hyphenation-character, etc.

space-before, **space-after**


```

<fo:list-block>
  <fo:list-item>
    <fo:list-item-label end-indent="label-end()">
      <fo:block>&#x2022;</fo:block>
    </fo:list-item-label>
    <fo:list-item-body start-indent="body-start()">
      <fo:block>Primer elemento</fo:block>
    </fo:list-item-body>
  </fo:list-item>
  <fo:list-item>
    <fo:list-item-label end-indent="label-end()">
      <fo:block>&#x2022;</fo:block>
    </fo:list-item-label>
    <fo:list-item-body start-indent="body-start()">
      <fo:block>Segundo elemento</fo:block>
    </fo:list-item-body>
  </fo:list-item>
</fo:list-block>

```

- Primer elemento
- Segundo elemento

```

<fo:table border="0.5pt solid"
  text-align="center"
  table-layout="fixed">
  <fo:table-column column-width="5cm"/>
  <fo:table-column column-width="5cm"/>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell padding="6pt" border="0.5pt solid">
        <fo:block> Uno . Uno </fo:block>
      </fo:table-cell>
      <fo:table-cell padding="6pt" border="0.5pt solid">
        <fo:block> Uno . Dos </fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
      <fo:table-cell padding="6pt" border="0.5pt solid">
        <fo:block> Dos . Uno </fo:block>
      </fo:table-cell>
      <fo:table-cell padding="6pt"
        border="0.5pt solid">
        <fo:block> Dos . Dos </fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-body>
</fo:table>

```

Uno . Uno	Uno . Dos
Dos . Uno	Dos . Dos

XSL-FO Enlaces e imágenes

Enlaces:

```
<fo:basic-link external-destination="..." />  
<fo:basic-link internal-destination="valorID" />
```

Cada bloque puede tener un atributo ID

Referencias a páginas:

```
<fo:page-number-citation ref-id="valorID" />
```

Gráficos

```
<fo:external-graphic src="url('smile.gif')"  
content-height="1em" content-width="1em"/>
```

XSL-FO Otras características

Notas a pie de página: [footnote](#)

Marcadores ([marker](#), [retrieve-marker](#))

Ejemplo, incluir nombre del capítulo en la cabecera

Líneas guía: [fo:leader](#)

Elementos [fo:float](#), sin posición absoluta

Orientación de escritura: [reference-orientation](#)

Bidireccionalidad (izqda a dcha y dcha a izqda)

Consultas XML: XQuery

XQuery

XQuery es el lenguaje de consultas de documentos XML
XQuery 1.0 es todavía un borrador desarrollado por el W3c
Inspirado: lenguaje de consultas de bases de datos (SQL)
Fuertemente tipado mediante XML Schema

Enunciados “FLWOR” :

- for** {iterador que enlaza variables}
- let** {colecciones}
- where** {condiciones}
- order by** {condiciones de orden}
- return** {generador de la salida}

Ejemplo de XQuery

Entrada (alumnos.xml)

```
<alumnos>
  <alumno dni="93940">
    <nombre>Jose</nombre>
    <apells>Bernardo</apells>
    <nota>7</nota>
  </alumno>
  ....
</alumnos>
```

Consulta (XQuery)

```
for $a in document("alumnos.xml")//alumno
where $a/nombre = "Jose"
and $a/apells = "Bernardo"
return <alumno>{
  $a/@dni,
  $a/nota
}</alumno>
```

Resultado

```
<alumno dni="93940">
  <nota>7</nota>
</alumno>
```

Programación XML

Programación XML

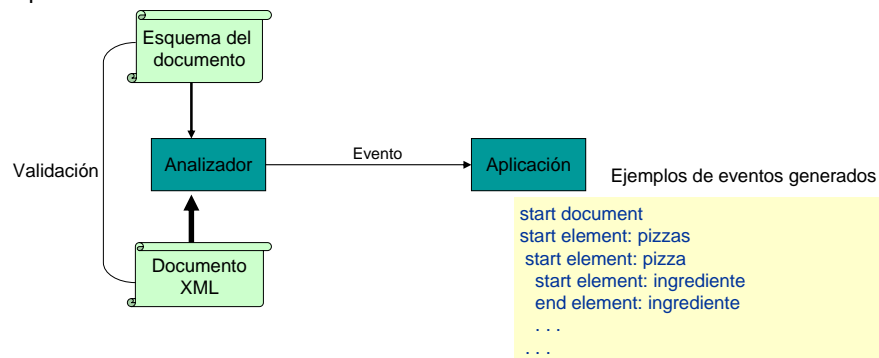
Principales técnicas:

- Guiada por eventos: SAX
- Basada en el recorrido del árbol: DOM
- APIs Específicos para Java: JDOM
- Pull-parsing (evaluación perezosa, XPP)
- Enlaces a esquemas (*Bindings*): JAXB

SAX: Modelos de eventos

SAX (Simple API for XML) es un analizador basado en eventos

El programador se encarga de tratar los eventos que se producen al procesar el documento



SAX: API simple para XML

Creada por desarrolladores XML a partir de los comentarios en la lista xml-dev en 1998

Funcionamiento:

Durante el reconocimiento del documento...

- ...cada vez que se identifica una estructura (elemento) se mira si hay un procedimiento que manipula ese elemento
 - se llama al procedimiento
 - cuando termina, continua con el reconocimiento

Los procedimientos permiten procesar el documento guiado por eventos

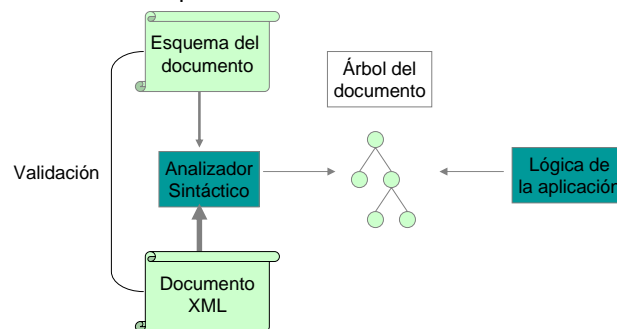
Los eventos son las ocurrencias de los elementos a los que hacen referencia.

Los datos son los elementos asociados con el elemento identificado

DOM (Document Object Model)

Se genera un árbol a partir del documento XML

DOM = Interfaz que define cómo recorrer/modificar dicho árbol



Problemas: Ficheros grandes

Solución: Deferred document (evaluación perezosa)

DOM: Modelo de Objetos de Documento

Origen en el Consorcio World Wide Web (W3C)

Es una interfaz independiente de la plataforma y el lenguaje de programación que permite acceder y manipular dinámicamente el contenido, el estilo y la estructura de un documento.

La norma se define en niveles en lugar de versiones:

Nivel 1: describe la funcionalidad y el modo de navegar por un modelo general

Nivel 2: se centra en tipos de documentos específicos (XML, HTML, CSS)

Nivel 3: más facilidades en tipos de documentos específicos (validación para XML)

API específica para Java que ofrece un modelo de objetos similar a DOM para manipular ficheros XML

A diferencia de DOM, utiliza clases y colecciones de Java

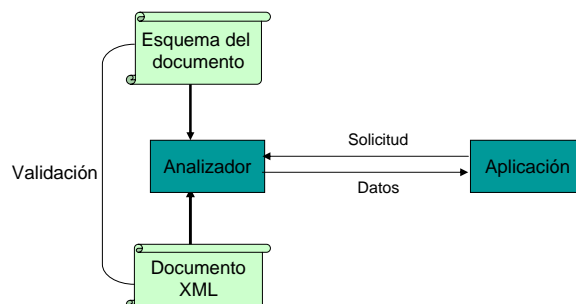
Puede integrarse con SAX y DOM

No contiene analizador propio

Objetivo: Facilitar procesamiento de XML

Pull Parsing Análisis bajo demanda

En Pull parsing, el control del análisis es realizado por la aplicación
 Se diferencia de SAX, que podría considerarse *Push parsing* (el analizador envía los eventos a la aplicación)
 Ejemplo de API: StAX (Streaming API for XML)



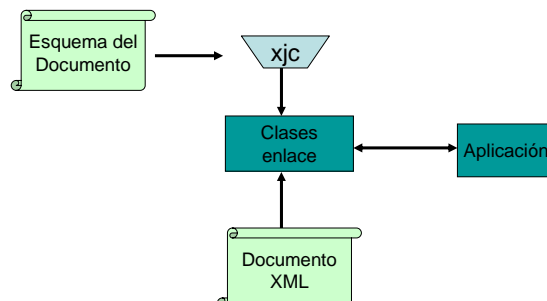
JAXB: Enlaces

JAXB: Java API for XML Binding

A partir del esquema del documento se generan clases de enlace

Permite trabajar con objetos de la aplicación

Generación automática de documentos válidos



Aplicaciones XML

XHTML, MathML, SVG, SMIL, X3D, WML

XHTML+CSS Evolución

(1999) XHTML 1.0 consiste en adaptar HTML para que sea un vocabulario XML

Principales diferencias:

Todas las etiquetas deben cerrarse (XML bien formado)

Los nombres de elementos deben ser minúsculas

Nuevas versiones:

(2000) XHTML Basic = Subconjunto para pequeños dispositivos

(2000) Modularización de XHTML: XHTML 1.1

(2003) XML Events. Gestión de eventos

En desarrollo:

XHTML 2.0, XFrames, XML Print, HLink...


XHTML+CSS Módulos

Structure: body, head, html, title
Text: abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var
Hypertext: a
List: dl, dt, dd, ol, ul, li
Object: object, param
Presentation: b, big, hr, i, small, sub, sup, tt
Edit: del, ins
Bidirectional Text: bdo
Forms: button, fieldset, form, input, label, legend, select, optgroup, option, textarea

Table: caption, col, colgroup, table, tbody, td, tfoot, th, thead, tr
Image: img
Client-side Image Map: area, map
Server-side Image Map: Atributo *ismap* de *img*
Intrinsic Events: Atributos de sucesos
Metainformation: meta
Scripting: noscript, script
Stylesheet : elemento style
Style Attribute : atributo style
Link : link
Base: base

XHTML+CSS Evolución de CSS

(1994) Primer borrador de CSS
 Objetivo: Permitir combinar preferencias visuales del autor y del usuario (*en cascada*)
 (1996) CSS nivel 1: Propiedades de fuentes, márgenes, colores, etc.
 (1998) CSS nivel 2: Añade posiciones absolutas, páginas, numeración automática, etc.
 En desarrollo CSS 3, añadirá selectores, texto vertical, interacción, etc.
 Otros perfiles de CSS para móviles, TV e impresión
 CSS no tiene sintaxis XML



MTI
INTERNACIONAL
MAGISTER EN TECNOLOGÍAS
DE LA INFORMACIÓN

XHTML+CSS: Ejemplo

AlCapone.html

```

<html>
<head>
<title>Pizzeria Al Capone</title>
<link rel="stylesheet" href="pizzeria.css">
</head>
<body>
<h1>Pizzería Al Capone</h1>
<p>Lista de enlaces</p>
<ul>
<li><a href="Pizzas.html">
Tipos de Pizzas</a></li>
<li><a href="http://www.mafia.it">
Patrocinadores</a></li>
<li><a href="#Contacto">
Contacto</a></li>
</ul>

<h2><a name="Contacto">Contacto</a></h2>
<p><span class="item">Dirección:</span>
C/ Génova Nº 3, Oviedo, España</p>
<p><span class="item">Teléfono:</span>
985203040</p>
</body>
</html>

```

Enlace a hoja de estilo → `<link rel="stylesheet" href="pizzeria.css">`

Sin aspectos visuales → `<h1>Pizzería Al Capone</h1>`

Identificación elementos → ``


pizzeria.css

```


body { color : yellow;
background: blue
}
a:link { color: red }
a:visited { color: white }
span.item { color : red }

```

Página visualizada



..... Curso sobre Tecnología Web
Versión 2005



MTI
INTERNACIONAL
MAGISTER EN TECNOLOGÍAS
DE LA INFORMACIÓN

XHTML+CSS: Ejemplo

pizzas.html


```

<html>
<head>
<title>Tipos Pizzas</title>
<link rel="stylesheet" href="pizzeria.css">
</head>
<body>
<h1>Pizzas del Restaurante Al Capone</h1>
<table><caption>Tipos de Pizzas</caption>
<thead>
<tr><th>Pizza</th><th>Ingredientes</th><th>Precio</th></tr>
</thead>
<tbody>
<tr><td>Barbacoa</td><td>Salsa barbacoa, Mozzarella, Pollo, Bacon, Ternera</td><td>8 &euro;</td></tr>
<tr><td>Margarita</td><td>Tomate, Jamón, Queso</td><td>6 &euro;</td></tr>
</tbody>
</table>
</body>
</html>

```

Referencia a la misma hoja de estilos → `<link rel="stylesheet" href="pizzeria.css">`

Misma apariencia → ``



..... Curso s

HTML carece de facilidades para incorporar fórmulas matemáticas

Se recurría a incluirlas como imágenes

Múltiples problemas:

- Fórmula como algo indivisible

- No es posible adaptar a diferentes formatos visuales

- Procesamiento de fórmulas: buscadores, índices, reutilización, etc

(1999) MathML 1.0

(2001) MathML 2.0: nuevos elementos y DOM

Material matemático a todos los niveles

Codificar tanto notaciones como significados

Facilitar conversión con otros formatos

Facilitar la visualización de expresiones complejas

Permitir la extensibilidad

Legible por personas...por ser XML pero...

NO está pensado para edición manual de fórmulas

2 estilos

Presentación: Estructura visual en 2 dimensiones

Contenido: Significado de las fórmulas

Presentación

```
<math>
<mrow>
  <msup>
    <mfenced>
      <mrow>
        <mi>a</mi>
        <mo>+</mo>
        <mi>b</mi>
      </mrow>
    </mfenced>
    <mn>2</mn>
  </msup>
</mrow>
</math>
```

$$(a+b)^2$$

Contenido

```
<math>
<apply>
  <power/>
  <apply>
    <plus/>
    <ci>a</ci>
    <ci>b</ci>
  </apply>
  <cn>2</cn>
</apply>
</math>
```

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
    <mi>x</mi><mo>=</mo>
    <mfrac>
      <mrow><mrow><mo>-</mo><mi>b</mi></mrow>
        <mo>±</mo>
      <msqrt><mrow><msup><mi>b</mi><mn>2</mn></msup>
        <mo>-</mo>
        <mrow><mn>4</mn></mrow><mo>&InvisibleTimes;</mo>
        <mi>a</mi><mo>&InvisibleTimes;</mo><mi>c</mi></mrow>
      </msqrt>
    </mfrac>
    <mrow><mn>2</mn></mrow><mo>&InvisibleTimes;</mo><mi>a</mi></mrow>
  </math>
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

MathML

Elementos Presentación

mi: Identificador	msub: Subíndice
mn: Número	msup: Superíndice
mo: Operador	msubsup: Subíndice y superíndice
mtext: Texto	munder: Escribir bajo un elemento
mspace: Espacio	mover: Escribir sobre un elemento
ms: String literal	munderover: Escribir bajo y sobre un elemento
mglyph: Carácter no estándar	mmultiscripts: Múltiples índices
mrow: Grupo horizontal	mtable: Tabla o array
mfrac: Fracción	mtr: Fila de tabla
msqrt: Raíz cuadrada	mttd: Datos de tabla
mroot: Raíz n-ésima	malaligngroup: Grupo de alineación
mstyle: Cambiar estilo	: Escribir sobre un elemento
merror: Introducir un error	malignmark: Marca un punto de alineación
mpadded: Ajustar espacio	mlabeledtr: Fila en tabla con etiqueta
mphantom: Contenido invisible	maction: Enlaza una acción
mfenced: Rodear contenido de paréntesis	
menclose: Símbolo de división	

MathML

Elementos de Contenido

ci: Identificador de contenido	quotient	int: Integral
cn: Número de contenido	exp:	diff: Derivada
csymbol: Símbolo	factorial	partialDiff: Der. parcial
apply: Aplicación de función a argumento	divide	sum: sumatorio
reln: Relación	max	product:
fn: Función	min	exp
interval: Intervalo	minus	ln
inverse: Inverso	plus	log
sep: Separador	power	logbase
condition: Condición	rem	sin
declare: Declaración	times	cos
lambda: Definición de función	root	tan
compose: Componer funciones	implies	sinh
...	forall	cosh
	exists	arcsin

MathML

Software Existente

Visualizadores:

Internet Explorer + MathPlayer (soporte mediante objeto externo)
Amaya: Visualizador estándar del W3C
Mozilla/Netscape 7: Soporte nativo

Editores:

Amaya
OpenOffice
MathType
...otros...

Conversores: LaTeX - MathML

SVG

Evolución

(2001) SVG 1.0 - Scalable Vector Graphics

(2003) SVG 1.1 Modularización

...actualmente: SVG 1.2 en desarrollo

Objetivos

Gráficos vectoriales: Precisión, escalabilidad, etc.
Compatibilidad con XML y vocabularios de la Web: CSS, Espacios de nombres, XLink, SMIL, ECMAScript, etc.
También permite incluir texto, imágenes *raster* e hiper-enlaces
Formato de texto (no binario): Facilita indexación, búsquedas, etc.

Buena acogida

Soportado en principales navegadores: IE, Mozilla, Amaya, etc.
Planes para incorporación en pequeños dispositivos

SVG

Formato Vectorial

Formato *raster*

Arrays de pixels. Al hacer zoom se pierde calidad



SVG = Formato vectorial

Al hacer zoom no se pierde calidad



SVG

Ejemplo

```

<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">

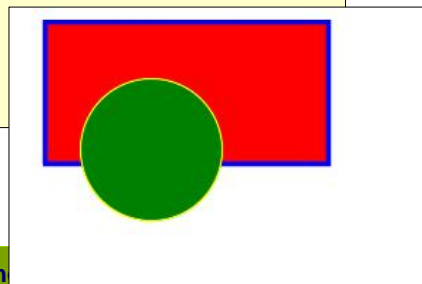
<svg width="300" height="200"
  xmlns="http://www.w3.org/2000/svg">

  <rect x="25" y="10" width="200" height="100"
    fill="red" stroke="blue" stroke-width="3" />

  <circle cx="100" cy="100" r="50"
    fill="green" stroke="yellow"/>

</svg>

```



SVG

Inclusión en HTML

```

<?xml version="1.0" encoding="iso-8859-1"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Ejemplo de Página con HTML</title></head>
<body>
<p>Esto es un párrafo con una imagen...</p>

<object width="300" height="200"
        data="simple.svg" type="image/svg+xml">

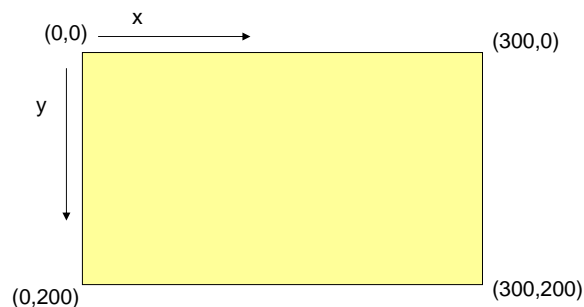
</object>

</body>
</html>
    
```

Para los visualizadores
que no admiten SVG

Espacio de trabajo y coordenadas

width, height = Especifican el ancho y alto de la imagen
viewBox = "x y alto ancho" Indica rectángulo de imagen que va a verse



SVG

Formas básicas

```

<circle
  cx="70" cy="100" r="50" />

<rect
  x="150" y="50" rx="20" ry="20"
  width="135" height="100" />

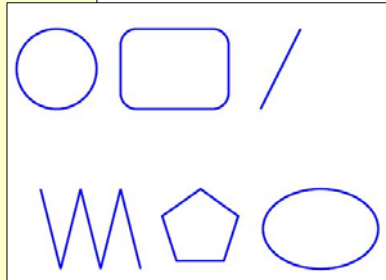
<line x1="325" y1="150" x2="375" y2="50" />

<polyline
  points="50, 250 75, 350 100, 250
         125, 350 150, 250 175, 350" />

<polygon
  points="250,250 297, 284 279,
         340 220, 340 202, 284" />

<ellipse
  cx="400" cy="300" rx="72" ry="50"/>

```



SVG

Path

path permite definir formas gráficas abiertas y cerradas a partir de líneas y curvas

Las formas básicas son casos particulares de path

```

<path d="M 16 9
        L 96 9 L 59 46 L 59 94 C 59 94 57 100 84 102
        L 84 103 L 31 103 L 31 102 C 56 100 54 94 54 94
        L 54 46 L 16 9 z " />

```



M x y = Mover al punto (x,y)
 L x y = Pintar una línea hasta el punto (x,y)
 C x1 y1 x2 y2 x3 y3 = Pintar una curva *bezier cúbica* con los puntos de control (x1,y1) (x2,y2) hasta (x3,y3)
 H n = Pintar horizontalmente hasta n
 V n = Pintar verticalmente hasta n
 Z = Cerrar la forma
 Otros...Q,S, T, A

text define un elemento de texto

tspan permite asignar propiedades a porciones de texto

Muchas propiedades compartidas con CSS

```

<text x="220" y="20">
  <tspan x="220" dy="30">
    Texto en varias líneas</tspan>
  <tspan x="220" dy="30">
    Siguiendo línea</tspan>
  <tspan x="220" dy="30"
    style="font-weight:bold;
    fill:green;stroke:blue">
    Con diferentes propiedades</tspan>
  <tspan x="220" dy="30"
    rotate="10 -10
    10 -10
    10 -10 10">
    incluso rotaciones</tspan>
</text>
  
```

Texto en varias líneas

Siguiendo línea

Con diferentes propiedades

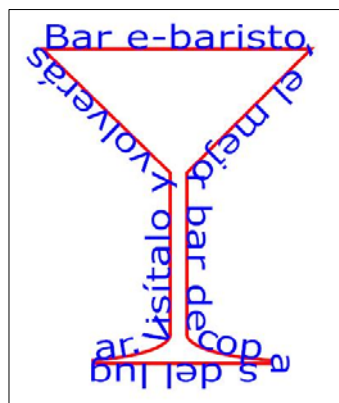
incluso rotaciones

Es posible indicar que el texto se escriba sobre un camino

```

<path id="copa"
  d="M 16 9 . . . L 16 9 z "
/>

<text font-size="10" fill="blue">
  <textPath xlink:href="#copa">
    Bar e-baristo, el mejor bar de
    copas del lugar.
    Visítalo y volverás
  </textPath>
</text>
  
```



SVG

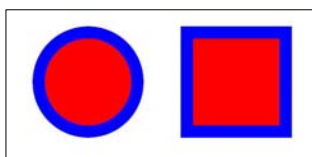
Agrupaciones

g define un grupo

Los elementos dentro de la agrupación pueden compartir propiedades

```

<g fill="red" stroke="blue" stroke-width="5">
  <circle cx="40" cy="30" r="20" />
  <rect x="80" y="10" height="40" width="40" />
</g>
    
```



SVG

Hiper-enlaces

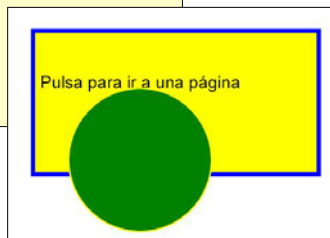
a define un hiper-enlace (similar a HTML)

El atributo es: **xlink:href**

Está previsto usar el vocabulario XLink que amplía las posibilidades de enlace entre documentos XML

```

<a xlink:href="pagina.html">
  <rect x="25" y="10" width="200" height="100"
    fill="yellow" stroke="blue" />
  <text x="30" y="50" text-anchor="start">
    Pulsa para ir a una página</text>
</a>
<circle cx="100" cy="100" r="50"
  fill="green" stroke="yellow" />
    
```



use incluye un elemento definido en la sección **defs**

SVG Transformaciones

translate(x,y) traslada el origen al punto (x,y)

scale(sx,sy) escala sobre sx y sy

rotate(a) rota un ángulo a

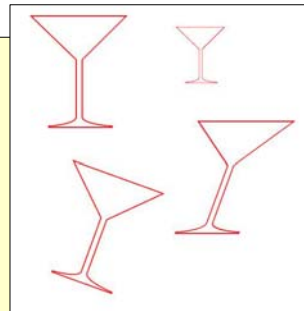
matrix(a,b,c,d,e,f) matriz de transformación

skewX(n) inclinación de n grados horizontal

skewY(n) inclinación de n grados vertical

```

<use transform="translate(10,10)"
      xlink:href="#copa" />
<use x="300" y="50"
      transform="scale(0.5,0.5)"
      xlink:href="#copa" />
<use x="100" y="100"
      transform="rotate(20)"
      xlink:href="#copa" />
<use x="200" y="100"
      transform="skewX(-20)"
      xlink:href="#copa" />
  
```



..... Versión 2005

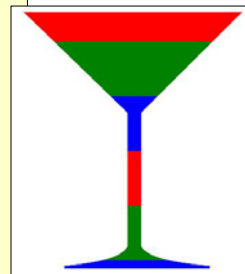
SVG Cortes (clipping)

clipPath(x,y) define un camino de corte

```

<clipPath id="corte">
  <path
    d="M 16 9 ... 16 9 z "
  />
</clipPath>

<g style="stroke:none;
      clip-path:url(#corte)">
  <rect style="fill:red"
    x="0" y="0" width="500" height="20"/>
  <rect style="fill:green"
    x="0" y="20" width="500" height="20"/>
  ...
  <rect style="fill:blue"
    x="0" y="240" width="500" height="20"/>
</g>
  
```



..... Curso sobre Tecnología Web Versión 2005

SVG

Relleno y gradientes

El atributo **fill** indica cómo rellenar una forma cerrada

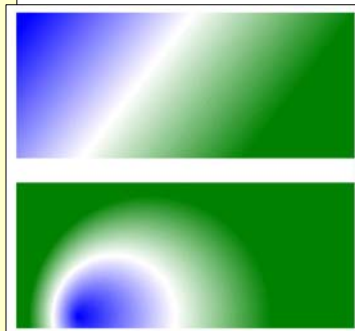
Se puede especificar un color mediante rgb

Se pueden definir gradientes mediante **linearGradient** y **radialGradient**

```

<defs>
<linearGradient id="grad1"
  gradientUnits="userSpaceOnUse"
  x1="30" y1="20" x2="200" y2="150">
  <stop offset="0" style="stop-color:blue" />
  <stop offset="0.5" style="stop-color:white" />
  <stop offset="1" style="stop-color:green" />
</linearGradient>

<radialGradient id="grad2"
  gradientUnits="userSpaceOnUse"
  cx="130" cy="270" r="100" fx="70" fy="270">
  <stop offset="0" style="stop-color:blue" />
  <stop offset="0.5" style="stop-color:white" />
  <stop offset="1" style="stop-color:green" />
</radialGradient>
</defs>
<rect ... style="fill:url(#grad1)"/>
<rect ... style="fill:url(#grad2)"/>
  
```



SVG

Trasparencia y Opacidad

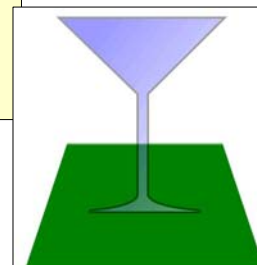
El atributo **opacity** indica nivel de transparencia

(1=no transparente, 0=transparente)

```

<path id="mesa" fill="green"
  d="M 20 70 L 95 70 L 115 130
    L 0 130 L 20 70 z" />

<path id="copa" fill="url(#grad1)"
  opacity="0.4" stroke="black"
  d="M 16 9 . . . L 16 9 z" />
  
```



SVG Efectos de Filtros

```

<filter id="filtro"
  filterUnits="userSpaceOnUse"
  x="0" y="0" width="500" height="500">
  <feGaussianBlur in="SourceAlpha"
    stdDeviation="10" result="sombra"/>
  <feOffset in="sombra" dx="10" dy="10"
    result="sombraDesp"/>
  <feSpecularLighting in="sombra"
    surfaceScale="5" specularConstant=".75"
    specularExponent="40" lighting-color="gray"
    result="brillo"/>
  <fePointLight x="-5000" y="-10000" z="20000"/>
</feSpecularLighting>
<feComposite in="brillo"
  in2="SourceAlpha" operator="in"
  result="brillo"/>
<feComposite in="SourceGraphic"
  in2="brillo" operator="arithmetic"
  k1="0" k2="1" k3="1" k4="0" result="objeto"/>
<feMerge>
  <feMergeNode in="sombraDesp"/><feMergeNode in="objeto"/>
</feMerge>
</filter>
<use xlink:href="#copa" filter="url(#filtro)"/>
  
```



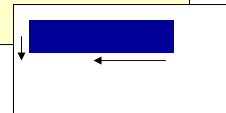
ón 2005

SVG Animaciones

Es posible modificar los valores de los atributos de forma dinámica

```

<rect x="20" y="10" width="120" height="40" fill="blue">
  <animate attributeName="width"
    from="120" to="20" begin="0s" dur="4s" fill="freeze"/>
  <animate attributeName="height"
    from="40" to="100" begin="2s" dur="6s" fill="freeze"/>
</rect>
  
```



animate anima atributos generales

set modifica valor de atributos discretos (ejemplo, *visibility*)

animateMotion movimiento de un objeto a lo largo de un camino

animateColor cambia los colores

animateTransform anima una transformación

Atributos de animación

begin = Comienzo de la animación

Puede ser un tiempo (2s) o valor relativo a otra animación

end = Fin de la animación

dur = Duración de la animación

repeatCount = N° de repeticiones (indefinite = infinitas)

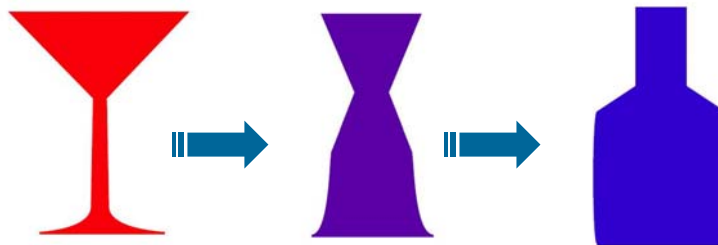
repeatDur = Duración de la repetición

min/max = Duración mínima/máxima

restart = Indica si es posible volver a activar la animación

fill = Indica estado al finalizar la animación (freeze|remove)

```
<path fill="red">
  <animate attributeName="d"
    from="M 16 9 . . . L 16 9 z "
    to="M 45 9 . . . L 45 9 z"
    fill="freeze" dur="5s" id="a1" begin="a2.end" />
  <animate attributeName="fill" from="red" to="blue" dur="5s"
    fill="freeze" repeatCount="indefinite"/>
</path>
```



```
<script type="text/ecmascript">
function addrect(evt) {
  var svgobj = evt.target;
  var svgdoc = svgobj.getOwnerDocument();
  var newnode = svgobj.cloneNode(false);
  svgstyle = newnode.getStyle();
  var x = 10 + 480 * Math.random();
  var y = 10 + 330 * Math.random();
  var width=10+100*Math.random();
  var height=10+50*Math.random();
  newnode.setAttribute('x',x);
  newnode.setAttribute('y',y);
  newnode.setAttribute('width',width);
  newnode.setAttribute('height',height);
  svgstyle.setProperty('opacity',0.3+0.7*Math.random());
  var contents = svgdoc.getElementById('contents');
  newnode = contents.appendChild(newnode);
}]]&gt;&lt;/script&gt;

&lt;g id="contents"&gt;
&lt;rect onclick="addrect(evt)" style="fill:red;opacity:1"
  x="150" y="100" width="20" height="20"/&gt;
&lt;rect onclick="addrect(evt)" style="fill:green;opacity:1"
  x="250" y="100" width="20" height="20"/&gt;
&lt;rect onclick="addrect(evt)" style="fill:blue;opacity:1"
  x="350" y="100" width="20" height="20"/&gt;
&lt;/g&gt;</pre>
</div>
<div data-bbox="516 187 770 306" data-label="Image">
<img alt="A collection of small, semi-transparent colored rectangles (red, green, blue) scattered on a white background, representing the output of the SVG DOM script." data-bbox="516 187 770 306"/>
</div>
<div data-bbox="635 426 782 440" data-label="Text">
<p>■ ■ ■ Versión 2005</p>
</div>
<div data-bbox="209 554 306 596" data-label="Page-Header">
<img alt="MTI INTERNACIONAL logo" data-bbox="209 554 306 596"/>
</div>
<div data-bbox="602 546 787 596" data-label="Page-Header">
<h1>SMIL<br/>Evolución</h1>
</div>
<div data-bbox="228 622 644 638" data-label="Text">
<p>SMIL = Synchronized Multimedia Integration Language</p>
</div>
<div data-bbox="259 639 605 654" data-label="Text">
<p>SMIL es a multimedia lo que HTML es a hipertexto</p>
</div>
<div data-bbox="259 654 734 682" data-label="Text">
<p>Objetivo: Integrar/sincronizar elementos de diferentes medios: vídeos, imágenes, sonidos, etc.</p>
</div>
<div data-bbox="228 684 356 699" data-label="Text">
<p>(1998) SMIL 1.0</p>
</div>
<div data-bbox="228 700 598 716" data-label="Text">
<p>(2001) SMIL 2.0 Creación de diferentes módulos</p>
</div>
<div data-bbox="259 717 504 732" data-label="Text">
<p>Combinación en otras aplicaciones:</p>
</div>
<div data-bbox="259 732 361 746" data-label="Text">
<p>XHTML+SMIL</p>
</div>
<div data-bbox="259 748 671 763" data-label="Text">
<p>SVG+SMIL: Las animaciones de SVG forman parte de SMIL</p>
</div>
<div data-bbox="259 764 297 777" data-label="Text">
<p>etc...</p>
</div>
<div data-bbox="212 842 782 857" data-label="Page-Footer">
<p>■ ■ ■ ■ ■ ■ ■ ■ ■ ■ Curso sobre Tecnología Web ■ ■ ■ ■ ■ Versión 2005</p>
</div>
<div data-bbox="925 953 958 971" data-label="Page-Footer">
<p>90</p>
</div>
```

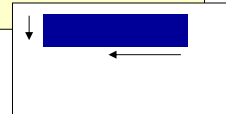
SMIL Animaciones en SVG

SMIL puede combinarse con SVG para realizar animaciones

Ejemplo, modificar valores de atributos

```

<rect x="20" y="10" width="120" height="40" fill="blue">
  <animate attributeName="width"
    from="120" to="20" begin="0s" dur="4s" fill="freeze"/>
  <animate attributeName="height"
    from="40" to="100" begin="2s" dur="6s" fill="freeze"/>
</rect>
    
```



SMIL también puede combinarse con XHTML o utilizarse en documentos independientes

X3D Evolución

Definir escenas de realidad virtual en Internet

Adaptación de VRML (Virtual Reality Modeling Language) a XML

Evolución:

(1994) Posibilidad de desarrollar un estándar para realidad virtual en Internet

Aparecen VRML 1.0 y VRML 2.0

(1997) VRML 97 = Estándar ISO Internacional

Objetivos: Independencia de plataforma, extensibilidad, bajo ancho de banda

(1999) Cambio nombre de Consorcio VRML a consorcio Web3D

(2003) Desarrollo de X3D

Conversión a sintaxis XML y Modularización

Características: Gráficos en 2D y 3D, Animación, Audio/Vídeo,

Interacción con el usuario, *scripting*, simulaciones físicas: comportamientos humanos, espacios geográficos, etc.



2005

(2002) Se crea Open Mobile Alliance

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>

  <card id="Tarjeta1" title="Ejemplo ">
    <p> Hola</p>
    <p>Qué tal?</p>
    <anchor>Siguiente
      <go href="#siguiente">
    </anchor>
  </card>

  <card id="siguiente" title="siguiente">
    <p>Final</p>
  </card>

</wml>
```



Portales basados en voz

Ejemplos: Contestadores automáticos de empresas

Artefactos empotrados (coches)

Objetivos:

Fácil creación de contenido hablado

Reconocimiento/generación de voz

Interacción con el usuario

(1995) Phone Markup Language de AT&T

(1998) Se crea el VoiceXML Forum

(2000) VoiceXML 1.0

(2004) VoiceXML 2.0

```
<vxml version="2.0">
<form id="start">
  <field name="answer">
    <noinput> Hey, don't sleep! </noinput>
    <nomatch> say 'yes' or 'no' </nomatch>
    <prompt> Are you sleepy? </prompt>
    <grammar root="main">
      <rule id="main" scope="public">
        <one-of>
          <item><ruleref uri="#yes" tag="yes"/></item>
          <item><ruleref uri="#no" tag="no"/></item>
        </one-of>
      </rule>
      <rule id="yes">
        <one-of><item>yes</item><item>yeah</item></one-of>
      </rule>
      <rule id="no">
        <one-of><item>no</item><item>not</item></one-of>
      </rule>
    </grammar>
    <filled>
      <if cond="answer=='yes'">So you are sleepy. Me too.
      <else/>So you are not sleepy. But I am.
      </if>
    </filled>
  </field>
</form>
</vxml>
```



■ ■ ■ Versión 2005

Selección de Enlaces

Página del consorcio: <http://www.w3c.org>

En español: <http://www.it.uc3m.es/~xml/enlaces.html>

Especificación anotada: <http://www.xml.com/axml/testaxml.htm>

XML en industria: <http://www.xml.org>

Diseño de vocabularios XML: <http://www.xmlpatterns.com>

Tutoriales: <http://www.w3schools.com>

<http://www.brics.dk/~amoeller/XML/>

Artículos de XML:

<http://www.topxml.com>

<http://www.xmlpatterns.com>

Software de XML

<http://www.xmlsoftware.com>

<http://www.xmlhack.com>

<http://www.garshol.priv.no/download/xmltools/>



■ ■ ■ ■ ■ Curso sobre Tecnología Web ■ ■ ■ ■ ■ Versión 2005

Fin de la Presentación

