

Lenguaje XML

Jose Emilio Labra Gayo
Departamento de Informática
Universidad de Oviedo

¿HTML como formato de representación?

```
<html>
<head>
<title>Pizzeria Al Capone</title>
</head>
<body bgcolor="blue" text="yellow">
<h1>Pizzería Al Capone</h1>
<table>
<caption>Lista de Pizzas</caption>
<tr>
<td>Barbacoa</td>
<td>Mozzarella, Queso, Bacon</td>
<td>7&euro;</td>
</tr>
...
</body>
</html>
```

En HTML, las marcas tienen un significado predefinido

Mezcla información de la pizza con presentación en tabla

Información de la Pizza

```
<pizza nombre="Barbacoa">
<ingredientes nombres="Mozzarella Queso Bacon" />
<precio moneda="euro" valor="7" />
</pizza>
```

Posteriormente, podría representarse en una tabla...

O en otros formatos no previstos inicialmente: Estadísticas, WAP, TV, . . .

Lenguajes de Marcas: de SGML a XML

- SGML *Standard Generalized Markup Language*
 - Utilizado para el intercambio de documentos
 - Principio: Separar contenido de la forma de representarlo
 - Permite utilizar un conjunto de marcas específico para cada aplicación
 - HTML es un subconjunto de SGML
 - Problema de SGML: Demasiado complicado para su adopción en la Web
- XML
 - Desarrollado por el consorcio Web (1995)
 - Versión simplificada de SGML
 - Objetivos:
 - Standard de intercambio de información a través de la Web
 - Formato abierto, independiente de la plataforma
 - Permite utilizar vocabularios específicos de una aplicación
 - Permite la auto-descripción de dichos vocabularios (documentos auto-descritos)
 - Las aplicaciones pueden descubrir el formato de la información y actuar en consecuencia

Ejemplo de XML

Las marcas tienen un significado propio de la aplicación

pizzas.xml

```
<?xml version="1.0"?>
<!DOCTYPE pizzas SYSTEM "pizzas.dtd">
<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>
```

DTD = Declaración de Tipo de Documento

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>
```

Estructura de árbol

```

graph TD
  pizzas --> pizza1[pizza]
  pizzas --> dots1[...]
  pizzas --> pizza2[pizza]
  pizza1 --> ingrediente1[ingrediente]
  pizza1 --> dots2[...]
  pizza1 --> ingrediente2[ingrediente]
  pizza2 --> ingrediente3[ingrediente]
  pizza2 --> dots3[...]
  pizza2 --> ingrediente4[ingrediente]
  
```


Definición de XML 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

Otras características de XML

- Comentarios
 - `<!-- el texto de un comentario no es analizado -->`
- Secciones CDATA
 - Si se desea introducir código sin analizar

```
<codigo_HTML>
  <html>
    <body>Hola</body>
  </html>
</codigo_HTML>
```

```
<codigo_HTML>
  <![CDATA[
    <html>
      <body>Hola</body>
    </html>
  ]]>
</codigo_HTML>
```

- Elementos

- (?) = 0, 1 elemento
- (*) = 0 ó más elementos
- (+) = 1 ó más elementos

```
<!ELEMENT pizza (ingrediente*, inventor?)>
<!ELEMENT servicio (domicilio | restaurante) >
<!ELEMENT ingrediente EMPTY>
<!ELEMENT inventor (#PCDATA)>
```

- Atributos

- #REQUIRED Obligatorio
- #IMPLIED Opcional
- #FIXED Constante

- Tipos de datos

- CDATA = Cualquier string
- NMTOKEN = Palabra (sin espacios)
- NMTOKENS = Lista de palabras
- Enumeración separada por |
- ID = Nombre único
- IDREF = Su valor debe ser el mismo que el de un ID

```
<!ATTLIST pizza nombre CDATA #REQUIRED>
<!ATTLIST ingrediente nombre CDATA #REQUIRED
calorías CDATA #IMPLIED>
<!ATTLIST pizza precio (euros|dólares) #REQUIRED>
<!ATTLIST persona código ID #REQUIRED>
<!ATTLIST impuesto tipo CDATA #FIXED "IVA">
<!ATTLIST dueño código IDREF #REQUIRED>
```

- Entidades

```
<!ENTITY mezcla "Mezcla de 4 quesos">
```

```
<pizza nombre="4 Quesos" precio="7">
<ingrediente nombre="&mezcla;" />
</pizza>
```

- Entidades predefinidas

- Permiten incluir etiquetas sin analizar

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

- Entidades parámetro

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

- Entidades externas

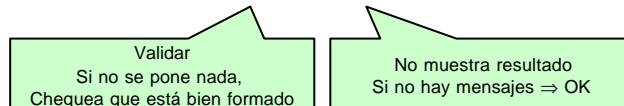
- Permiten dividir la definición en varios documentos

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

- Creación de ficheros XML y validación
- Procesadores de XML
 - Chequean que está bien formado
 - Validan
- Productos
 - Visuales: *XML Writer*, *XML Spy*, ...
 - Modo texto: *xmllint*, *msxml*, ...
- *xmllint* forma parte de la librería *libxml* de *GNOME*



```
xmllint --valid --noout fichero.xml
```



- 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
- Es una sintaxis de documentos, no un lenguaje de programación

```
int main(void) {
    printf("Hola");
    return 0;
}
```

```
<function name="main" type="int">
<arg type="void" />
<call function="printf">
  <param>Hola</param>
</call>
<return value="0"/>
</function>
```

- 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>
```

- Poco eficiente como lenguaje de almacenamiento de bases de datos

Espacios de Nombres

- XML NameSpaces permite especificar espacios de nombres para evitar colisiones de identificadores

```
<establecimiento>
<nombre>Pizzería Al Capone</nombre>
<direccion>C/ Génova Nº 3,Oviedo, España</direccion>
<teléfono>985203040</teléfono>
</establecimiento>
```

```
<persona>
<nombre>Vito Corleone</nombre>
<teléfono>985223344</teléfono>
<dni> 98765432</dni>
</persona>
```

...y si queremos añadir información del dueño?

```
<establecimiento>
  <nombre>Pizzería Al Capone</nombre>
  <direccion>C/ Génova Nº 3,Oviedo, España</direccion>
  <teléfono>985203040</teléfono>
  <dueño>
    <persona>
      <nombre>Vito Corleone</nombre>
      <teléfono>985223344</teléfono>
      <dni> 98765432</dni>
    </persona>
  </dueño>
</establecimiento>
```

<nombre> se refiere a persona o a establecimiento?

Espacios de Nombres

- Un alias se crea asignando un nombre a una URL
 - El ámbito del alias abarca al nodo y a sus hijos
- Espacio por defecto (no se pone alias)

```
<alias:etiqueta xmlns:alias="direccion URL">
  <alias:subetiquetas />
</alias:etiqueta>
```

```
<etiqueta xmlns="direccion URL">
  <subetiquetas />
</etiqueta>
```

```
<establecimiento xmlns="establecimiento.dtd"
  xmlns:p="persona.dtd">
  <nombre>Pizzería Al Capone</nombre>
  <direccion>C/ Génova Nº 3,Oviedo, España</direccion>
  <teléfono>985203040</teléfono>
  <dueño>
    <p:persona>
      <p:nombre>Vito Corleone</p:nombre>
      <p:teléfono>985223344</p:teléfono>
      <p:dni> 98765432</p:dni>
    </p:persona>
  </dueño>
</establecimiento>
```

Enlaces y referencias en XML

- Enlaces de HTML

```
<a href="http://www.uniovi.es">Universidad de Oviedo</a>
```

- Limitaciones
 - Empotrados en código fuente
 - Sólo permiten navegación en una dirección
 - Sólo conectan 2 recursos
 - No especifican el comportamiento del navegador
- XLink + XPointer = Solución aportada por XML

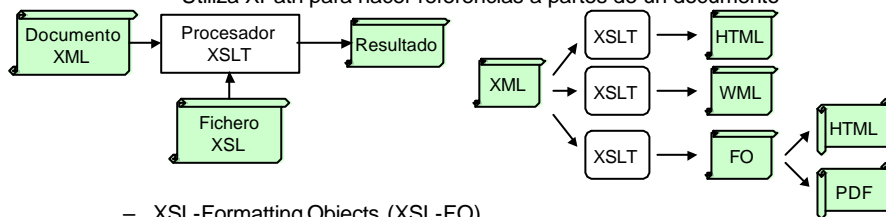
```
<patrocinadores xlink:href="patrocinadores.html"
  xlink:role="mostrar lista de patrocinadores"
  xlink:title="Lista de patrocinadores"
  xlink:show="new"
  xlink:actuate="onRequest"/>
```

- Problemas de los DTDs
 - Difíciles de manipular (no son XML)
 - No son extensibles (una vez definido, no es posible añadir nuevos vocabularios a un DTD)
 - No soportan tipos de datos (ej. enteros, flotantes, etc.)
- XML Schema = Vocabulario XML para definir esquemas de documentos

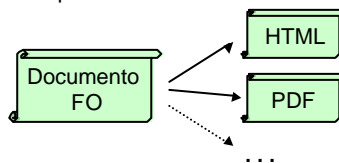
```
<xsd:element name="pizza" type="TipoPizza" />

<xsd:complexType name="TipoPizza">
  <xsd:sequence>
    <xsd:element name="nombre" type="xsd:string"/>
    <xsd:element name="ingrediente"
      type="TipoIngrediente"
      minOccurs="0"
      maxOccurs="unbounded" />
    <xsd:element name="precio" type="xsd:positiveInteger"/>
  </xsd:sequence>
</xsd:complexType>
...
```

- XSL (eXtensible Stylesheet Language)
 - XSL Transformations (XSLT)
 - Lenguaje de transformación de documentos
 - Utiliza XPath para hacer referencias a partes de un documento



- XSL-Formatting Objects (XSL-FO)
 - Lenguaje que incluye instrucciones de formato independientes del dispositivo



- El documento XML se puede asociar a una transformación XSLT
- Algunos visualizadores, al recibir un documento XML, transforman el documento y visualizan el resultado

```
<?xml version="1.0"?>
<!DOCTYPE pizzas SYSTEM "pizzas.dtd">
<?xml-stylesheet type="text/xsl" href="pizzas.xsl" ?>
<pizzas>
. . .
</pizzas>
```

- 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>
```

```
<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

- El procesador XSLT recorre el árbol desde la raíz
 - Recorre los elementos padre antes que los hijos
 - Para cada elemento, si existe una plantilla aplicable, se aplica y ya no se examinan más elementos descendientes (salvo que se solicite)
 - Principales Instrucciones
 - `<xsl:apply-templates select="Patrón de XPath">`
Solicita que continúe aplicando plantillas
 - `<xsl:value-of select="Expresión de XPath" />`
Generar un valor a partir de la expresión
 - `<xsl:for-each select="Expresión de XPath" />`
Para iterar sobre la serie de valores de la expresión
 - `<xsl:copy-of select="Expresión de XPath" />`
Copiar nodos del árbol

Expresiones de XPath

<code>a/b</code>	todos los <code></code> hijos directos de <code><a></code>
<code>a//b</code>	todos los <code></code> descendientes de <code><a></code>
<code>/</code>	nodo raíz
<code>.</code>	nodo actual
<code>..</code>	nodo padre del nodo actual
<code>*</code>	cualquier nodo
<code>a b</code>	elementos <code><a></code> o <code></code>
<code>a/b[1]</code>	primer <code></code> hijo directo de <code><a></code>
<code>//b</code>	todos los <code></code> en cualquier parte del documento
<code>./b</code>	todos los <code></code> descendientes del nodo actual
<code>a[@b]</code>	todos los <code><a></code> con el atributo <code>b</code>
<code>a[@b="hola"]</code>	todos los <code><a></code> cuyo atributo <code>b</code> valga "hola"
<code>\$a</code>	valor de la variable <code>a</code>

XPath incluye funciones para manipular datos de tipo booleano, numérico y cadenas de caracteres

- Instrucciones condicionales

<xsl:if> Condicional simple
 <xsl:choose> Condicional múltiple (case)
 <xsl:when> Elementos del condicional múltiple
 <xsl:otherwise> Valor por defecto

```

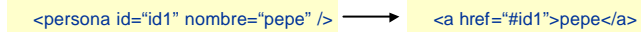
<xsl:choose>
<xsl:when test="$scontador = 2">...
<xsl:when test="$scontador = 2">...
<xsl:otherwise>...
</xsl:choose>
  
```

- Generación de nodos XML

<xsl:number> Añade un número
 <xsl:attribute> Añade un atributo
 <xsl:element> Añade un elemento
 <xsl:comment> Añade un comentario
 <xsl:processing-instruction> Genera una instrucción de procesamiento
 <xsl:text> Genera un nodo de texto

```

<xsl:element name="a">
<xsl:attribute name="href">
  #<xsl:value-of select="@id">
  <xsl:value-of select="@nombre">
</xsl:attribute>
</xsl:element>
  
```



- Otras instrucciones

<xsl:variable> Declara una variable
 XSLT es un lenguaje declarativo
 No hay asignación destructiva! ☺

```

<xsl:variable name="tamaño">5
</xsl:variable>

<xsl:if test="$tamaño = '5' ". . .
</xsl:if>
  
```

<xsl:sort> Clasificar nodos
 <xsl:include> Incluir otra hoja de estilos
 <xsl:import> Importar otra hoja
 <xsl:call-template> Llamar a otra plantilla
 <xsl:param> Declara el valor por defecto de un parámetro
 <xsl:with-param> Asignar un valor a un parámetro

Enumerar las pizzas con un índice hipertextual, ordenarlas por precio y mostrar la media de los precios,

The screenshot shows a web browser displaying a page titled "Pizzas del Restaurante Al Capone". The page content includes a list of pizza types: "Tipos de pizzas: 1. Barbacon | 2. Hawaiian | 3. 4 Quesos | 4. Margarita". Below this is a table with columns "Pizza", "Ingredientes", and "Precio". The table lists four pizzas: Margarita (price 6), Hawaiian (price 7), 4 Quesos (price 7), and Barbacon (price 8). At the bottom, it shows "Media de precios = 7". Callouts point to specific parts: "Enumerar las pizzas" points to the list of pizza types; "Incluir un índice al principio" points to the number '1' before 'Barbacon'; "Ordenar por precio" points to the 'Precio' column; and "Calcular la media de los precios" points to the average price calculation.

Pizza	Ingredientes	Precio
Margarita	Tomate Jamón Queso	6
Hawaiiana	Tomate Mozzarella Jamón Pina Queso	7
4 Quesos	Tomate Mezcla de 4 quesos	7
Barbacon	Salsa Barbacon Pollo Bacon Bacon Ternera	8

```

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

<xsl:template match="pizzas" mode="cabecera">
  <div class="header">Tipos de pizza:
  <xsl:for-each select="pizza">
    <xsl:number value="position()" format="1. " />
    <xsl:element name="a">
      <xsl:attribute name="href"=#P<xsl:number level="single"/></xsl:attribute>
      <xsl:value-of select="@nombre"/></xsl:element >
    </xsl:for-each>
  </div><hr/>
</xsl:template>
...

```

Callouts explain the XSLT code: "Dos modos de recorrido" points to the two `<xsl:apply-templates>` calls; "Numera las pizzas" points to the `<xsl:number>` call; and "Genera una lista de enlaces de la forma Margarita" points to the `<xsl:element name="a">` block.

```

<xsl:template match="pizzas">
<table border="1"><caption>Tipos de Pizzas</caption>
<tr><th>Pizza</th><th>Ingredientes</th><th>Precio</th></tr>
<xsl:for-each select="//pizza">
<xsl:sort data-type="number" select="@precio" />
<xsl:apply-templates select="." />
</xsl:for-each>
</table>
<p>Media de precios = <xsl:value-of select="sum(//@precio) div count(//@precio)" /></p>
</xsl:template>

<xsl:template match="pizza">
<tr><td><xsl:element name="a">
<xsl:attribute name="name">P<xsl:number level="single"/></xsl:attribute>
</xsl:element>
<xsl:value-of select="@nombre" />
</xsl:template>

```

Ordena las pizzas por precio

Calcula la media de los precios

Genera referencias de la forma ...

```

<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>

<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>

```

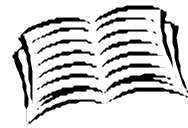
fact x = if x = 0 then 1 else x * fact (x - 1)

- Es un lenguaje de programación Turing-completo
 - Lenguaje declarativo (sin asignación destructiva)
 - Admite recursividad, funciones de orden superior, evaluación perezosa
 “The functional programming XSLT – a proof through examples” (D. Novatchev)
<http://www.xml.top/xsl/articles/fp>
- 4 tipos de datos (enteros, booleanos, strings y conjuntos de nodos)
 - Sin chequeo estático de tipos
 - Seguridad? Eficiencia? (no son objetivos de diseño)
- Flexible: La sintaxis de XPath se adapta a posibles cambios en la estructura. No se valida el documento
 - Bueno para hacer tareas sencillas rápidamente
 - Empotrado en navegadores
- Los programas XSLT son documentos XML
 - La sintaxis es poco amigable para el programador
 - Necesidad de *buenas* herramientas de autor

- | | |
|---|---|
| <ul style="list-style-type: none"> • MathML <ul style="list-style-type: none"> – Visualización de ecuaciones matemáticas) • SVG <ul style="list-style-type: none"> – Gráficos vectoriales • SMIL <ul style="list-style-type: none"> – Presentaciones multimedia • P3P <ul style="list-style-type: none"> – Descripción de características de privacidad • WML <ul style="list-style-type: none"> – Similar a HTML para teléfonos móviles • VoiceML <ul style="list-style-type: none"> – Portales basados en voz | <ul style="list-style-type: none"> • XML Signature <ul style="list-style-type: none"> • Firma de recursos Web • XKMS <ul style="list-style-type: none"> • Firmas y criptografía • XML Query <ul style="list-style-type: none"> – Consultas de documentos (Bases de datos) • XBRL <ul style="list-style-type: none"> – Contabilidad • ebXML <ul style="list-style-type: none"> • Negocios electrónicos (<i>e-business</i>) • SyncXML <ul style="list-style-type: none"> • Sincronización de dispositivos • UPnP <ul style="list-style-type: none"> • <i>Plug and Play</i> universal |
|---|---|

Selección de referencias

1. Página del consorcio: <http://www.w3c.org>
2. En español: <http://www.it.uc3m.es/~xml/enlaces.html>
3. Especificación anotada: <http://www.xml.com/axml/testaxml.htm>
4. XML en industria: <http://www.xml.org>
5. Artículos de XML: <http://www.topxml.com>
6. Software de XML
 1. <http://www.xmlsoftware.com>
 2. <http://www.xmlhack.com>
 3. <http://www.garshol.priv.no/download/xmltools/>



Fin de la Presentación