

Ciclo Formativo de Grado Superior de Administración de Sistemas Informáticos en red

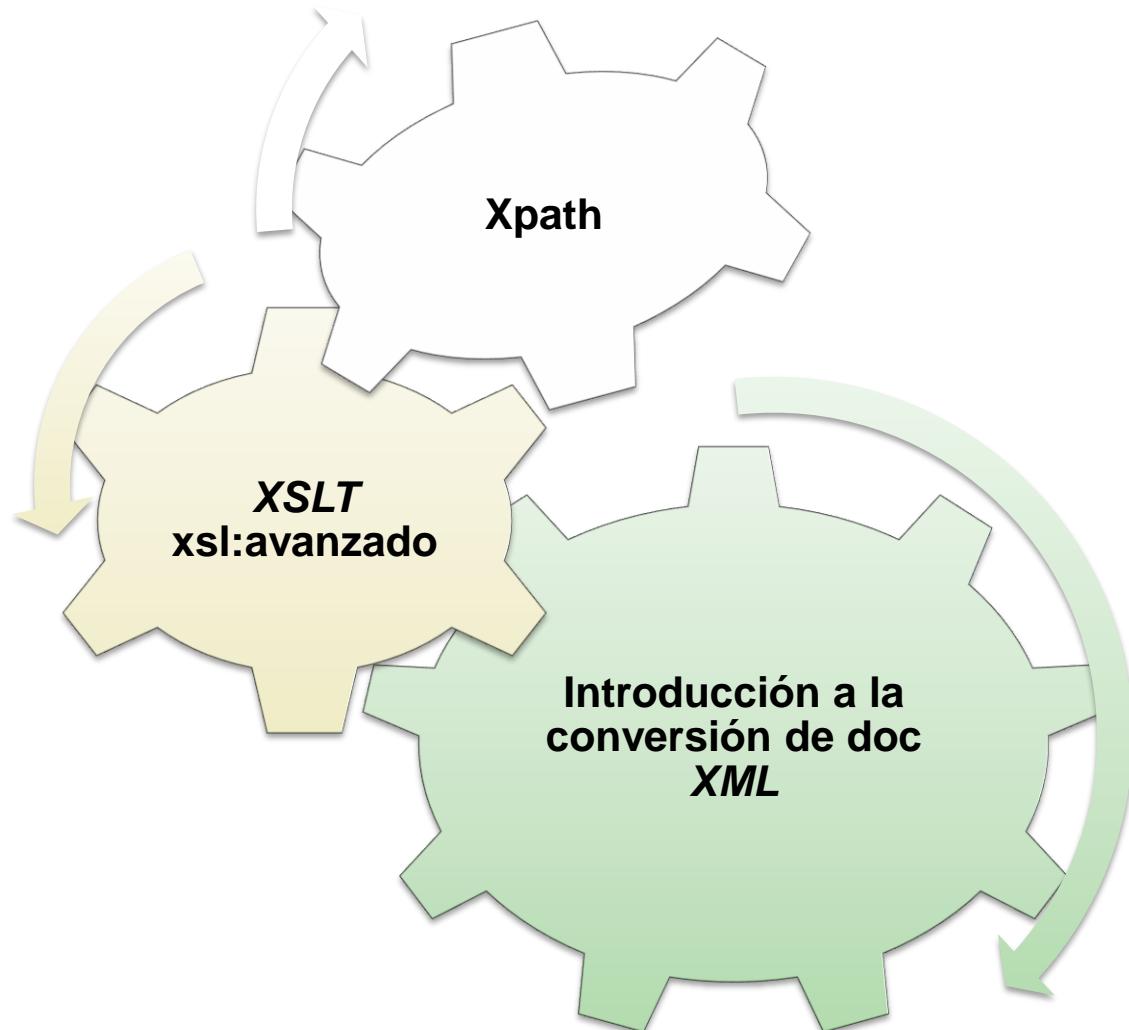


Módulo Profesional: **LMSGI**
U.T. 8.- Conversión y adaptación de documentos XML.
XSL/XSLT/XPATH ..

*Departamento de Informática y Comunicación
IES San Juan Bosco (Lorca-Murcia)
Profesor: Juan Antonio López Quesada*



Índice de Contenidos



Abstract/Resumen:

Al igual que XML, XSLT es un lenguaje de programación. Forma parte de la *trilogía transformadora* de XML, compuesta por las CSS (*Cascading Style Sheets*, hojas de estilo en cascada), que permite dar una apariencia en el navegador determinada a cada una de las etiquetas XML; XSLT (*XML Stylesheets Language for Transformation*, o lenguaje de transformación basado en hojas de estilo); y XSL:FO, (*Formatting Objects*, objetos de formateo), o transformaciones para fotocomposición, o, en general, para cualquier cosa que no sea XML, como por ejemplo HTML "del viejo" o PDF (el formato de Adobe).

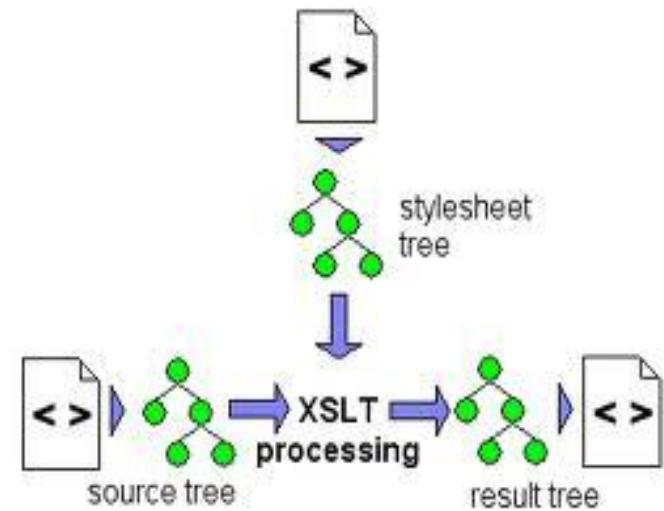
XSLT es pues, un lenguaje que se usa para convertir documentos XML en otros documentos XML; puede convertir un documento XML que obedezca a un DTD a otro que obedezca otro diferente, un documento XML bien formado a otro que siga un DTD, o, lo más habitual, convertirlo a "formatos finales", tales como WML (usado en los móviles WAP) o XHTML.



Introducción a la conversión de doc XML

Necesidades de Conversión

- ◆ XML se presenta como un estándar para “transmitir” datos a través de Internet.
- ◆ Ante la posibilidad de que distintos “centros” o “aplicaciones” utilicen esquemas o DTD diferentes, es necesario un sistema que permita “transformar” los datos de un documento XML.
- ◆ XSLT (*eXtensible Stylesheet Language – Transformations*), describe un lenguaje basado en XML para transformar documentos XML a cualquier otro formato.



Introducción a la conversión de doc XML

Aplicaciones de la Transformación

- ◆ Normalmente, utilizaremos XSLT para transformar documentos entre esquemas XML que permitan su procesamiento por distintos sistemas.
- ◆ También utilizaremos XSLT para transformar documentos XML en HTML, WML* o cualquier otro formato que facilite su presentación en la pantalla de un ordenador o en impresora.
- ◆ La transformación de XML a HTML es el principal uso que se hace de XSLT.

El Wireless Markup Language es un lenguaje cuyo origen es el XML (eXtensible Markup Language). Este lenguaje se utiliza para construir las páginas que aparecen en las pantallas de los teléfonos móviles y los asistentes personales digitales (PDA)

Introducción a la conversión de doc XML

Aplicaciones de la Transformación

- ◆ No debemos confundir las transformaciones XSLT con la presentación de documentos XML con CSS.
- ◆ Con XSLT, generaremos un documento HTML a partir de un documento XML. Se tratará de dos documentos “distintos”
- ◆ Con CSS, el navegador recibe un documento XML que formatea utilizando las reglas CSS para presentarlo en pantalla de forma que sea más fácilmente legible, pero es el mismo documento.

Introducción a la conversión de doc XML

XSLT, XSL, XSL FO...

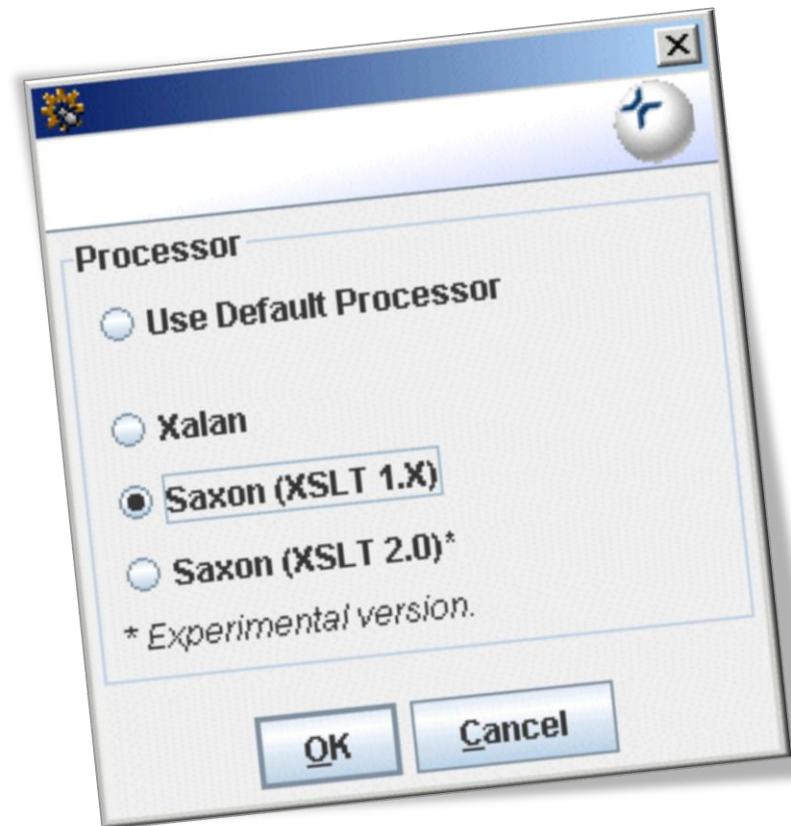
- La XSL es una especificación desarrollada dentro del W3C para aplicar formato a los documentos XML de forma estandarizada. La XSL es un lenguaje para escribir hojas de estilo que consta de dos partes:
 - **XSL-FO** (*eXtensible Stylesheet Language Formatting Objects*): *un lenguaje de formateo, que no es más que un vocabulario XML para especificar objetos de formateo (FO).*
 - **XSLT** (*eXtensible StyleSheet Language Transformations*), *es un lenguaje de transformación, mediante el cual se puede transformar un documento XML en otro XML.*
- **XSL FO** cuenta con escaso soporte por parte de la industria debido a su complejidad.
- Su propósito es definir la forma en la que se debe presentar un documento XML en papel o en pantalla.
- En este sentido, **XSL FO** sería una especificación similar a CSS.

Introducción a la conversión de doc XML

XSLT, XSL, XSL FO...

■ Actualmente contamos con varias *herramientas* para realizar transformaciones XSLT:

- Saxon, desarrollado en Java por Michael Kay (*un gurú de XSLT*)
- xt, diseñado por James Clark
- En las prácticas usaremos Altova XMLSpy



Introducción a la conversión de doc XML

Estructura de una hoja XSLT...

```
<?xml version="1.0"?>  
<fecha>  
    <dia>24</dia>  
    <mes>7</mes>  
    <anio>1982</anio>  
</fecha>
```

```
<xsl:stylesheet version="1.0"  
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
    <xsl:output method="xml" encoding="iso-8859-1"/>  
    <xsl:template match="fecha">  
        <cuando>  
            <xsl:value-of select="anio"/>  
        </cuando>  
    </xsl:template>  
</xsl:stylesheet>
```



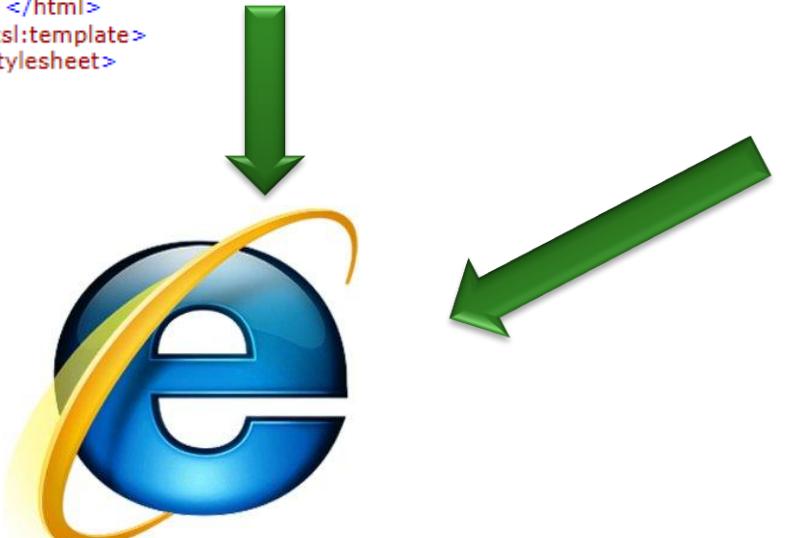
```
c:\tmp> xt documento1.xml documento1.xsl doc1.xml
```

```
<?xml version="1.0" encoding="iso-8859-1"?>  
<cuando>1982</cuando>
```

Introducción a la conversión de doc XML

Estructura de una hoja XSLT...

```
<?xml version="1.0" encoding="UTF-8"?>
- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  - <xsl:template match="libros">
    - <html>
      - <body>
        <h1>Mi primer documento XSLT</h1>
        <strong>Libro:</strong>
        <xsl:value-of select="libro/titulo"/>
        <br/>
        <strong>Autor:</strong>
        <xsl:value-of select="libro/autor"/>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```



A screenshot of the Windows WordPad application window titled 'libros - WordPad'. The XML code shown is:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="mi-primer-xslt.xsl"?>
<libros>
  <libro>
    <titulo>Fuente Ovejuna</titulo>
    <autor>Lope de Vega</autor>
    <isbn>84-9815-002-7</isbn>
  </libro>
  <libro>
    <titulo>La Celestina</titulo>
    <autor>Fernando de Rojas</autor>
    <isbn>84-96390-96-9</isbn>
  </libro>
  <libro>
    <titulo>Fuente Ovejuna</titulo>
    <autor>Lope de Vega</autor>
    <isbn>84-9815-002-7</isbn>
  </libro>
</libros>
```

To the right of the window, there is a stack of three books. The top book is black with 'XML' written in white. The middle book is grey with a blue cube icon. The bottom book is also grey.

Introducción a la conversión de doc XML

Estructura de una hoja XSLT...

- Una hoja de estilo XSLT es un documento XML. Debe estar **bien formado**.
- Las hojas de estilo se guardarán siempre en archivos independientes con extensión .xsl
- Deben comenzar con una declaración XML:
`<?xml version="1.0"?>`
- El elemento raíz de la hoja de estilo XSLT es **stylesheet**.
- Este elemento contendrá a todos los demás, y debe ir precedido por el alias xsl correspondiente al espacio de nombres para hojas de estilo XSLT.

Introducción a la conversión de doc XML

Estructura de una hoja XSLT...

- En las hojas de estilo XSLT, los nombres de los elementos “reservados” por la especificación, proceden de un mismo espacio de nombres, y por lo tanto deben escribirse precedidos por el correspondiente alias **xsl**.
- El alias debe “apuntar” a la URL:
 - <http://www.w3.org/1999/XSL/Transform>
- De esta forma, el elemento raíz quedará así:

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

Introducción a la conversión de doc XML

Estructura de una hoja XSLT...

- Entre las marcas de inicio y de fin del elemento raíz **xsl:stylesheet**, se escribirán las reglas de transformación propiamente dichas.
- Cada regla se definirá mediante un elemento ***xsl:template***.
- La regla indica qué instancias de los elementos del documento XML se van a transformar.
- La regla también indicará cómo se deben transformar cada una de ellas.

Introducción a la conversión de doc XML

Estructura de una hoja XSLT...

```
<xsl:template match="//nombre">  
  <h2>  
    <xsl:value-of select=".." />  
  </h2>  
</xsl:template>
```

Regla de Transformación

Xpath

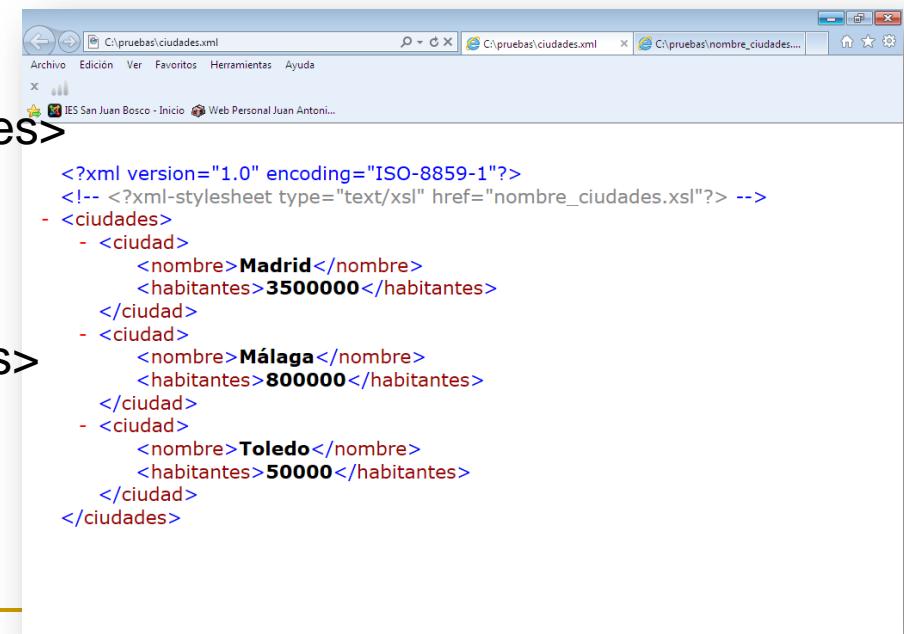
- La regla se aplicará a todas las instancias del elemento nombre que se encuentre en el xml al que se va a aplicar la transformación. Esto se indica mediante el atributo match que acompaña al elemento xsl:template.
- Entre las etiquetas de inicio y de fin del elemento xsl:template se escribe la transformación que se debe realizar, es decir, qué texto y qué marcas se escribirán en el documento resultado de la transformación, cada vez que se encuentre una instancia del elemento nombre en el documento origen.
- Con <xsl:value-of...>, se recupera y escribe en el documento resultado el valor del elemento que está siendo procesado.

Introducción a la conversión de doc XML

Ejemplo de Transformación XSLT

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xmlstylesheet type="text/xsl" href="nombre_ciudades.xsl"?>
<ciudades>
  <ciudad>
    <nombre>Madrid</nombre>
    <habitantes>3500000</habitantes>
  </ciudad>
  <ciudad>
    <nombre>Málaga</nombre>
    <habitantes>800000</habitantes>
  </ciudad>
  <ciudad>
    <nombre>Toledo</nombre>
    <habitantes>50000</habitantes>
  </ciudad>
</ciudades>
```

ciudades.xml

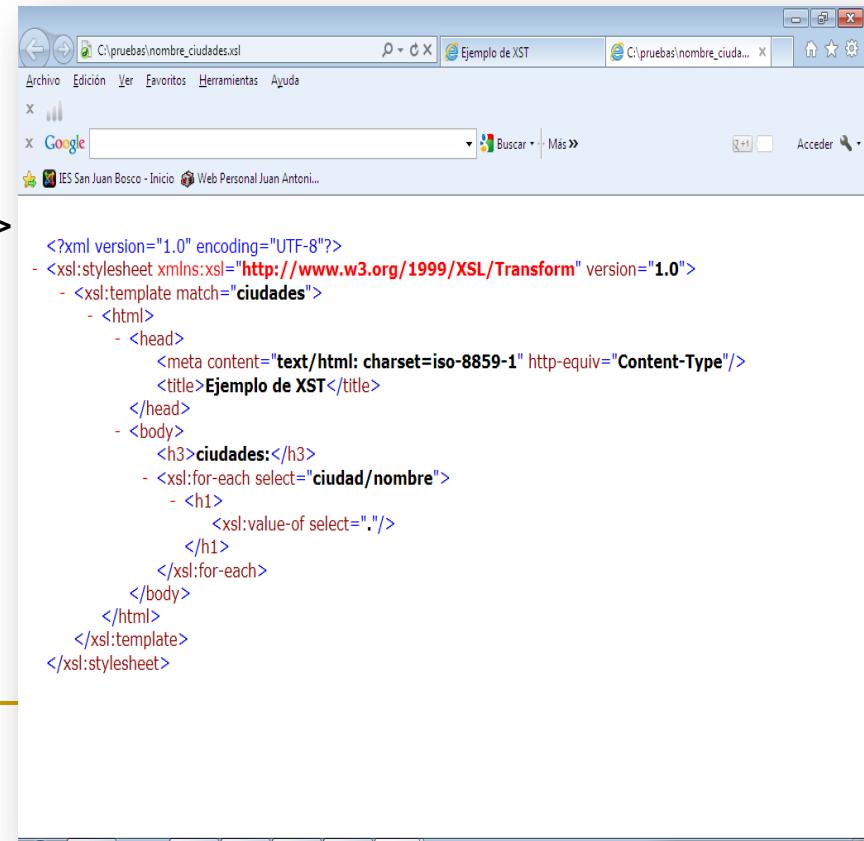


Introducción a la conversión de doc XML

Ejemplo de Transformación XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="ciudades">
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
            <title>Ejemplo de XST</title>
        </head>
        <body>
            <h3>ciudades:</h3>
            <xsl:for-each select="ciudad/nombre">
                <h1><xsl:value-of select=". ">
                </xsl:value-of></h1>
            </xsl:for-each>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

nombre_ciudades.xsl



Introducción a la conversión de doc XML

Ejemplo de Transformación XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
            <title>Ejemplo de XST</title>
        </head>
        <body>
            <h1>ciudades:</h1>
            <xsl:apply-templates select="//nombre" />
            <h1>ciudades:</h1>
            <xsl:apply-templates select="//nombre" />
        </body>
    </html>
</xsl:template>
<xsl:template match="//nombre">
    <h3><xsl:value-of select="."/></h3>
</xsl:template>
</xsl:stylesheet>
```

nombre_ciudades.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  - <xsl:template match="/">
    - <html>
      - <head>
        <meta content="text/html; charset=iso-8859-1" http-equiv="Content-Type"/>
        <title>Ejemplo de XST</title>
      </head>
      - <body>
        <h1>ciudades:</h1>
        <xsl:apply-templates select="//nombre"/>
        <h1>ciudades:</h1>
        <xsl:apply-templates select="//nombre"/>
      </body>
    </html>
  </xsl:template>
  - <xsl:template match="//nombre">
    - <h3>
      <xsl:value-of select="."/>
    </h3>
  </xsl:template>
</xsl:stylesheet>
```

Introducción a la conversión de doc XML

Ejemplo de Transformación XSLT

- La regla `<xsl:template match="/">` se ejecuta cuando se encuentra el elemento raíz del documento XML.
- Dentro de esta regla, podemos incluir llamadas a otras reglas definidas en la hoja de estilo, mediante el elemento:
`<xsl:apply-templates select="..." />`
- El atributo `select` tomará como valor el nombre del elemento asociado a la regla que queremos “disparar”.
- Esto nos ofrece un control real sobre el “orden” de ejecución de las reglas.

Introducción a la conversión de doc XML

Ejemplos

```
<?xml version="1.0" encoding="UTF-8"?>
<libros>
    <libro>
        <titulo>Fuente Ovejuna</titulo>
        <autor>Lope de Vega</autor>
        <isbn>84-9815-002-7</isbn>
    </libro>
    <libro>
        <titulo>La Celestina</titulo>
        <autor>Fernando de Rojas</autor>
        <isbn>84-96390-96-9</isbn>
    </libro>
    <libro>
        <titulo>Fuente Ovejuna</titulo>
        <autor>Lope de Vega</autor>
        <isbn>84-9815-002-7</isbn>
    </libro>
</libros>
```

Introducción a la conversión de doc XML

Ejemplos

- Lo que vamos a hacer mediante el XSLT es obtener información del documento XML y crear un documento XHTML donde se vean dichos datos.
- Para construir el documento XSLT lo primero que tenemos que saber es que, este, es a su vez otro documento XML. Por lo tanto, la primera línea que nos encontraremos será la definición del XML.

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

- Lo siguiente es empezar a definir la hoja de estilos y el namespace asociado:

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

Introducción a la conversión de doc XML

Ejemplos

- En todo documento XSLT, su cuerpo, es formado por las plantillas. Estas nos indicaran a partir de que elemento del documento XML vamos a empezar a trabajar. Para nuestro ejemplo, y ya que vamos a pintar en pantalla el primer libro y el primer autor, nos posicionaremos dentro de la estructura libros de la siguiente forma:

```
<xsl:template match="libros">
```

- El propio documento XSLT combina sentencias de transformación con código HTML.

Introducción a la conversión de doc XML

Ejemplos

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
    <title>Ejemplo de XST</title>
  </head>
  <body>
    ...Código XSLT...
  </body>
</html>
```

Aunque las sentencias del XSLT son bastantes, la que nosotros necesitamos conocer en este momento es la que nos ayuda a obtener el valor de una etiqueta XML. Cabe indicar que todas las etiquetas XSLT empiezas por `xsl` que es el namespace definido anteriormente.

Introducción a la conversión de doc XML

Ejemplos

- Así, para recuperar el valor de una etiqueta usamos `xsl:value-of`. Como atributo de esta etiqueta encontramos `select`. El cual, nos sirve para seleccionar el nodo (elemento) del cual queremos extraer su valor:

```
<xsl:value-of select="libro/titulo"/>  
<xsl:value-of select="libro/autor"/>
```

- Tanto en el template con el atributo `match`, como en la etiqueta `xsl:value-of` con el atributo `select` estamos utilizando valores **XPath**. Estos lo que vienen a reflejar, en grandes rasgos, son elementos dentro del árbol xml.
- Lo último que nos quedará hacer es reflejar en el documento xml la relación con su fichero de transformación. Para ello utilizamos la siguiente línea dentro del fichero xml

Introducción a la conversión de doc XML

Ejemplos

```
<?xmlstylesheet type="text/xsl" href="MiPrimerXSLT.xsl"?>
```

- Seguro que en el futuro, todos los navegadores incluirán el soporte de este potente lenguaje de transformación.

Mi primer documento XSLT

Titulo:Fuente Ovejuna

Autor:Lope de Vega

Introducción a la conversión de doc XML

Ejemplos

- ❖ Podríaadirse la plantilla de estilos asociada a la información que se va a visualizar:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="libros">
    <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
            <title>Ejemplo de XST</title>
            <style type="text/css">
                .estilo1 {
                    color:red;
                }
            </style>
        </head>
        <body>
            <h1> Mi primer documento XSLT </h1>
            <strong class="estilo1">Titulo:</strong>
            <xsl:value-of select="libro/titulo"/><br/>
            <strong class="estilo1">Autor:</strong>
            <xsl:value-of select="libro/autor"/>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

Mi primer documento XSLT

Titulo:Fuente Ovejuna
Autor:Lope de Vega

Introducción a la conversión de doc XML

Ejemplos

- A la hora de presentar datos una de los artificios más usados son las tablas. Con el lenguaje HTML es muy sencillo el montar una de ellas. Nos basta con controlar las etiquetas **table**, **tr** y **td**.
- En el siguiente ejemplo utilizaremos XSLT para montar una tabla XHTML. Los datos de dicha tabla serán los datos que vayan en el XML. Para ello, lo primero queharemos será crear nuestro documento XML con la información a mostrar. Usamos un XML que representa información de libros, utilizado anteriormente.
- Básicamente, como descripción de nuestro ejemplo, lo que vamos a hacer es recorrer los elementos de un **xpath** determinado e ir creando filas de la tabla. Pero vamos por partes. Lo primero que tenemos que hacer en nuestro documento XSL, dentro de la plantilla principal es declarar la cabecera de la tabla:

Introducción a la conversión de doc XML

Ejemplos

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
            <title>Tablas</title>
        </head>
        <body>
            <h1>Mis Libros</h1>
            <table border="1">
                <tr bgcolor="skyblue">
                    <th align="left">Titulo</th>
                    <th align="left">Autor</th>
                </tr>
                .....
            
```

Introducción a la conversión de doc XML

Ejemplos

- Una vez que tenemos la cabecera, tenemos que iterar por los elementos. Para cada fila hay que crear una etiqueta TR, que es una fila, y para cada elemento una etiqueta TD, que es una celda. Quedándonos el siguiente código:

```
<xsl:for-each select="libros/libro">  
    <tr>  
        <td><xsl:value-of select="titulo"/></td>  
        <td><xsl:value-of select="autor"/></td>  
    </tr>  
</xsl:for-each>  
</table>  
</body>  
</html>  
</xsl:template>  
</xsl:stylesheet>
```

Introducción a la conversión de doc XML

Ejemplos

- Podemos comprobar que estamos mezclando el código XHTML con el código XSLT sin ningún problema. Para recorrer los elementos del documento XML utilizamos la etiqueta `xsl:for-each`, la cual, mediante el atributo `SELECT` identifica un ***path en el árbol ARBOL***.
- En nuestro ejemplo, al mostrar los contenidos de los libros, ***el path será libros/libro***.
- Los valores de los elementos son recuperados mediante el `xsl:value-of`, que al igual que sucede con `xsl:for-each`, tiene un atributo `SELECT` con el elemento XML a recuperar.

Mis Libros

Titulo	Autor
Fuente Ovejuna	Lope de Vega
La Celestina	Fernando de Rojas
Don Juan Tenorio	Jose Zorilla

Introducción a la conversión de doc XML

Ejemplos

- Nuestras hojas de transformación no se tienen que limitar a recuperar toda la información de un fichero XML y a ponerla en otro tipo de formato.
- Puede ser que cierta información del documento no nos interese, o tenga que ser evaluada. Es por ello que puede darse el caso de que necesitemos filtrar información en una XSLT.
- Para poder realizar esa evaluación de datos nos podemos apoyar en estructuras condicionales como xsl:if, xsl:choose y las funciones que tiene el lenguaje XSL.
- Vamos a utilizar el fichero XML con los datos de los libros. Pero en este caso vamos a añadir un nuevo campo en la estructura. Este nuevo campo será el precio del libro. El fichero de libros nos quedará así:

Introducción a la conversión de doc XML

Ejemplos

```
<?xml version="1.0" encoding="UTF-8"?>
<libros>
    <libro>
        <titulo>Fuente Ovejuna</titulo>
        <autor>Lope de Vega</autor>
        <isbn>84-9815-002-7</isbn>
        <precio>24</precio>
    </libro>
    <libro>
        <titulo>La Celestina</titulo>
        <autor>Fernando de Rojas</autor>
        <isbn>84-96390-96-9</isbn>
        <precio>32</precio>
    </libro>
</libros>
<libro>
    <titulo>Fuente Ovejuna</titulo>
    <autor>Lope de Vega</autor>
    <isbn>84-9815-002-7</isbn>
    <precio>56</precio>
</libro>
<libro>
    <titulo>Fuente Ovejuna</titulo>
    <autor>Lope de Vega</autor>
    <isbn>84-9815-002-7</isbn>
    <precio>22</precio>
</libro>
</libros>
```

Introducción a la conversión de doc XML

Ejemplos

- La condición de nuestra hoja de transformación será el generar un listado de libros, pero siempre y cuando, estos, tengan un precio superior a 30 euros.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
  <h1>Mis Libros</h1>
  <xsl:for-each select="libros/libro">
    <xsl:if test="precio > 30">
      Titulo:<xsl:value-of select="titulo"/><br/>
      Autor:<xsl:value-of select="autor"/><br/>
      Precio:<xsl:value-of select="precio"/><br/>
    </xsl:if>
  </xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```



Introducción a la conversión de doc XML

Ejemplos

- Cada día tratamos más fuentes de información basadas en XML. Ficheros que intercambiamos, bases de datos basadas en XML o simplemente registros que contienen información variopinta.
- Podemos apoyarnos en XSLT para dejar ordenado un fichero XML. Y al igual que cuando tratamos una base de datos, podemos dejarle ordenado por el campo que nosotros queramos.
- En primer lugar vamos a recorrer los elementos que queremos mostrar. En nuestro caso mostraremos los libros y autores de los mismos. Esto lo podemos llevar a cabo con la etiqueta `xsl:for-each`, la cual, mediante el atributo `SELECT` demarca el path del documento XML sobre el que queremos iterar. El path elegido será `libros/libro`.
- En el caso de que queramos recuperar los valores de un elemento en concreto utilizamos la etiqueta `xsl:value-of`. En este etiqueta, al igual que en la etiqueta `xsl:for-each`, el atributo `select` nos indicará el elemento a recuperar.
- Para ordenar los elementos utilizamos la etiqueta `xsl:sort`. Esta etiqueta tiene un atributo `select` que indica el path sobre el que queremos ordenar.

Introducción a la conversión de doc XML

Ejemplos

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
  <h1>Mis Libros</h1>
  <xsl:for-each select="libros/libro">
    <xsl:sort select="titulo"/>
    Titulo:<xsl:value-of select="titulo"/><br/>
    Autor:<xsl:value-of select="autor"/><br/>
    Precio:<xsl:value-of select="precio"/><br/>
  </xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

<xsl:sort lang="idioma" data-type=" text"
order=" ascending | descending"
case-order="upper-first|lower-first"
select="expresion_XPath">
</xsl:sort>

Mis Libros

Titulo:Fuente Ovejuna
Autor:Lope de Vega
Precio:24

Titulo:Fuente Ovejuna
Autor:Lope de Vega
Precio:56

Titulo:Fuente Ovejuna
Autor:Lope de Vega
Precio:22

Titulo:La Celestina
Autor:Fernando de Rojas
Precio:32

Introducción a la conversión de doc XML

Ejemplos

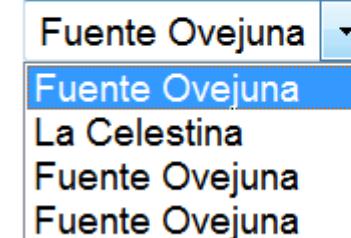
- La pretensión de este ejemplo es ver como podemos cargar un combo (desplegable) mediante una hoja XSLT. Es decir, utilizar los datos de un XML para que aparezcan como opciones de un combo.
- Para conseguir la carga del combo tenemos que definir una plantilla. Dicha plantilla lo que tendrá que hacer es recorrer los elementos libro e ir generando las etiquetas que son las que representan el contenido del combo. Para iterar sobre los elementos utilizamos la etiqueta `xsl:for-each`.
- En dicha etiqueta tenemos que establecer como atributo de iteración, `select`, el conjunto de etiquetas sobre las que queremos iterar. En nuestro caso `libros/libro`. Y en cada iteración obtener el valor del atributo.
- Para esto utilizaremos la etiqueta `xsl:value-of`, que al igual que la anterior tiene un atributo `select`, que en este caso indica la etiqueta de la cual se quiere recuperar el valor.

Introducción a la conversión de doc XML

Ejemplos

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
  <h1>Mis Libros</h1>
  <select>
    <xsl:for-each select="libros/libro">
      <option><xsl:value-of
select="titulo"/></option>
    </xsl:for-each>
  </select>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Mis Libros



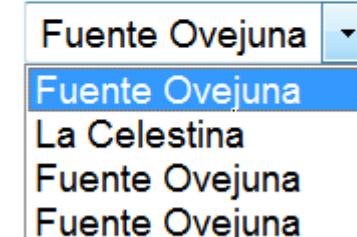
Introducción a la conversión de doc XML

Ejemplos

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
  <h1>Mis Libros</h1>
    <xsl:element name="select">
      <xsl:for-each select="libros/libro">
        <xsl:element name="option">
          <xsl:attribute name="value"> <xsl:value-of select="titulo"/>
          </xsl:attribute>
          <xsl:value-of select="titulo"/>
        </xsl:element>
      </xsl:for-each>
    </xsl:element>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML><HEAD>
<META content="text/html; charset=unicode" http-equiv="Content-Type">
<META name=GENERATOR content="MSHTML
9.00.8112.16443"></HEAD>
<BODY>
<H1>Mis Libros</H1><SELECT><OPTION selected
value="Fuente Ovejuna">Fuente
Ovejuna</OPTION> <OPTION value="La Celestina">La
Celestina</OPTION> <OPTION
value="Fuente Ovejuna">Fuente Ovejuna</OPTION>
<OPTION
value="Fuente Ovejuna">Fuente
Ovejuna</OPTION></SELECT> </BODY></HTML>
```

Mis Libros



Introducción a la conversión de doc XML

Ejemplos

- Si estás tratando un fichero XML es muy probable que necesites conocer el número de elementos que lo componen, de cara a poderlo manipular de una forma más sencilla. Sobre todo si están iterando en un bucle por el número de elementos de un determinado nodo.
- Saber el número de elementos de un fichero xml dado un nodo en concreto es una tarea muy sencilla.
- Lo primero que haremos en nuestro fichero XSLT será el saber que parte de la estructura del documento XML nos interesa contar. Es decir, en un nodo en concreto, para saber el número de elementos que dicha estructura tiene. En este ejemplo, la idea es contar el número de libros que hay en el documento XML.
- Para contar el número de elementos disponemos de una función. Esta es la función count(): count(xpath)
- El parámetro de la función será una expresión xpath que haga relación a alguna estructura del documento XML que estamos tratando.

Introducción a la conversión de doc XML

Ejemplos

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/libros">
  <html>
    <body>
      <h1>Mis Libros: <xsl:value-of select="count(libro)"/></h1>
      <xsl:for-each select="libro">
        <xsl:value-of select="titulo"/><br/>
      </xsl:for-each>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Mis Libros: 4

Fuente Ovejuna
La Celestina
Fuente Ovejuna
Fuente Ovejuna

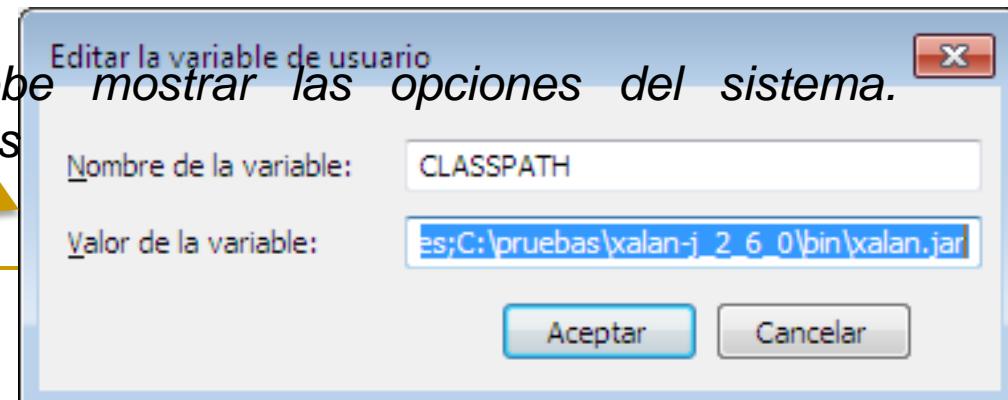
Introducción a la conversión de doc XML

EJERCICIOS GUIADOS HACIENDO USO DEL PROCESADOR XALAN

- El procesador Xalan de Apache, que también incluye el analizador XML Xerces, es una herramienta escrita en Java, y lo único que se necesita para hacerla funcionar es una máquina virtual Java (JVM), tal como la de Microsoft, Sun o IBM. Desde Java tiene la ventaja de que los desarrollos son válidos tanto para Windows como para Linux. Para instalarlo se puede obtener la última versión en el servidor de RedIris.

Pasos de la instalación <http://apache.xmlcity.org/>:

- *Disponer del RunTime de Java o instalarlo desde SUN.*
- *Desempaquetar xalan-j-current-bin-2jars.zip y dejar el resultado en el sitio definitivo. <http://apache.xmlcity.org/xml/xalan-j/xalan-j-current-bin-2jars.zip>*
- *Añadir al CLASSPATH los .jar (ficheros de librería) Xalan (el procesador de XSLT):*
- *Probar la instalación. Esto debe mostrar las opciones del sistema.*
`java org.apache.xalan.xslt.Process`



Introducción a la conversión de doc XML

EJERCICIOS GUIADOS HACIENDO USO DEL PROCESADOR XALAN

- Para empezar, vamos a tratar de presentar una hoja XML de lo más simple (tienda0.xml):

```
<?xml version="1.0" encoding='ISO-8859-1'?>
<?xmlstylesheet href="tienda0.xsl" type="text/xsl"?>
<tienda>
  <nOMBRE>La tiendecilla</nOMBRE>
  <TELEFONO>953 87 12 23</TELEFONO>
</tienda>
```

- Este ejemplo lo iremos extendiendo hasta que contenga un catálogo de una tienda virtual. Por lo pronto incluimos solamente datos básicos sobre la tienda.

Introducción a la conversión de doc XML

EJERCICIOS GUIADOS HACIENDO USO DEL PROCESADOR XALAN

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match='/'>
<html>
  <head><title>Generado con tienda-html.xsl</title></head>
  <body>
    <h1> <xsl:apply-templates /> </h1>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

```
java org.apache.xalan.xslt.Process -IN
tienda0.xml -XSL tienda-html.xsl -OUT
tienda0.html
```

Introducción a la conversión de doc XML

EJERCICIOS GUIADOS HACIENDO USO DEL PROCESADOR XALAN

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright © 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Profesor>cd c:\pruebas

c:\pruebas>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 8439-6514

Directorio de c:\pruebas

30/04/2012  21:02    <DIR>          .
30/04/2012  21:02    <DIR>          ..
10/03/2011  11:47            330 tienda-html.xsl
10/03/2011  11:47            189 tienda0.xml
27/02/2004  13:33    <DIR>          xalan-j_2_6_0
                  2 archivos           519 bytes
                  3 dirs   102.427.607.040 bytes libres

c:\pruebas>java org.apache.xalan.xslt.Process -IN tienda0.xml
-XSL tienda-html.xsl -OUT tienda0.html
```

Introducción a la conversión de doc XML

EJERCICIOS GUIADOS HACIENDO USO DEL PROCESADOR XALAN

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Profesor>cd c:\pruebas

c:\pruebas>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 8439-6514

Directorio de c:\pruebas

30/04/2012 21:02    <DIR>   .
30/04/2012 21:02    <DIR>   .
10/03/2011 11:47      330 tienda-html.xls
10/03/2011 11:47      189 tienda0.xml
27/02/2004 13:33    <DIR>   xalan-j_2_6_0
                  2 archivos      519 bytes
                  3 dirs  102.427.607.040 bytes libres

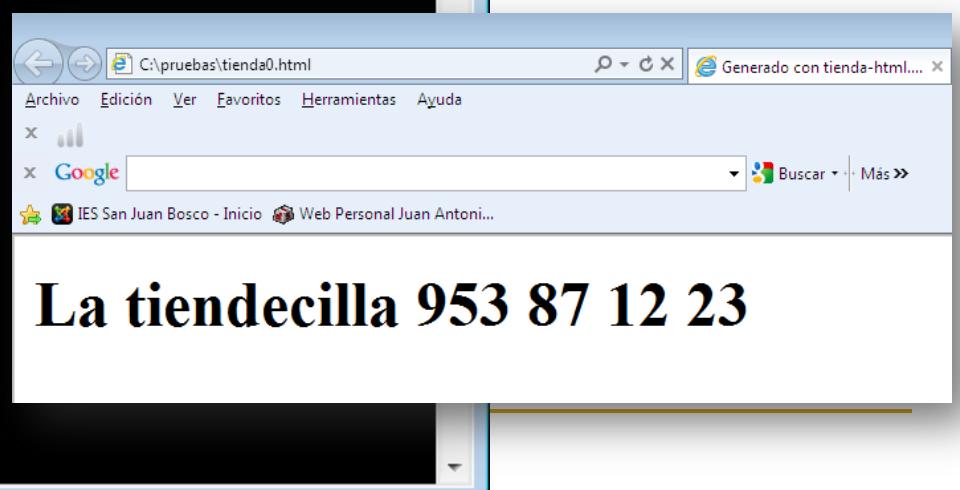
c:\pruebas>java org.apache.xalan.xslt.Process -IN tienda0.xml
-XSL tienda-html.xls -OUT tienda0.html

c:\pruebas>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 8439-6514

Directorio de c:\pruebas

30/04/2012 21:05    <DIR>   .
30/04/2012 21:05    <DIR>   .
10/03/2011 11:47      330 tienda-html.xls
30/04/2012 21:05      210 tienda0.html
10/03/2011 11:47      189 tienda0.xml
27/02/2004 13:33    <DIR>   xalan-j_2_6_0
                  3 archivos      729 bytes
                  3 dirs  102.427.623.424 bytes libres

c:\pruebas>_



The screenshot shows a Microsoft Internet Explorer window with the following details:



- Title bar: C:\pruebas\tienda0.html
- Toolbar: Back, Forward, Stop, Refresh, Home, Favorites, Tools, Help.
- Address bar: Google
- Search bar: Buscar
- Links: IES San Juan Bosco - Inicio, Web Personal Juan Antoni...
- Content area: The page displays the converted HTML content: "La tiendecilla 953 87 12 23".

```

Introducción a la conversión de doc XML

EJERCICIOS GUIADOS HACIENDO USO DEL PROCESADOR XALAN

- Veamos un documento XML algo más complicado: vamos a tratar de procesar documentos XML que describen diferentes artículos de una tienda. Una tienda virtual está formada por diferentes productos, que a su vez tienen una serie de características: código de producto, tipo de artículo, sección. Una tienda virtual se describiría en un fichero como el siguiente (tienda1.xml):

```
<?xml version="1.0" encoding='ISO-8859-1'?>
<tienda>
<nombre>Tienda Virtual </nombre>
<telefono>923 20 20 20 </telefono>
<producto>
<codigo>92 </codigo>
<cantidad>10 </cantidad>
<articulo>Radio-Cassete </articulo>
</producto>
<producto>
<codigo>103 </codigo>
<cantidad>50 </cantidad>
<articulo>Reloj Cocina </articulo>
</producto>
<producto>
<codigo>1312 </codigo>
<cantidad>3 </cantidad>
<articulo>Sofá </articulo>
</producto>
</tienda>
```

Introducción a la conversión de doc XML

EJERCICIOS GUIADOS HACIENDO USO DEL PROCESADOR XALAN

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

```
<xsl:stylesheet version="1.0"
```

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

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

```
    <html>
```

```
        <xsl:apply-templates />
```

```
    </html>
```

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

The screenshot shows a Windows Command Prompt window titled 'Administrador: C:\Windows\system32\cmd.exe'. The window displays the following command-line session:

```
c:\> Administrador: C:\Windows\system32\cmd.exe
10/03/2011 11:47      189 tienda0.xml
27/02/2004 13:33  <DIR>      xalan-j_2_6_0
                  2 archivos      519 bytes
                  3 dirs  102.427.607.040 bytes libres

c:\>pruebas>java org.apache.xalan.xslt.Process -IN tienda0.xml
          -XSL tienda-html.xsl -OUT tienda0.html

c:\>pruebas>dir
El volumen de la unidad C no tiene etiqueta.
El n mero de serie del volumen es: 8439-6514

Directorio de c:\>pruebas

30/04/2012 21:05  <DIR>      .
30/04/2012 21:05  <DIR>      330 tienda-html.xsl
10/03/2011 11:47            210 tienda0.html
30/04/2012 21:05            189 tienda0.xml
10/03/2011 11:47            729 bytes
27/02/2004 13:33  <DIR>      xalan-j_2_6_0
                  3 archivos      729 bytes
                  3 dirs  102.427.623.424 bytes libres

c:\>pruebas>java org.apache.xalan.xslt.Process -IN tienda1.xml
          -XSL tienda1-html.xsl -OUT tienda1.html

c:\>pruebas>dir
El volumen de la unidad C no tiene etiqueta.
El n mero de serie del volumen es: 8439-6514

Directorio de c:\>pruebas

30/04/2012 21:34  <DIR>      .
30/04/2012 21:34  <DIR>      906 tienda1-html.xsl
30/04/2012 21:33            619 tienda1.html
30/04/2012 21:34            548 tienda1.xml
27/02/2004 13:33  <DIR>      xalan-j_2_6_0
                  3 archivos      2.073 bytes
                  3 dirs  102.425.157.632 bytes libres

c:\>pruebas>
```

Introducción a la conversión de doc XML

EJERCICIOS GUIADOS HACIENDO USO DEL PROCESADOR XALAN

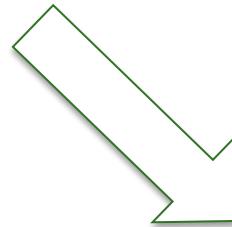
```
<xsl:template match='tienda'>
    <head><title><xsl:value-of select='nombre' /> (Generado por tienda1-
html.xsl)</title></head>
    <body>
        <h1><xsl:value-of select='nombre' /> </h1>
        <h2>Teléfono: <xsl:value-of select='telefono' /> </h2>
        <h2>Nuestros mejores productos </h2>
        <table>
            <tr><th>Código</th><th>Existencias</th><th>Artículo</th></tr>
            <xsl:apply-templates select='producto' />
        </table>
    </body>
</xsl:template>
```

Introducción a la conversión de doc XML

EJERCICIOS GUIADOS HACIENDO USO DEL PROCESADOR XALAN

```
<xsl:template match='producto'>  
    <tr><xsl:apply-templates select='codigo|cantidad|articulo' /></tr>  
</xsl:template>
```

```
<xsl:template match='codigo|cantidad|articulo'>  
    <td><xsl:value-of select='.' /></td>  
</xsl:template>  
</xsl:stylesheet>
```



La tiendecilla

Teléfono: 953 87 12 23

Nuestros mejores productos

Código	Existencias	Artículo
[92]	[10]	[Radio-Casette]
[103]	[50]	[Reloj Cocina]
[1312]	[3]	[Sofá]

Introducción a la conversión de doc XML

EJERCICIOS GUIADOS HACIENDO USO DEL PROCESADOR XALAN

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>La tiendecilla (Generado por tienda1-html.xsl)</title>
</head>
<body>
<h1>La tiendecilla</h1>
<h2>Tel&eacute;fono: 953 87 12 23</h2>
<h2>Nuestros mejores productos </h2>
<table>
<tr><th>C&oacute;digo</th><th>Existencias</th><th>Art&iacute;culo</th></tr>
<tr><td>[92]</td><td>[10]</td><td>[Radio-Casette]</td></tr>
<tr><td>[103]</td><td>[50]</td><td>[Reloj Cocina]</td></tr>
<tr><td>[1312]</td><td>[3]</td><td>[Sof&aacute;]</td></tr>
</table>
</body>
</html>
```

Introducción a la conversión de doc XML

EJERCICIO GUIADO XSLT, XSL-FO y FOP

La XSL es una especificación desarrollada dentro del W3c para aplicar formato a los documentos XML de forma estandarizada. La XSL es un lenguaje para escribir hojas de estilo que consta de dos partes:

XSL-FO (*eXtensible Stylesheet Language Formatting Objects*): *un lenguaje de formateo, que no es más que un vocabulario XML para especificar objetos de formateo (FO).*

XSLT (*eXtensible StyleSheet Language Transformations*), *es un lenguaje de transformación, mediante el cual se puede transformar un documento XML en otro XML.*

Cuando trabajamos con XML, una de las técnicas más útiles es la utilización de XSL para generar distintas salidas (HTML, WML, etc)

En numerosas ocasiones, la salida que debemos generar es un documento más complejo (PDF, Word, etc..) para ello, se extendió XSL incluyendo características para definir la representación gráfica de los elementos. Esta extensión se denomina XSL:FO, Formatting Object.

Introducción a la conversión de doc XML

EJERCICIO GUIADO XSLT, XSL-FO y FOP

Apache™ FOP (Formatting Objects Processor) is a print formatter driven by XSL formatting objects (XSL-FO) and an output independent formatter. It is a Java application that reads a formatting object (FO) tree and renders the resulting pages to a specified output. Output formats currently supported include PDF, PS, PCL, AFP, XML (area tree representation), Print, AWT and PNG, and to a lesser extent, RTF and TXT. The primary output target is PDF.

The Apache™ FOP project is part of the Apache™ Software Foundation, which is a wider community of users and developers of open source projects.



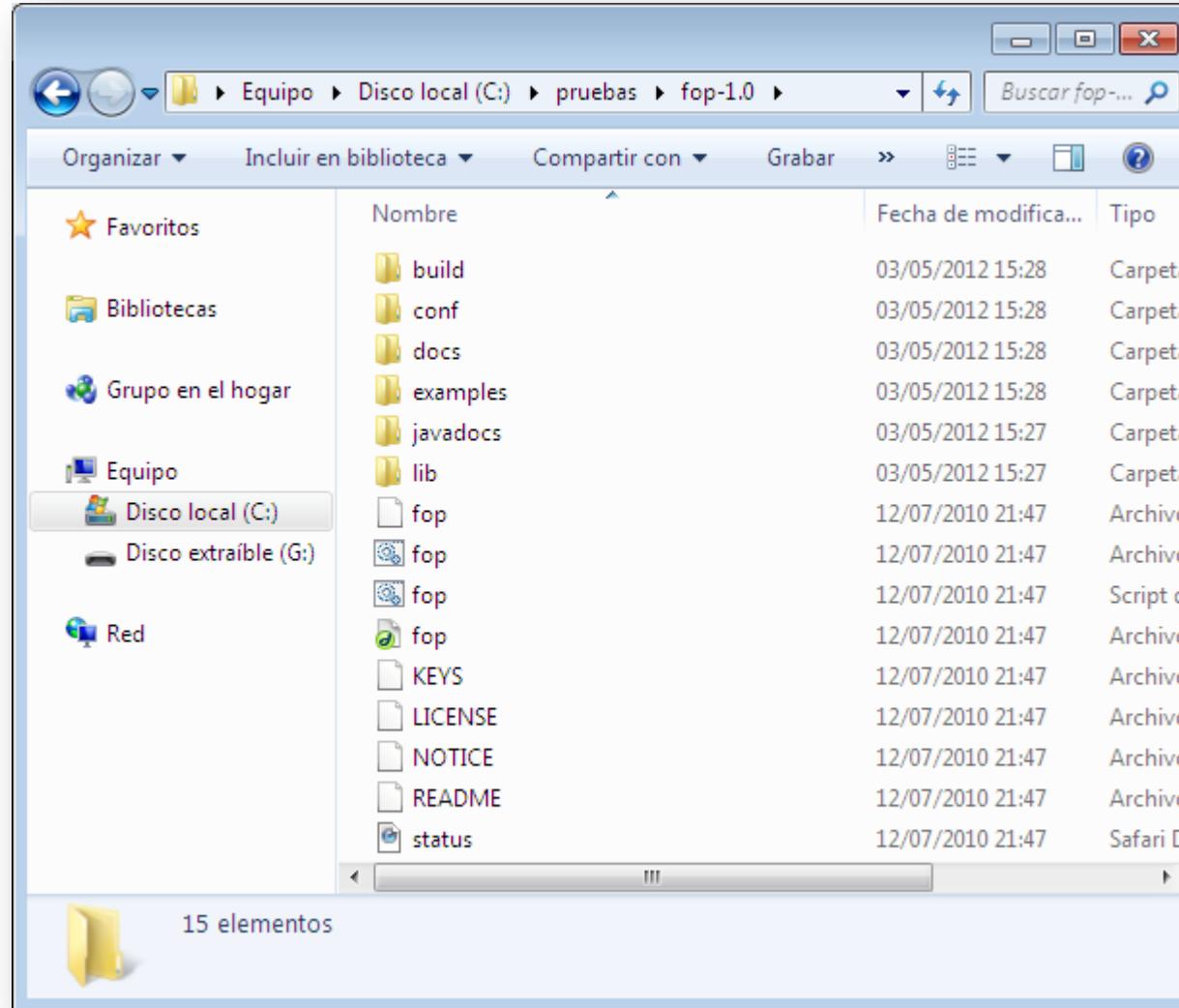
<http://xmlgraphics.apache.org/fop/download.html>



<http://xmlgraphics.apache.org/fop>

Introducción a la conversión de doc XML

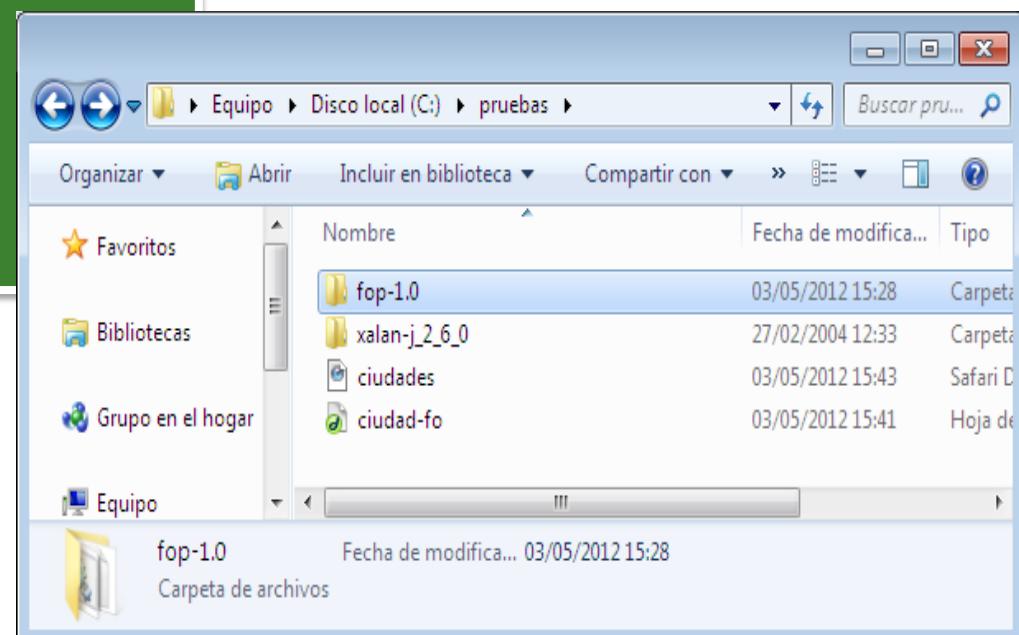
EJERCICIO GUIADO XSLT, XSL-FO y FOP



Introducción a la conversión de doc XML

EJERCICIO GUIADO XSLT, XSL-FO y FOP

```
<?xml version="1.0" encoding="UTF-8"?>
<ciudades>
    <ciudad>
        <nombre>Madrid</nombre>
        <habitantes>3500000</habitantes>
    </ciudad>
    <ciudad>
        <nombre>Malaga</nombre>
        <habitantes>800000</habitantes>
    </ciudad>
    <ciudad>
        <nombre>Toledo</nombre>
        <habitantes>50000</habitantes>
    </ciudad>
</ciudades>
```



Introducción a la conversión de doc XML

EJERCICIO GUIADO XSLT, XSL-FO y FOP

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/ciudades">
    <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
      <fo:layout-master-set>
        <fo:simple-page-master master-name="Letter" page-height="11in" page-width="8.5in">
          <fo:region-body region-name="only_region" margin="1in" background-
color="#CCCCCC"/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="Letter">
        <fo:flow flow-name="only_region">
          <fo:block text-align="center"><xsl:value-of select="ciudad/." /></fo:block>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
</xsl:stylesheet>
```

Introducción a la conversión de doc XML

EJERCICIO GUIADO XSLT, XSL-FO y FOP

```
c:\pruebas>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 8439-6514

Directorio de c:\pruebas

03/05/2012  15:52    <DIR>      .
03/05/2012  15:52    <DIR>      ..
03/05/2012  15:59           848 ciudad-fo.xsl
03/05/2012  15:43           320 ciudades.xml
03/05/2012  15:28    <DIR>      fop-1.0
03/05/2012  13:33    <DIR>      xalan-j_2_6_0
27/02/2004  13:33           2 archivos   1.168 bytes
                           4 dirs   102.145.298.432 bytes libres

c:\pruebas>java org.apache.xslt.Process -IN ciudades.xml -XSL ciudad-fo.xs
l -OUT salida.fo

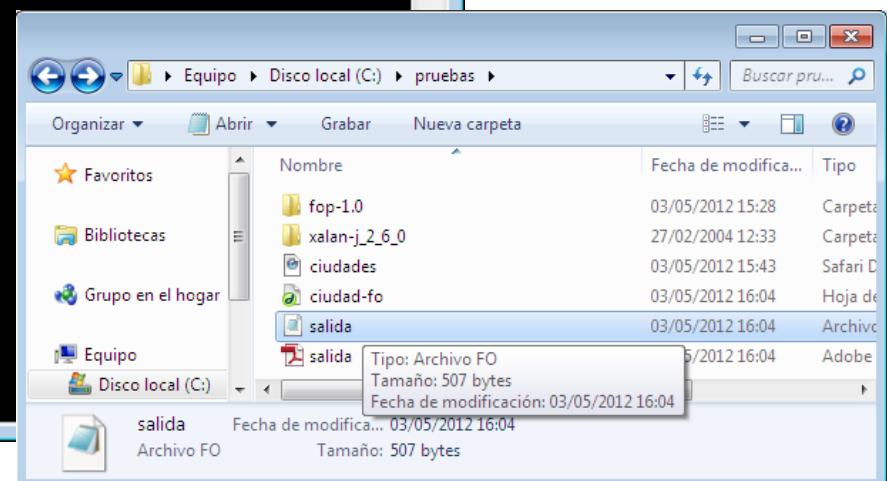
c:\pruebas>fop-1.0\fop salida.fo salida.pdf

c:\pruebas>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 8439-6514

Directorio de c:\pruebas

03/05/2012  16:00    <DIR>      .
03/05/2012  16:00    <DIR>      ..
03/05/2012  15:59           848 ciudad-fo.xsl
03/05/2012  15:43           320 ciudades.xml
03/05/2012  15:28    <DIR>      fop-1.0
03/05/2012  16:00           489 salida.fo
03/05/2012  16:00           5.092 salida.pdf
27/02/2004  13:33    <DIR>      xalan-j_2_6_0
                           4 archivos   6.749 bytes
                           4 dirs   102.145.290.240 bytes libres

c:\pruebas>_
```



Introducción a la conversión de doc XML

EJERCICIO GUIADO XSLT, XSL-FO y FOP

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set><fo:simple-page-master page-width="8.5in" page-
height="11in" master-name="Letter">
<fo:region-body background-color="#CCCCCC" margin="1in" region-
name="only_region"/>
</fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="Letter">
<fo:flow flow-name="only_region"><fo:block text-align="center">
    Madrid
    3500000
</fo:block>
</fo:flow>
</fo:page-sequence
></fo:root>
```

Introducción a la conversión de doc XML

EJERCICIO GUIADO XSLT, XSL-FO y FOP

```
c:\pruebas>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 8439-6514

Directorio de c:\pruebas

03/05/2012  15:52    <DIR>      .
03/05/2012  15:52    <DIR>      ..
03/05/2012  15:59           848 ciudad-fo.xsl
03/05/2012  15:43           320 ciudades.xml
03/05/2012  15:28    <DIR>      fop-1.0
27/02/2004  13:33    <DIR>      xalan-j_2_6_0
               2 archivos          1.168 bytes
               4 dirs   102.145.298.432 bytes libres

c:\pruebas>java org.apache.xslt.Process -IN ciudades.xml -XSL ciudad-fo.xsl -OUT salida.fo
1

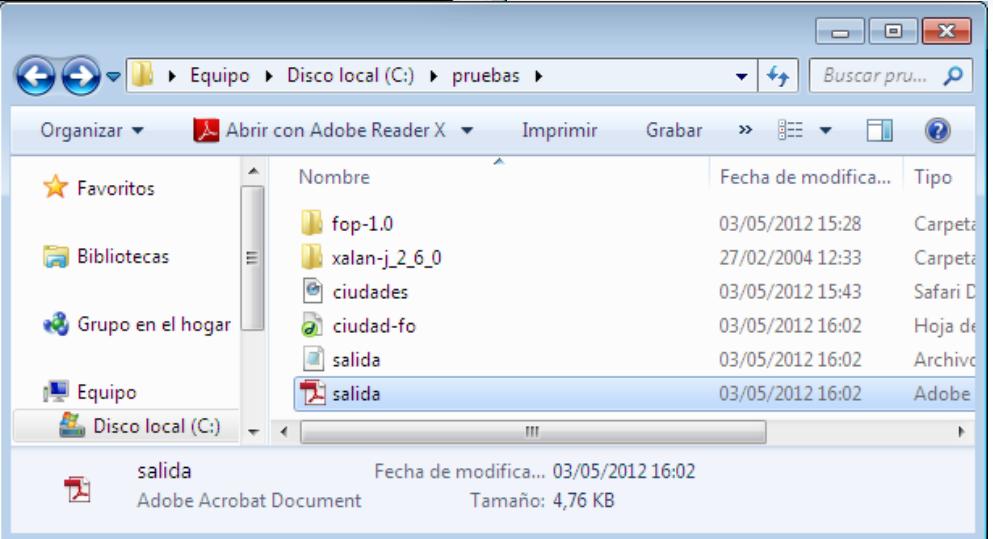
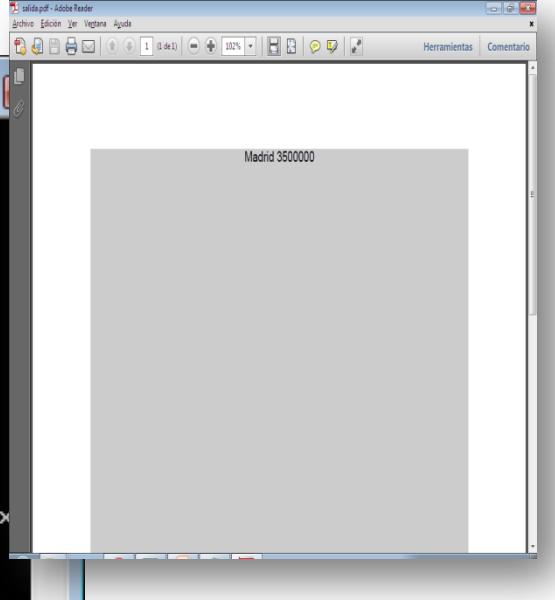
c:\pruebas>fop-1.0\fop salida.fo salida.pdf

c:\pruebas>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 8439-6514

Directorio de c:\pruebas

03/05/2012  16:00    <DIR>      .
03/05/2012  16:00    <DIR>      ..
03/05/2012  15:59           848 ciudad-fo.xsl
03/05/2012  15:43           320 ciudades.xml
03/05/2012  15:28    <DIR>      fop-1.0
03/05/2012  16:00           489 salida.fo
03/05/2012  16:00           5.092 salida.pdf
27/02/2004  13:33    <DIR>      xalan-j_2_6_0
               4 archivos          6.749 bytes
               4 dirs   102.145.290.240 bytes libres

c:\pruebas>_
```



Introducción a la conversión de doc XML

Otras Herramienta y entornos de procesamiento

- Apache Cocoon (<http://cocoon.apache.org>), usualmente llamado simplemente Cocoon, es un framework para aplicaciones web construido en torno a los conceptos de pipeline, encapsulación y desarrollo de aplicaciones web basado en componentes. El framework se enfoca en la publicación pasada en XML y XSLT, habiendo sido construido en el lenguaje de programación Java.



http://cocoon.apache.org/1284_1_1.html

Un **framework para aplicaciones web** es un *framework* diseñado para apoyar el desarrollo de sitios web dinámicos, aplicaciones web y servicios web. Este tipo de frameworks intenta aliviar el exceso de carga asociado con actividades comunes usadas en desarrollos web. Por ejemplo, muchos *framework* proporcionan bibliotecas para acceder a bases de datos, estructuras para plantillas y gestión de sesiones, y con frecuencia facilitan la reutilización de código.

un **framework** o *infraestructura digital*, es una estructura conceptual y tecnológica, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

Introducción a la conversión de doc XML

Otras Herramienta y entornos de procesamiento

- Hay muchas formas de usar las hojas de estilo. Lo más normal es usarlas dentro de un entorno de publicación tal como el Cocoon, o el IBM Transcoding Publisher, AxKit u otros por el estilo.
- Un entorno de publicación de XML permite mantener sitios completos basados en XML, y generar páginas en diferentes formatos a partir de ellos. En general, recomendamos Cocoon, que es una herramienta gratuita y Open Source basada en Java, y además una de las más avanzadas en el sector. Para una solución profesional, es mejor el IBM TP, pues forma parte del servidor de aplicaciones WebSphere, y cuenta con interfaz gráfico avanzado; el problema es el coste.



En muchos casos, lo que se necesita es aplicar hojas de estilo dentro de una aplicación, o usarlas desde línea de comandos a partir de otra aplicación o otro lenguaje de programación. En ese caso, lo más útil es usar un procesador de hojas de estilo, que habitualmente se encuentra en conjunción con un parser XML, con o sin validación. De estos, hay los siguientes:

Introducción a la conversión de doc XML

Otras Herramienta y entornos de procesamiento

- **Xalan**, que será el que más usemos en este tutorial, y que además es gratuito. La versión actual es la 2.6. Es una herramienta escrita en Java, y lo único que se necesita para hacerla funcionar es una máquina virtual Java (JVM), tal como la de Microsoft, Sun o IBM. Xalan necesita un parser XML para funcionar, tal como el Xerces, aunque el fichero correspondiente (xerces.jar) viene incluido en la distribución.
- **Saxon**, un procesador bastante rápido, también escrito en Java, con su propio parser incluido, aunque se puede usar uno externo. Hay una versión denominada Instant Saxon, un procesador rápido, y fácil de usar, sólo para Windows. Por lo pronto, es el único que implementa la versión 2.0 de XSLT.
- Hay otros muchos procesadores, tales como el **XT de James Clark**, **xsltproc** o **el Sablotron**. Todos ellos y muchos más se pueden encontrar en la página de procesadores XSLT de XMLSoftware.com. En Linux se recomienda el **xsltproc**, basado en la librería XML/XSL de gnome; es muy rápido, es un ejecutable y no hace falta instalar Java para usarlo.

XSLT xsl:avanzado

El elemento <xsl:.....>

- En las reglas XSLT, entre sus marcas de inicio y de fin, se puede incluir:
 - *Texto que se escribirá “tal cual” en el documento resultado de la transformación.*
 - *Marcas HTML o XML que se añadirán al documento resultado de la transformación.*
 - *Elementos reservados de la especificación XSLT que realizarán una acción como recuperar el valor de un elemento, ordenar los resultados, llamar a otras reglas de la hoja de estilo, etc.*

```
<?xml version="1.0"?>
<?xmlstylesheet type="text/xsl"
href="http://www.anaya.es/docs/xml/ejemplo.xsl"?>
<documento>
    <titulo>Programar ASP</titulo>
    <páginas>456</páginas>
    <anno-pub>2001</anno-pub>
</documento>
```

XSLT xsl:avanzado

El elemento <xsl:apply-templates...>

- Sirve para llamar a otras plantillas desde el interior de una plantilla. Con esta instrucción añadimos nodos a la lista de procesado. Se añaden los nodos seleccionados mediante el atributo “select”.
- La sintaxis es la siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl=http://www.w3.org/1999/XSL/Transform>
<xsl:output method="xml" version="1.0"
             encoding="UTF-8" indent="yes"/>
.....
<xsl:apply-templates select="Expresion XPath" mode="identificador">
</xsl:apply-templates>
.....
</xsl:stylesheet>
```

XSLT xsl:avanzado

El elemento <xsl:apply-templates...>

- Como ejemplo supongamos que se tiene la siguiente DTD:

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

- Un documento XML válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!—Ejemplo para XSLT -->
<!DOCTYPE Nombres SYSTEM "ejemplo_para_XSLT.dtd">
<Nombres>
<nombre>XML</nombre>
<nombre>C++</nombre>
<nombre>HTML</nombre>
<nombre>Java</nombre>
<nombre>Javascript</nombre>
</Nombres>
```

XSLT xsl:avanzado

El elemento <xsl:apply-templates...>

- Y un ejemplo de XSLT puede ser el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
<xsl:template match="/">
  <html>
    <head>Ejemplo de XSLT </head>
    <body>
      <xsl:apply-templates select="nombre"/>
    </body>
  </html>
</xsl:template>
<xsl:template match="nombre">
  <!--?¿?¿?¿?-->
</xsl:template>
</xsl:stylesheet>
```

XSLT xsl:avanzado

xsl:instrucción

- No son elementos de nivel superior; son las *instrucciones contenidas dentro de los templates* (*plantillas*).
- Indican cómo realizar el procesamiento
 - xsl:value-of es un caso simple.
- Otras instrucciones permiten realizar tratamientos condicionales, iteraciones, construcción de elementos en el árbol resultado, etc:
 - xsl:sort
 - xsl:if
 - xsl:choose, xsl:when, xsl:otherwise
 - xsl:for-each
- Otras instrucciones permiten ordenar, definir variables, parametrizar las transformaciones.. etc

XSLT xsl:avanzado

El elemento <xsl:value-of...>

- Este elemento busca en el nodo que viene indicado con la expresión Xpath del atributo “select”, e inserta el valor del atributo o elemento encontrado en el árbol de salida.
- La sintaxis es la siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/">
        <xsl:value-of select="expresion_XPath" disable-output-escaping="yes">
        </xsl:value-of>
    </xsl:template>
</xsl:stylesheet>
```

XSLT xsl:avanzado

El elemento <xsl:value-of...>

- Por ejemplo, para mostrar el valor del elemento `titulo`, que es un hijo del elemento `ejemplar`, podríamos utilizar la siguiente regla:

```
<xsl:template match="//ejemplar">  
    <xsl:value-of select=".//titulo" />  
</xsl:template>
```

El valor del atributo `select` se puede leer de la siguiente forma: “dame el valor del elemento `titulo` que es hijo del elemento que estoy procesando”. En este caso, cada uno de los elementos `ejemplar`.

Esto se indica mediante ./ expresión Xpath

XSLT xsl:avanzado

El elemento <xsl:value-of...>

- Los atributos que puede contener son lo siguientes:
 - **select** - Muestra con que debe compararse el árbol del documento XML origen. El valor de este atributo es una expresión XPath.
 - **disable-output escaping** - Hace que en el documento de salida muestre o no los caracteres “&” y “<”, en lugar de “&” o “<”. Cuando el valor de este atributo es “yes” pone en el documento de salida los caracteres “&” y “<”. En cambio si el valor de este atributo es “no” pondrá como salida “&” o “<” respectivamente.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Nombres SYSTEM "ejemplo_para_XSLT.dtd">
<Nombres>
    <nombre>&lt;XML&gt;</nombre>
    <nombre>&lt;C++&gt;</nombre>
    <nombre>&lt;HTML&gt;</nombre>
    <nombre>&lt;Java&gt;</nombre>
    <nombre>&lt;Javascript&gt;</nombre>
</Nombres>
```

```
<XML> <C++> <HTML> <Java> <Javascript>
```

```
&lt;XML&gt; &lt;C++&gt; &lt;HTML&gt; &lt;Java&gt; &lt;Javascript&gt;;
```

XSLT xsl:avanzado

El elemento <xsl:if...>

- Para indicar qué instancias de un elemento queremos procesar, o realizar una “ejecución condicional de código”, en XSLT disponemos del elemento xsl:if
- xsl:if va acompañado de un atributo test que contiene una “condición” (*lenguaje xpath*).
- Si la condición se cumple para el elemento que se está procesando, la regla de ejecutará. Por ejemplo:

```
<xsl:if test="@destino='JFK'"> <!-- atributo destino del contexto actual “elemento” -->
  <tr>
    <td><xsl:value-of select="@numero" /></td>
    <td><xsl:value-of select="@origen" /></td>
    <td><xsl:value-of select="@destino" /></td>
    <td><xsl:value-of select="@hora" /></td>
  </tr>
</xsl:if>
```

XSLT xsl:avanzado

El elemento <xsl:choose..>, <xsl:when..> y <xsl:otherwise..>

- Estos elementos “amplían” las posibilidades del elemento xsl:if.
- Permiten indicar qué transformación se debe realizar en el caso de que se cumpla una condición, y en el resto de casos.
- Se utilizan de forma conjunta. El elemento xsl:choose contendrá a uno o más elementos xsl:when y a un elemento xsl:otherwise.
- El elemento xsl:when incluye un atributo test que tomará como valor la expresión que se evaluará. Si se cumple, se ejecutará el código escrito entre las etiquetas de inicio y de fin del elemento xsl:when.
- El elemento xsl:otherwise contendrá el código que se ejecutará si no se cumplen las expresiones indicadas en los atributos test de los elementos xsl:when.

<xsl:choose>

<xsl:when test="condición">

...

</xsl:when>

<xsl:when test="condición">

...

</xsl:when>

<xsl:otherwise>

...

</xsl:otherwise>

</xsl:choose>

XSLT xsl:avanzado

El elemento <xsl:choose..>, <xsl:when..> y <xsl:otherwise..>

```
<xsl:choose>
    <xsl:when test="expresión">
        .....
    </xsl:when>
    <xsl:when test="expresión2">
        .....
    </xsl:when>
    <xsl:otherwise>
        .....
    </xsl:otherwise>
</xsl:choose>
```

```
<xsl:for-each select="¿TODAS LAS TAREAS?">
    <p>
        <xsl:value-of select="¿NOMBRE DE TAREA?"/>-
        <xsl:choose>
            <xsl:when test="@hora-fin < 12"> Por la mañana </xsl:when>
            <xsl:when test="@hora-ini > 12"> Por la tarde </xsl:when>
            <xsl:otherwise>Al mediodía</xsl:otherwise>
        </xsl:choose>
    </p>
</xsl:for-each>
```

XSLT xsl:avanzado

El elemento <xsl:for-each>

```
<xsl:for-each select="expresión xpath" order-by="elemento">  
    ? ? ? ? ? ?  
</xsl:for-each>
```

```
<xsl:for-each select="cotizacion" order-by="nombre">  
<tr>  
    <td><xsl:value-of select="nombre"/></td>  
    <td><xsl:value-of select="mercado"/></td>  
    <td><xsl:value-of select="precio"/></td>  
    <td><xsl:value-of select="fecha/dia"/>-  
        <xsl:value-of select="fecha/mes"/>-  
        <xsl:value-of select="fecha/anio"/></td>  
</tr>  
</xsl:for-each>
```

Si queremos que sea ascendente o descendente

```
<xsl:for-each select="cotizacion" order-by="-nombre">
```

XSLT xsl:avanzado

El elemento <xsl:sort..>

- Las reglas se van activando y ejecutando a medida que se recorre el documento origen que se quiere transformar.
- De esta forma, las reglas se ejecutan en el orden en el que se van encontrando los elementos en el documento.
- Este comportamiento por defecto puede cambiarse en las hojas de estilo XSLT, a diferencia de lo que sucedía en las hojas de estilo CSS
- Esto permite “reordenar” los contenidos del documento XML, de una forma distinta a como están ordenadas en el documento XML inicial.

XSLT xsl:avanzado

El elemento <xsl:sort..>

```
<xsl:sort select="expression" lang="language-code"
           data-type="text|number|qname"
           order="ascending|descending"
           case-order="upper-first|lower-first"/>
```

- Para ordenar los contenidos, se utiliza el elemento xsl:sort
- Acepta dos atributos:
 - select – que toma como valor el nombre del elemento que se va a utilizar como criterio de ordenación y
 - order – que indica si se debe utilizar un orden ascendente o descendente.

```
<xsl:apply-templates select="//ciudad">
    <xsl:sort select="ciudad" order="descending" />
</xsl:apply-templates>
```

En el ejemplo anterior, modificar la xslt para que los libros se ordenen por título ascendente

XSLT xsl:avanzado

El elemento @atributo

- En XSLT podemos “filtrar” o indicar qué instancias de un elemento queremos procesar, tomando como criterio de selección el valor de los atributos que acompañan a los elementos
- Para hacer esto, en un elemento xsl:value-of, podemos recuperar el valor de un atributo mediante la expresión `@nombreAtributo` , por ejemplo:

```
<xsl:template match="vuelo">
    <tr>
        <td><xsl:value-of select="@numero" /></td>
        <td><xsl:value-of select="@origen" /></td>
        <td><xsl:value-of select="@destino" /></td>
        <td><xsl:value-of select="@hora" /></td>
    </tr>
</xsl:template>
```

XSLT xsl:avanzado

El elemento <xsl:variable..>

- El elemento xsl:variable se utiliza para declarar una variable. Las variables nos permiten realizar operaciones con los datos del documento XML para luego mostrar el resultado en el documento “resultado”. Es importante señalar que cuando se le asigna un valor, éste ya no se puede cambiar
- Para declarar una variable, se utilizará la sintaxis:

```
<xsl:variable name="nombre_de_la_variable" select="expresionXPath"/>
```

Como por ejemplo:

```
<xsl:variable name="var" select="15" />
```

```
<xsl:variable name="totalPrecio" select="sum(//total)" />
```

XSLT xsl:avanzado

El elemento <xsl:variable..>

```
<?xml version="1.0" encoding="UTF-8"?>
<pedido>
    <cliente>
        <nombre>Construcciones Barcelona</nombre>
        <domicilio>Gran Via 45, 2º</domicilio>
        <localidad>Barcelona</localidad>
    </cliente>
    <detalle>
        <item>
            <material>Tornillos-5</material>
            <unidades>10000</unidades>
            <precio>3</precio>
            <total>30000</total>
        </item>
        <item>
            <material>Paletas</material>
            <unidades>100</unidades>
            <precio>500</precio>
            <total>50000</total>
        </item>
        <item>
            <material>Ladrillos</material>
            <unidades>600</unidades>
            <precio>23</precio>
            <total>13800</total>
        </item>
    </detalle>
</pedido>
```

XSLT xsl:avanzado

El elemento <xsl:variable..>

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/pedido">
        <html>
            <head><title>Pedido</title></head>
            <body>
                <xsl:apply-templates />
            </body></html>
        </xsl:template>
        <xsl:template match="detalle">
            <table width="85%">
                <tr>
                    <th>Material</th>
                    <th>Unidades</th>
                    <th>Precio</th>
                    <th>Total Pts.</th>
                </tr>
                <xsl:for-each select="item">
                    <tr>
                        <td><xsl:value-of select="material" /></td>
                        <td><xsl:value-of select="unidades" /></td>
                        <td><xsl:value-of select="precio" /></td>
                        <td><xsl:value-of select="total" /></td>
                    </tr>
                </xsl:for-each>
            </table>
        </xsl:template>
    </xsl:stylesheet>
```

XSLT xsl:avanzado

El elemento <xsl:variable..>

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="totalPrecio" select="sum(//total)" />
<xsl:template match="/">
    <html>
        <head><title>Pedido</title></head>
        <body>
            <xsl:apply-templates />
        </body></html>
    </xsl:template>
    <xsl:template match="detalle">
        <table width="85%">
            <tr>
                <th>Material</th>
                <th>Unidades</th>
                <th>Precio</th>
                <th>Total Pts.</th>
            </tr>
            <xsl:for-each select="item">
                <tr>
                    <td><xsl:value-of select="material" /></td>
                    <td><xsl:value-of select="unidades" /></td>
                    <td><xsl:value-of select="precio" /></td>
                    <td><xsl:value-of select="total" /></td>
                </tr>
            </xsl:for-each>
        </table>
        <h4>Total a pagar: <xsl:value-of select="$totalPrecio" /></h4>
    </xsl:template>
</xsl:stylesheet>
```

XSLT xsl:avanzado

El elemento <xsl:param ..> PARAMETRIZACIÓN XSLT

- Si sólo se pudieran generar páginas estáticas, poco habríamos ganado con respecto al HTML; la gracia del binomio XML/XSLT es que se puede generar contenido diferente dependiendo de las entradas. Para ello debe haber alguna forma de pasarle parámetros a las hojas, y eso se hace, cómo si no, con la orden xsl:param.
- A la vez, teniendo en cuenta las entradas, habrá que tomar decisiones; o quizás dependiendo de los atributos.

```
<xsl:param name='nombre_parámetro' />
```

XSLT xsl:param avanzado

El elemento <xsl:param ..> PARAMETRIZACIÓN XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<libros>
  <libro>
    <titulo>Fuente Ovejuna</titulo>
    <autor>Lope de Vega</autor>
    <isbn>84-9815-002-7</isbn>
    <precio>24</precio>
  </libro>
  <libro>
    <titulo>La Celestina</titulo>
    <autor>Fernando de Rojas</autor>
    <isbn>84-96390-96-9</isbn>
    <precio>32</precio>
  </libro>
</libros>

<libro>
  <titulo>Fuente Ovejuna</titulo>
  <autor>Lope de Vega</autor>
  <isbn>84-9815-002-7</isbn>
  <precio>56</precio>
</libro>
<libro>
  <titulo>Fuente Ovejuna</titulo>
  <autor>Lope de Vega</autor>
  <isbn>84-9815-002-7</isbn>
  <precio>22</precio>
</libro>
</libros>
```

XSLT xsl:avanzado

El elemento <xsl:param ..> PARAMETRIZACIÓN XSLT

- La condición de nuestra hoja de transformación será el generar un listado de libros, pero siempre y cuando, estos, tengan un precio superior al valor proporcionado como parámetro.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:param name="paraPrecio"/>
<xsl:template match="/">
<html>
<body>
<h1>Mis Libros</h1>
<xsl:for-each select="libros/libro">
    <xsl:if test="precio > $paraPrecio ">
        Titulo:<xsl:value-of select="titulo"/><br/>
        Autor:<xsl:value-of select="autor"/><br/>
        Precio:<xsl:value-of select="precio"/><br/>
    </xsl:if>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

XSLT xsl:avanzado

El elemento <xsl:param ..> PARAMETRIZACIÓN XSLT

- Para usar esta hoja pasándole parámetros desde la línea de órdenes, habrá que hacerlo así (usando saxon):

```
java com.icl.saxon.StyleSheet tiendecilla.xml tiendecilla-html-1.xls  
seccion='Muebles'
```

o bien, usando Xalan:

```
java org.apache.xalan.xslt.Process -IN libros.xml -XSL libros-html.xls -  
PARAM paraPrecio 50 -OUT libros.html
```

Si se usa la hoja de estilo dentro de un entorno de publicación como COCOON, habrá que pasarle los parámetros al documento XML, bien directamente desde el URL o desde un formulario, algo así:
[http://misitio.com/libros.xml?paraPrecio=50.](http://misitio.com/libros.xml?paraPrecio=50)



XSLT xsl:param avanzado

El elemento <xsl:param ..> PARAMETRIZACIÓN XSLT

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <libros>
  - <libro>
    <titulo>Fuente Ovejuna</titulo>
    <autor>Lope de Vega</autor>
    <isbn>84-9815-002-7</isbn>
    <precio>24</precio>
  </libro>
  - <libro>
    <titulo>La Celestina</titulo>
    <autor>Fernando de Rojas</autor>
    <isbn>84-96390-96-9</isbn>
    <precio>32</precio>
  </libro>
  - <libro>
    <titulo>Fuente Ovejuna</titulo>
    <autor>Lope de Vega</autor>
    <isbn>84-9815-002-7</isbn>
    <precio>56</precio>
  </libro>
  - <libro>
    <titulo>Fuente Ovejuna</titulo>
    <autor>Lope de Vega</autor>
    <isbn>84-9815-002-7</isbn>
    <precio>22</precio>
  </libro>
</libros>
```

XSLT xsl:param avanzado

El elemento <xsl:param ..> PARAMETRIZACIÓN XSLT

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:param name="paraPrecio"/>
  - <xsl:template match="/">
    - <html>
      - <body>
        <h1>Mis Libros</h1>
        - <xsl:for-each select="libros/libro">
          - <xsl:if test="precio > $paraPrecio ">
            Titulo:
            <xsl:value-of select="titulo"/>
            <br/>
            Autor:
            <xsl:value-of select="autor"/>
            <br/>
            Precio:
            <xsl:value-of select="precio"/>
            <br/>
          </xsl:if>
        </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

XSLT xsl:param avanzado

El elemento <xsl:param ..> PARAMETRIZACIÓN XSLT

```
c:\pruebas>dir
El volumen de la unidad C no tiene etiqueta.
El n mero de serie del volumen es: 8439-6514

Directorio de c:\pruebas

02/05/2012 12:28    <DIR>      .
02/05/2012 12:28    <DIR>      ..
02/05/2012 12:27          548 libros-html.xsl
02/05/2012 12:24          670 libros.xml
27/02/2004 13:33    <DIR>      xalan-j_2_6_0
                2 archivos        1.218 bytes
                3 dirs   101.867.003.904 bytes libres

c:\pruebas>java org.apache.xalan.xslt.Process -IN libros.xml -XSL libros-html.xs
l -PARAM paraPrecio 50 -OUT libros.html

c:\pruebas>dir
El volumen de la unidad C no tiene etiqueta.
El n mero de serie del volumen es: 8439-6514

Directorio de c:\pruebas

02/05/2012 12:29    <DIR>      .
02/05/2012 12:29    <DIR>      ..
02/05/2012 12:27          548 libros-html.xsl
02/05/2012 12:29          124 libros.html
02/05/2012 12:24          670 libros.xml
27/02/2004 13:33    <DIR>      xalan-j_2_6_0
                3 archivos        1.342 bytes
                3 dirs   101.867.003.904 bytes libres

c:\pruebas>_
```

Mis Libros

Titulo:Fuente Ovejuna
Autor:Lope de Vega
Precio:56

XSLT xsl:avanzado

El elemento <xsl:copy-of ..>

- Se utiliza para copiar un conjunto de nodos del documento origen, al documento resultado de la transformación.
- Se copiarán todos los nodos hijos y los atributos (en el caso de los elementos que los tengan).
- Este elemento es especialmente útil cuando se quiere convertir un documento XML a otro documento XML con una estructura diferente.
- El elemento xsl:copy-of irá acompañado por un atributo select que toma como valor una expresión que determinará los nodos que se van a copiar.
- Este elemento también se puede utilizar para copiar en el documento resultado el valor de una variable. En este caso, se escribirá como valor del atributo select el nombre de la variable precedido por el carácter \$.

XSLT xsl:avanzado

El elemento <xsl:copy-of ..>

```
<?xml version="1.0" encoding="UTF-8"?>
<?xmlstylesheet type="text/xsl" href="dlibros3.xsl"?>
<repertorio>
    <libro>
        <titulo>Don Quijote de la Mancha</titulo>
        <autor>Miguel de Cervantes</autor>
        <anno-pub>1987</anno-pub>
        <isbn>84-568-94-3</isbn>
    </libro>
    <libro>
        <titulo>La Galatea</titulo>
        <autor>Miguel de Cervantes</autor>
        <anno-pub>1989</anno-pub>
        <isbn>84-568-9424</isbn>
    </libro>
    <libro>
        <titulo>La Celestina</titulo>
        <autor>Fernando de Rojas</autor>
        <anno-pub>1998</anno-pub>
        <isbn>84-568-95-12</isbn>
    </libro>
</repertorio>
```

XSLT xsl:avanzado

El elemento <xsl:copy-of ..>

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/">
        <repertorio>
            <xsl:copy-of select="//libro[starts-with(autor, 'Miguel de Cervantes')]" />
        </repertorio>
    </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<repertorio>
    <libro>
        <titulo>Don Quijote de la Mancha</titulo>
        <autor>Miguel de Cervantes</autor>
        <anno-pub>1987</anno-pub>
        <isbn>84-568-94-3</isbn>
    </libro>
    <libro>
        <titulo>La Galatea</titulo>
        <autor>Miguel de Cervantes</autor>
        <anno-pub>1989</anno-pub>
        <isbn>84-568-9424</isbn>
    </libro>
</repertorio>
```

XSLT xsl:avanzado

El elemento <xsl:copy ..>

- Similar al elemento anterior, se utiliza para copiar elementos, pero no se copiarán sus atributos ni sus elementos hijos
- Cuando se aplica sobre elementos, se copia el elemento, pero no su valor.
- Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/">
        <repertorio>
            <xsl:apply-templates select="//autor" />
        </repertorio>
    </xsl:template>
    <xsl:template match="autor">
        <xsl:copy />
    </xsl:template>
</xsl:stylesheet>
```

XSLT xsl:avanzado

El elemento <xsl:copy..>

- En el ejemplo anterior, se crea un elemento autor vacío en el documento destino, para cada elemento autor existente en el documento original
- Para copiar el valor de los elementos autor, habría que modificar la XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" />
    <xsl:template match="/">
        <repertorio>
            <xsl:apply-templates select="//autor" />
        </repertorio>
    </xsl:template>
    <xsl:template match="autor">
        <xsl:copy>
            <xsl:value-of select=". " />
        </xsl:copy>
    </xsl:template>
</xsl:stylesheet>
```

XSLT xsl:avanzado

El elemento <xsl:element ..>

- Se utiliza para crear elementos en el documento resultado de la transformación.
- Es especialmente útil cuando se utiliza XSLT para transformar un documento XML en otro con una estructura diferente.
- xsl:element irá acompañado por un atributo name que tomará como valor el nombre del elemento que se va a crear.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <body>
      <h1>Mis Libros</h1>
      <xsl:element name="select">
        <xsl:for-each select="libros/libro">
          <xsl:element name="option">
            <xsl:attribute
              name="value"> <xsl:value-of select="titulo"/>
            </xsl:attribute>
            <xsl:value-of
              select="titulo"/>
          </xsl:element>
        </xsl:for-each>
        </xsl:element>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

XSLT xsl:avanzado

El elemento <xsl:attribute ..>

- Permite crear un atributo en el documento resultado de la transformación.
- Irá acompañado por un atributo name, que recogerá el nombre del atributo.

```
<xsl:element name="option">
  <xsl:attribute name="value">
    <xsl:value-of select="titulo"/>
  </xsl:attribute>
  <xsl:value-of select="titulo"/>
</xsl:element>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h1>Mis Libros</h1>
<xsl:element name="select">
<xsl:for-each select="libros/libro">
  <xsl:element name="option">
    <xsl:attribute
      name="value"> <xsl:value-of select="titulo"/>
    </xsl:attribute>
    <xsl:value-of
      select="titulo"/>
  </xsl:element>
</xsl:for-each>
</xsl:element>
....
```

XSLT xsl:avanzado

El elemento <xsl:element ..> <xsl:attribute ..>

```
<BODY BGCOLOR="#00FFFF">
```

```
  <P>Esto es una prueba</P>
```

```
</BODY>
```



```
<xsl:element name="BODY">
  <xsl:attribute name="BGCOLOR">
    <xsl:value-of select="@color">
  </xsl:attribute>
  <xsl:element name="P">
    <xsl:value-of select="parrafo">
  </xsl:element>
</xsl:element>
```

XSLT xsl:avanzado

El elemento <xsl:comment>

- Este elemento se utilizará para crear un comentario en el documento resultado de la transformación.
- El elemento xsl:comment contendrá el texto del comentario, sin las marcas <!-- y -->

```
<xsl:comment>
    Comentario
</xsl:comment>
```

XSLT xsl:avanzado

El elemento <xsl:processing-instruction..>

- Se utiliza para crear una instrucción de procesamiento en el documento resultado de la transformación.
- Debe ir acompañado por un atributo name, que es obligatorio, y que toma como valor el nombre de la instrucción de procesamiento.
- Entre sus etiquetas de inicio y de fin se escribirán los calificadores de la instrucción de procesamiento, entre las marcas <xsl:text> y </xsl:text>.

Ejemplo

El siguiente código crearía una instrucción de procesamiento en el documento destino:

```
<xsl:template match="/">
  <xsl:processing-instruction name="xmlstylesheet">
    <xsl:text>type="text/xsl" href="hojaEstilo.xsl"</xsl:text>
  </xsl:processing-instruction>
</xsl:template>
```

XSLT xsl:avanzado

El elemento <xsl:processing-instruction..>

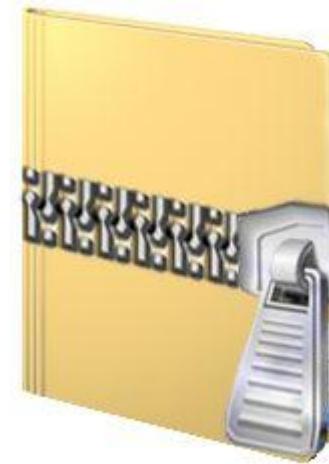
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<xsl:processing-instruction name="xml-stylesheet">
<xsl:text>type="text/xsl" href=" nombre_ciudades.xsl" </xsl:text>
</xsl:processing-instruction>
</xsl:template>
.....Pasar de un XML a otro XML.....
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="_____ .xsl"?>
<ciudades>
<ciudad>
<nombre>Madrid</nombre>
<habitantes>3500000</habitantes>
</ciudad>
<ciudad>
<nombre>Málaga</nombre>
<habitantes>800000</habitantes>
</ciudad>
</ciudades>
```

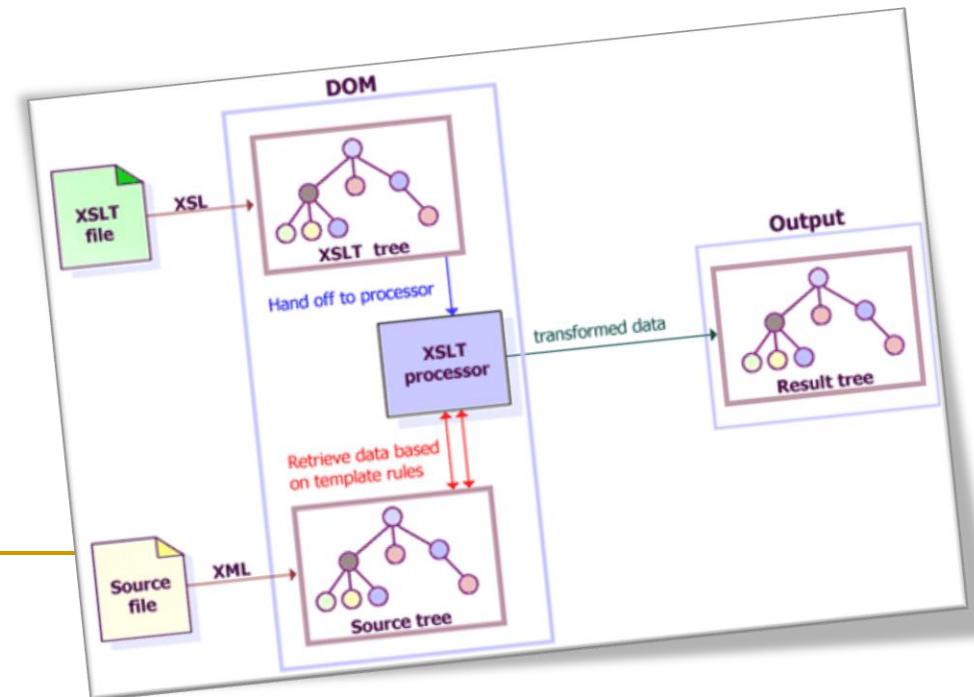
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="nombre_ciudades.xsl"?>
<poblaciones>
<ciudad>
<nombre>Madrid</nombre>
</ciudad>
<ciudad>
<nombre>Málaga</nombre>
</ciudad>
</poblaciones>
```

XSLT xsl:avanzado

Ejemplos



Ejemplo de Examen
View FireFox



Xpath: XML Path Language

Introducción

- *Todo el procesamiento realizado con un fichero XML está basado en la posibilidad de direccionar o acceder a cada una de las partes que lo componen, de modo que podamos tratar cada uno de los elementos de forma diferenciada.*
- *La forma de seleccionar información dentro de él es mediante el uso de XPath, que es la abreviación de lo que se conoce como XML Path Language. Con XPath podremos seleccionar y hacer referencia a texto, elementos, atributos y cualquier otra información contenida dentro de un fichero XML.*
- *XPath sirve para decir cómo debe procesar una hoja de estilo el contenido de una página XML, pero también para poder poner enlaces o cargar en un navegador zonas determinadas de una página XML, en vez de toda la página.*
- *XPath en sí es un lenguaje sofisticado y complejo, de tipo declarativo, pero distinto de los lenguajes procedurales que solemos usar (C,C++, Basic, Java...).*

Xpath: XML Path Language

W3C Recommendation 16 November 1999

W3C Recommendation



<http://www.w3.org/TR/xpath/>

XML Path Language (XPath) Version 1.0

W3C Recommendation 16 November 1999

This version:

<http://www.w3.org/TR/1999/REC-xpath-19991116>

(available in [XML](#) or [HTML](#))

Latest version:

<http://www.w3.org/TR/xpath>

Previous versions:

<http://www.w3.org/TR/1999/PR-xpath-19991008>

<http://www.w3.org/1999/08/WD-xpath-19990813>

<http://www.w3.org/1999/07/WD-xpath-19990709>

<http://www.w3.org/TR/1999/WD-xslt-19990421>

Editors:

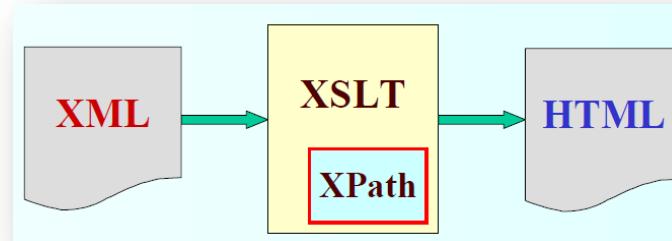
James Clark <[jjc@jclark.com](mailto:jtc@jclark.com)>

Steve DeRose (Inso Corp. and Brown University) <Steven_DeRose@Brown.edu>

Xpath: XML Path Language

Introducción

- XPath es una especificación del W3C (aprobada el mismo día que XSLT).
- Define cómo acceder a partes de un documento XML.
- 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: /fecha/dia
- XPath se usa en XSLT, pero también en XSL-FO, XPointer, XLink, y otros.



- En XSLT, XPath se utiliza en los valores de atributos (atributos tales como **match** o **select**, empleados en las etiquetas **xsl:value-of** y **xsl:template** respectivamente).

Xpath: XML Path Language

Modelo de datos de Xpath

- Un documento XML es procesado por un analizador (o parser) construyendo un árbol de nodos.
 - Este árbol comienza con un elemento raíz, que se diversifica a lo largo de los elementos que cuelgan de él y acaba en nodos hoja, que contienen sólo texto, comentarios, instrucciones de proceso o incluso que están vacíos y sólo tienen atributos.
- El árbol de nodos:
 - Comienza en el nodo raíz
 - Acaba en los nodos hoja
- **XPath** selecciona partes del documento XML basándose en la estructura en árbol.

Xpath: XML Path Language

Modelo de datos de Xpath

```
/  
|  
+---libro  
|   |  
+---titulo  
|   |  
+---(texto)Dos por tres calles  
  
+---autor  
|   |  
+---(texto)Josefa Santos  
  
+---capitulo [num=1]  
|   |  
+---(texto)La primera calle  
|   |  
+---parrafo  
|   |  
+---(texto)Era una sombría noche ...  
+---parrafo  
|   |  
+---(texto)Ella, cual inocente mariposa...  
  
+---capitulo [num=2]
```

```
<libro>  
  <titulo>Dos por tres calles</titulo>  
  <autor>Josefa Santos</autor>  
  <capitulo num="1">  
    La primera calle  
    <parrafo>  
      Era una sombría noche del mes de agosto...  
    </parrafo>  
    <parrafo destacar="si">  
      Ella, inocente cual  
      <enlace href="http://www.enlace.es">mariposa</enlace>  
      que surca el cielo en busca de libaciones...  
    </parrafo>  
  </capitulo>
```

Xpath: XML Path Language

Tipos de nodos

■ **Nodo Raíz**

- Se identifica por ***/***. No se debe confundir el nodo raíz con el elemento raíz del documento.
 - El documento XML de nuestro ejemplo tiene por elemento raíz a libro, éste será el primer nodo que cuelgue del nodo raíz del árbol, el cual es: ***/***.

■ **Nodo Elemento**

- Cualquier elemento de un documento XML se convierte en un nodo elemento dentro del árbol.
- Cada elemento tiene su nodo padre. El nodo padre de cualquier elemento es, a su vez, un elemento, excepto el elemento raíz, cuyo padre es el nodo raíz.
- Los nodos elemento tienen a su vez hijos, que puede ser nodos de cualquier tipo excepto raíz.

Xpath: XML Path Language

Tipos de nodos

■ **Nodos Texto**

- ❑ referencia a todos los caracteres del documento que no está marcados con alguna etiqueta.

■ **Nodo Atributo**

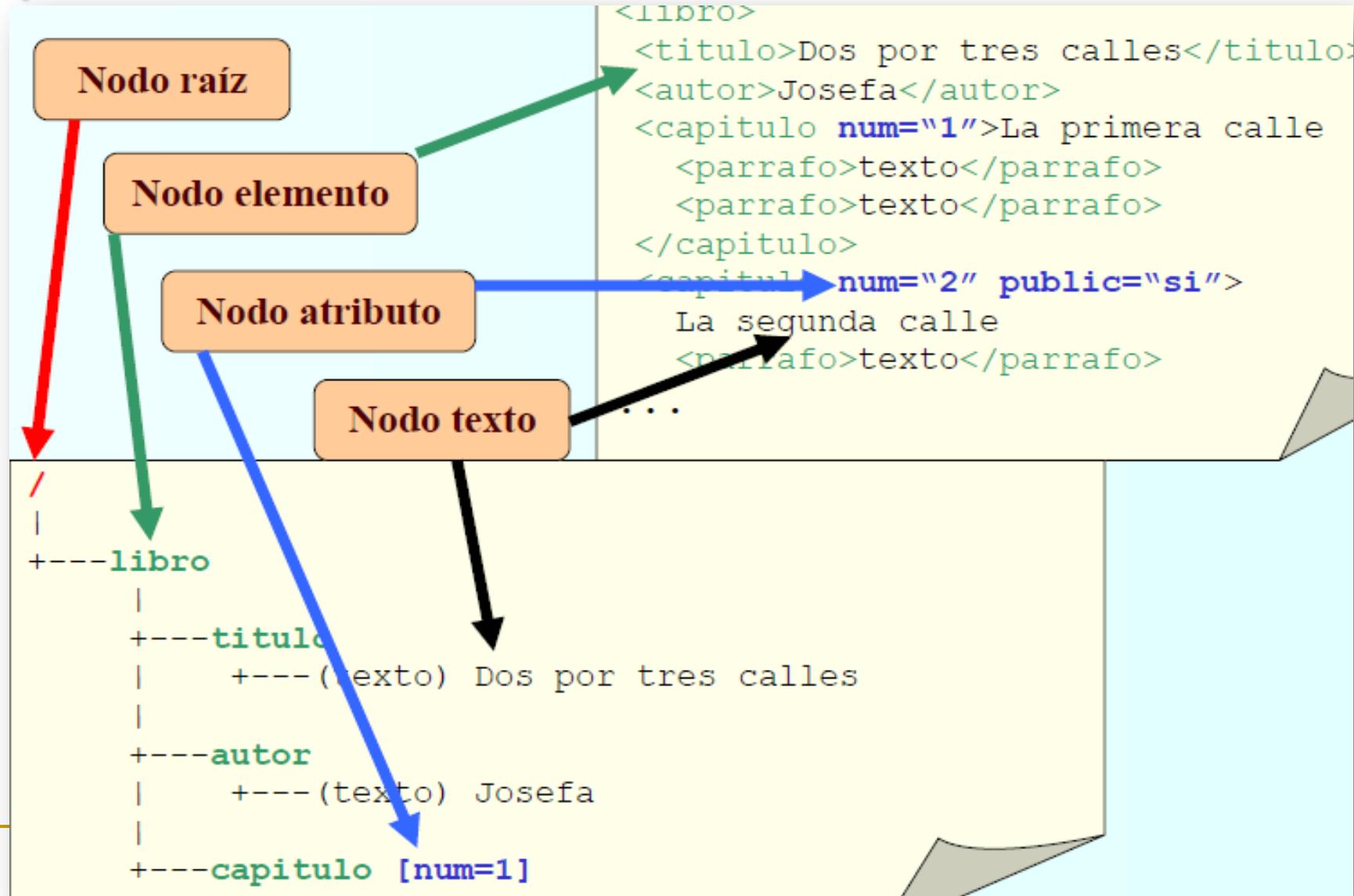
- ❑ Más que hijos del nodo elemento que los contiene son como etiquetas añadidas a dicho nodo.
- ❑ Cada nodo atributo consta de un nombre, un valor (que es siempre una cadena).

■ **Nodos comentario, y de Instrucciones de proceso.**

- ❑ Al contenido de estos nodos se puede acceder con la propiedad string-value.

Xpath: XML Path Language

Tipos de nodos



Xpath: XML Path Language

Expresiones Xpath

Una "instrucción" en lenguaje XPath se denomina **expresión**.

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

Lo que devuelve es procesado por la regla XSL.

Las expresiones XPath se usan sobre todo en los atributos match, select (y test).

Xpath: XML Path Language

Expresiones Xpath

- La evaluación de una expresión Xpath proporciona 4 posibles tipos de resultados: **conjunto de nodos (node-set)**, **booleano**, **número**, ó **cadena**.
- Tokens válidos en una expresión XPath
 - Paréntesis y similares: () { } []
 - Elemento actual . y elemento padre ..
 - Atributo @, * y separador ::
 - La 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)

Xpath: XML Path Language

Location Paths

Un **location path** es la más importante de los tipos de expresiones que se pueden especificar en notación XPath.

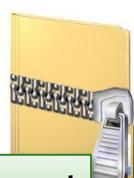
La sintaxis de un location path es similar a la usada a la hora de describir los directorios que forman una unidad de disco en Unix.

Xpath: XML Path Language

Location Paths

- Una ruta de directorio de Linux indica la ruta a un archivo particular:
 - Ejemplo: /home/juan/documentos
 - Referencia a un único directorio llamado *documentos* que cuelga de */home/juan*
- Location Paths se corresponde con la idea intuitiva de “ruta de directorio”, pero un location path siempre devuelve un *node-set*:
 - **XPath indica la ruta hasta varios nodos, basándose en la estructura del documento XML**
 - Ejemplo: /libro/capitulo/parrafo
 - Referencia a **todos** los elementos *parrafo* que cuelguen de cualquier *capitulo* del *libro*

```
<libro>
  <titulo>Dos por tres calles</titulo>
  <autor>Josefa Santos</autor>
  <capitulo num="1">
    La primera calle
    <parrafo>
      Era una sombría noche del mes de diciembre
    </parrafo>
    <parrafo destacar="si">
      Ella, inocente cual
      <enlace href="http://www.enlace.es">
        que surca el cielo en busca de lil
      </enlace>
    </parrafo>
  </capitulo>
  <capitulo num="2" public="si">
    La segunda calle
    <parrafo>
      Era una obscura noche del mes de diciembre
    </parrafo>
    <parrafo>
      Ella, inocente cual
      <enlace href="http://www.abejilla.com">
        que surca el viento en busca del nido
      </enlace>
    </parrafo>
  </capitulo>
```



Xpath: XML Path Language

Location Paths: Sintaxis

- Si se indica el camino completo, la búsqueda comienza en el nodo raíz.

Ejemplo:

- /libro/capitulo/parrago
 - Al leer / se selecciona el nodo raíz como nodo contexto.
 - Al leer **libro** se selecc. los elem libro que cuelgan del contexto (/)
 - Al leer **capitulo** se selecc. los elem. capitulo que cuelgan del contexto (en ese momento es **libro**)
 - Al leer **parrago** se selecc los elem parrago que cuelgan del contexto (en ese momento es **capitulo**)

Xpath: XML Path Language

Location Paths: Sintaxis

XML validator, XSLT transformer and Simple XPath online tester

Experimental: you may paste part of XML and try the Fix button

XPath:

output to own window

XML:

```
<?xml version="1.0"?>
<!--<?xmlstylesheet type="text/xsl" href="evalua_xpath.xsl"?>-->
<libro>
    <titulo>Dos por tres calles</titulo>
    <autor>Josefa Santos</autor>
    <capitulo num="1">
        La primera calle
        <parrafo>
            Era una sombría noche del mes de agosto...
        </parrafo>
        <parrafo destacar="si">
            Ella, inocente cual
            <enlace href="http://www.enlace.es">mariposa</enlace>
            que surca el cielo en busca de libaciones...
        </parrafo>
    </capitulo>
    <capitulo num="2" public="si">
        La segunda calle
        <parrafo>
            Era una obscura noche del mes de septiembre...
        </parrafo>
        <parrafo>
            Ella, inocente cual
            <enlace href="http://www.abejilla.es">abejilla</enlace>
            que surca el viento en busca del néctar de las flores...
        </parrafo>
    </capitulo>
    <apendice num="a" public="si">
        La tercera calle
        <parrafo>
```

Test! **Format!** **Validate!** **Fix!** **Color!** **Save!** **Transform!**

feedback

[http://www>xpathtester.com/test](http://www.xpathtester.com/test)

XML validator, XSLT transformer and Simple XPath online tester

Experimental: you may paste part of XML and try the Fix button

XPath:

output to own window

XML:

```
/libro/capitulo/parrafo
```

Test! **Format!** **Validate!** **Fix!** **Color!** **Save!** **Transform!**

feedback

```
<?xml version="1.0" encoding="UTF-8"?>

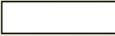
<root>
    <parrafo>Era una sombría noche del mes de agosto...</parrafo>
    <parrafo destacar="si">Ella, inocente cual
        <enlace href="http://www.enlace.es">mariposa</enlace> que surca el cielo en busca de libaciones...
    </parrafo>
    <parrafo>Era una obscura noche del mes de septiembre...</parrafo>
    <parrafo>Ella, inocente cual
        <enlace href="http://www.abejilla.es">abejilla</enlace> que surca el viento en busca del néctar de las flores...
    </parrafo>
</root>
```

Xpath: XML Path Language

Location Paths: Sintaxis

www.zvon.org/comp/r/tut-XPath_1.html#intro

 ZVON.org

  + Google

⇒ interactive index to zvon materials

 [home](#)

Follow Zvon:
  

Recommend this page at:
      

Contact:
zvon.org
@gmail.com

[About Zvon](#)

 New note app for iPhone

XPath 1.0 Tutorial

[Back](#) | [Forward](#) | [Previous](#) | [Next](#)

XPath is described in [XPath 1.0 standard](#). In this tutorial selected XPath features are demonstrated on many examples.

Standard excerpt:

XPath is the result of an effort to provide a common syntax and semantics for functionality shared between XSL Transformations and XPointer. The primary purpose of XPath is to address parts of an XML document. In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values. XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax. XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document.

Zvon offers other [XPath related materials](#).

Prepared by: Miloslav Nic (Mila)

[Previous](#) | [Next](#)

 For a flurry of SEO tips, tricks, articles and advice - visit the HP SEO blog.

Pages (23)
Keywords (34)

filter:
 enable regexp (?)

[First](#) [Prev](#) [Next](#)
1 - 20 **filter: off (23)**

List of XPaths
XPath as filesystem addressing
Start with //
All elements: *
Further conditions inside []
Attributes
Attribute values
Nodes counting
Playing with names of selected elements
Length of string
Combining XPaths with |
Child axis
Descendant axis
Parent axis
Ancestor axis
Following-sibling axis

http://www.zvon.org/comp/r/tut-XPath_1.html

Xpath: XML Path Language

Location Paths: Sintaxis

The screenshot shows the w3schools.com website. At the top, there's a navigation bar with links for HOME, HTML, CSS, XML, JAVASCRIPT, ASP, PHP, SQL, MORE..., REFERENCES, and EXAM. A search bar is also present. On the left, there's a sidebar with a 'TOP 100:' section listing Free Website Templates, Free CSS Templates, Free HTML Templates, and Free Web Templates. Below that is the 'XPath Tutorial' section with links to XPath HOME, Intro, Nodes, Syntax, Axes, Operators, Examples, and Summary. Further down is the 'XPath Reference' section with a link to XPath Functions. The main content area features a large 'XPath Tutorial' heading, a 'iStockphoto' logo, and a banner with the text 'CONTENIDO CREATIVO PARA TUS IDEAS'. A green 'XPath' logo is at the bottom of the page. The central text area describes XPath as a tool for navigating XML documents and mentions its role in XSLT and XQuery. It ends with a red link 'Start learning XPath now!'

TOP 100:
Free Website Templates
Free CSS Templates
Free HTML Templates
Free Web Templates

XPath Tutorial

XPath HOME
XPath Intro
XPath Nodes
XPath Syntax
XPath Axes
XPath Operators
XPath Examples
XPath Summary

XPath Reference

XPath Functions

XPath Tutorial

« W3Schools Home

Next Chapter »

XPath is used to navigate through elements and attributes in an XML document.

XPath is a major element in W3C's XSLT standard - and XQuery and XPointer are both built on XPath expressions.

[Start learning XPath now!](#)

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

http://www.w3schools.com>xpath>xpath_examples.asp

Xpath: XML Path Language

Location Paths: Sintaxis

http://www.w3schools.com>xpath>xpath_examples.asp/

Edit and Click Me »

```
<html>
<body>
<script type="text/javascript">
function loadXMLDoc(dname)
{
if (window.XMLHttpRequest)
{
 xhttp=new XMLHttpRequest();
}
else
{
 xhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xhttp.open("GET",dname,false);
xhttp.send("");
return xhttp.responseXML;
}

xml=loadXMLDoc("books.xml");
path="/bookstore/book/title"
// code for IE
if (window.ActiveXObject)
{
var nodes=xml.selectNodes(path);
```

Your Result:

Everyday Italian
Harry Potter
XQuery Kick Start
Learning XML

Edit the code above and click to see the result.

Xpath: XML Path Language

Location Paths: Nodo Contexto

- Un location path siempre tiene un nodo contexto
 - Es similar al concepto de directorio actual:
 - ls ./juan/documentos
- En XPath, si la expresión comienza por / estamos dando un path absoluto, partiendo del nodo raíz.
- ***Si no comienza por /, estamos dando un camino relativo desde el nodo actual (nodo contexto).***

Xpath: XML Path Language

Location Paths: Nodo Contexto

■ 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 subexpresión 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. Luego el Nodo Contexto puede ser en si un conjunto de nodos, para los que se evaluaría una expresión empezando por uno de ellos, que será el Current node en el momento de evaluar la expresión.

Xpath: XML Path Language

Location Paths: Nodo Contexto

/libro/capitulo/parrafo

- El analizador comienza por leer **/**, lo cual le dice que debe seleccionar el nodo raíz, independientemente del nodo contexto que en ese momento exista. En el momento en que el evaluador de XPath localiza el nodo raíz, éste pasa a ser el nodo contexto de dicha expresión.
- El analizador leería ahora **libro**, lo cual le dice que seleccione TODOS los elementos que cuelgan del nodo contexto (que ahora mismo es el nodo raíz) y que se llamen libro. En este caso solo hay uno... porque solo puede haber un elemento raíz. El nodo contexto pasa a ser libro.
- A continuación el analizador leería **capítulo**, entonces selecciona TODOS los elementos que cuelgan del nodo contexto (ahora es el nodo libro).
 - En un disco sería imposible que hubiera dos directorios con el mismo nombre colgando de un mismo directorio padre.
 - En estos momentos hay dos elementos que encajan con el patrón **/libro/capítulo**.
- ***El analizador continua leyendo parrafo. Con lo que selecciona TODOS los elementos parrafo que cuelgan del nodo contexto...jjpero NO hay un nodo contexto, sino DOS!!.*** ***El evaluador de expresiones lo va a recorrer uno por uno haciendo que, mientras evalúa un determinado nodo, ése sea el nodo contexto de ese momento.***

Xpath: XML Path Language

Location Paths

Un location path consta de:

- ❑ **Eje (axis).** Es la relación entre el nodo de contexto y el Location Path
 - El eje a veces está implícito (no se pone, y entonces se sobreentenderá “es hijo de”).
 - Para nosotros esto será lo común.
- ❑ **Prueba de nodo** (node test). Es el “nombre de directorio”
- ❑ **Predicado (predicate).** Expresión XPath entre corchetes.
 - El predicado es opcional

eje::pruebanodo[predicado]

Xpath: XML Path Language

Location Paths: Prueba de Nodos

- La forma más simple es escribir simplemente el **nombre del nodo** (su etiqueta)
- * que simboliza cualquier nombre
- Ejemplos:
 - /libro/capitulo[last()]/parrafo
 - Encaja con cualquier nodo “parrafo” que sea hijo del último nodo “capitulo” que sea hijo de un nodo “libro” que será el nodo raíz
 - /libro/capitulo/parrafo[1]
 - Encaja con el primer nodo “parrafo” que sea hijo de un nodo “capitulo” que sea hijo de un nodo “libro” que será el nodo raíz
 - /libro/*
 - * simboliza cualquier nombre: encaja con cualquier nodo que sea hijo del nodo “libro” que será el hijo del nodo raíz.
 - capitulo[1]/*
 - Encaja con cualquier nodo que sea hijo del primer “capitulo” que sea hijo del nodo de contexto
- IMPORTANTE: // indica “que sea hijo de cualquiera”
 - Los atributos se especifican con el **prefijo @**.

Xpath: XML Path Language

Location Paths: Prueba de Nodos

XPath:

output to own window

```
/libro/capitulo[last()]/parrafo
```

Test! **Format!** **Validate!** **Fix!** **Color!** **Save!** **Transform!**

```
<?xml version="1.0" encoding="UTF-8"?>

<root>
  <parrafo>Era una obscura noche del mes de septiembre...</parrafo>
  <parrafo>Ella, inocente cual
    <enlace href="http://www.abejilla.es">abejilla</enlace> que surca el viento en busca del nectar de las flores...
  </parrafo>
</root>
```

XPath:

output to own window

```
/libro/capitulo/parrafo[1]
```

Test! **Format!** **Validate!** **Fix!** **Color!** **Save!** **Transform!**

```
<?xml version="1.0" encoding="UTF-8"?>

<root>
  <parrafo>Era una sombría noche del mes de agosto...</parrafo>
  <parrafo>Era una obscura noche del mes de septiembre...</parrafo>
</root>
```

Xpath: XML Path Language

Location Paths: Prueba de Nodos

/AAA

Select the root element AAA

```
<AAA>
<BBB/>
<CCC/>
<BBB/>
<BBB/>
<DDD>
    <BBB/>
</DDD>
<CCC/>
</AAA>
```

/AAA/CCC

Select all elements CCC which are children of the root element AAA

```
<AAA>
<BBB/>
<CCC/>
<BBB/>
<BBB/>
<DDD>
    <BBB/>
</DDD>
<CCC/>
</AAA>
```

// BBB

Select all elements BBB

```
<AAA>
<BBB/>
<CCC/>
<BBB/>
<DDD>
    <BBB/>
</DDD>
<CCC>
    <DDD>
        <BBB/>
        <BBB/>
    </DDD>
</CCC>
</AAA>
```

// DDD/ BBB

Select all elements BBB which are children of DDD

```
<AAA>
<BBB/>
<CCC/>
<BBB/>
<DDD>
    <BBB/>
</DDD>
<CCC>
    <DDD>
        <BBB/>
        <BBB/>
    </DDD>
</CCC>
</AAA>
```

Xpath: XML Path Language

Location Paths: Prueba de Nodos

/AAA/CCC/DDD/*

Select all elements enclosed by elements /AAA/CCC/DDD

```
<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </CCC>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
```

/*/*/*/BBB

Select all elements BBB which have 3 ancestors

```
<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </CCC>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
```

Xpath: XML Path Language

Location Paths: Prueba de Nodos

//*

Select all elements

```
<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </CCC>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
```

¿Qué conjunto de nodos se seleccionaría si solo hubiera puesto /*?

Xpath: XML Path Language

Location Paths: Evaluacion de expresiones Xpath on line

XML validator, XSLT transformer and Simple XPath online tester

Experimental: you may paste part of XML and try the Fix button

XPath:

XML:

```
<?xml version="1.0"?>
<!--<?xml-stylesheet type="text/xsl" href="evalua_xpath.xsl"?>-->
<libro>
    <titulo>Dos por tres calles</titulo>
    <autor>Josefa Santos</autor>
    <capitulo num="1">
        La primera calle
        <parrafo>
            Era una sombría noche del mes de agosto...
        </parrafo>
        <parrafo destacar="si">
            Ella, inocente cual
            <enlace href="http://www.enlace.es">mariposa</enlace>
            que surca el cielo en busca de libaciones...
        </parrafo>
    </capitulo>
    <capitulo num="2" public="si">
        La segunda calle
        <parrafo>
            Era una obscura noche del mes de septiembre...
        </parrafo>
        <parrafo>
            Ella, inocente cual
            <enlace href="http://www.abejilla.es">abejilla</enlace>
            que surca el viento en busca del néctar de las flores...
        </parrafo>
    </capitulo>
    <apendice num="a" public="si">
        La tercera calle
        <parrafo>
```

Test! Format! Validate! Fix! Color! Save! Transform!

feedback

XML validator, XSLT transformer and Simple XPath online tester

Experimental: you may paste part of XML and try the Fix button

XPath:

XML:

```
/libro/capitulo/parrafo
```

Test! Format! Validate! Fix! Color! Save! Transform!

feedback

```
<?xml version="1.0" encoding="UTF-8"?>

<root>
    <parrafo>Era una sombría noche del mes de agosto...</parrafo>
    <parrafo destacar="si">Ella, inocente cual
        <enlace href="http://www.enlace.es">mariposa</enlace> que surca el cielo en busca de libaciones...
    </parrafo>
    <parrafo>Era una obscura noche del mes de septiembre...</parrafo>
    <parrafo>Ella, inocente cual
        <enlace href="http://www.abejilla.es">abejilla</enlace> que surca el viento en busca del néctar de las flores...
    </parrafo>
</root>
```

<http://www.xpathtester.com/test>

Xpath: XML Path Language

Location Paths: Evaluacion de expresiones Xpath on line

Testbed

XPath Expression:

[Evaluate](#)

History : //nombre [Clear History](#)

Load your own source file [Examinar...](#) [Upload File](#)

Result is a NodeSet containing 3 elements

[show key](#)

```
<documentRoot>
  <?version="1.0" encoding="UTF-8"?>
  <?type="text/xsl" href="nombre_ciudades.xsl"?>
  <ciudades>
    <ciudad>
      <nombre>Madrid</nombre>
      <habitantes>3500000</habitantes>
    </ciudad>
    <ciudad>
      <nombre>Malaga</nombre>
      <habitantes>800000</habitantes>
    </ciudad>
    <ciudad>
      <nombre>Toledo</nombre>
      <habitantes>50000</habitantes>
    </ciudad>
  </ciudades>
</documentRoot>
```

<http://www.whitebeam.org/library/guide/TechNotes/xpathtestbed.rhtm>

Xpath: XML Path Language

Location Paths: Evaluacion de expresiones Xpath on line

XPath_1 - Zvon XLab

[Back](#) | [Forward](#) | [Previous](#) | [Next](#)

Expressions

nodes selected by XPath expression are highlighted

Select XML file: **1** 2 3 4 5

```
<aaa>
  <bbb/>
  <ccc a="1" b="2">33333</ccc>
</aaa>
```

Create and transform XML documents Flexible & easy to use!
Free trial AdChoices ▾

enable regexp (?)

[First](#) [Prev](#) [Next](#)

1 - 3 filter: off (3)

* Expressions

Axes

Keys

http://www.zvon.org/comp/tests/r/test-xlab.html#XPath_1~Expressions

Xpath: XML Path Language

Location Paths: Prueba de nodo

*

- Devuelve todos los nodos de tipo principal (es decir, elemento, atributo o espacio de nombres), pero no nodos de texto, comentarios y de instrucciones de proceso.
- **Ejemplo:** Seleccionar todos los nodos principales descendientes de los **parrafo**:
 - //parrafo/*

node()

- El predicado node() devuelve todos los nodos de todos los tipos.
- **Ejemplo:** Seleccionar todos los nodos descendientes de los **parrafo**:
 - //parrafo/node()

text()

- **Ejemplo:**
 - Seleccionar el texto de todos los nodos parrafo:
 - //parrafo/text()
 - Seleccionar TODO el texto que cuelga de todos los nodos parrafo:
 - //parrafo//text()
 - ojo: ¿ves la diferencia con el anterior? Véase *las siguientes diapositivas*

Xpath: XML Path Language

Location Paths: Prueba de nodo

XPath Expression:

```
//parrafo/*
```

History : //parrafo/*

Clear History

Load your own source file

XPath Expression:

```
//parrafo/node()
```

History : //parrafo/node()
//parrafo/*

Clear History

Load your own source file

Result is a NodeSet containing 3 elements

```
<documentRoot>
  <?version="1.0"?>
  <?type="text/xsl" href="evalua_xpath.xsl"?>
  <libro>
    <titulo>Dos por tres calles</titulo>
    <autor>Josefa Santos</autor>
    <capitulo num="1" >
      La primera calle
      <parrafo>
        Era una sombría noche del mes de agosto...
      </parrafo>
      <parrafo destacar="si" >
        Ella, inocente cual
        <enlace href="http://www.enlace.es" >mariposa</enlace>
        que surca el cielo en busca de libaciones...
      </parrafo>
    </capitulo>
    <capitulo num="2" public="si" >
      La segunda calle
    </capitulo>
```

Result is a NodeSet containing 12 elements

```
<documentRoot>
  <?version="1.0"?>
  <?type="text/xsl" href="evalua_xpath.xsl"?>
  <libro>
    <titulo>Dos por tres calles</titulo>
    <autor>Josefa Santos</autor>
    <capitulo num="1" >
      La primera calle
      <parrafo>
        Era una sombría noche del mes de agosto...
      </parrafo>
      <parrafo destacar="si" >
        Ella, inocente cual
        <enlace href="http://www.enlace.es" >mariposa</enlace>
        que surca el cielo en busca de libaciones...
      </parrafo>
    </capitulo>
    <capitulo num="2" public="si" >
      La segunda calle
    </capitulo>
```

Xpath: XML Path Language

Location Paths: Prueba de nodo

XPath Expression:

```
//parrafo/text()
```

History :
//parrafo/text()
//parrafo/node()
//parrafo/*

Load your own source file

Examinar... Upload File

XPath Expression:

```
//parrafo//text()
```

History :
//parrafo//text()
//parrafo/text()
//parrafo/node()
//parrafo/*

Load your own source file

Examinar... Upload File

Evaluate

Result is a NodeSet containing 9 elements

```
<documentRoot>
<?version="1.0"?>
<?type="text/xsl" href="evalua_xpath.xls"?>
<libro>
<titulo>Dos por tres calles</titulo>
<autor>Josefa Santos</autor>
<capitulo num="1" >
La primera calle
<parrafo>
Era una sombría noche del mes de agosto...
</parrafo>
<parrafo destacar="si" >
Ella, inocente cual
<enlace href="http://www.enlace.es" >mariposa</enlace>
que surca el cielo en busca de libaciones...
</parrafo>
</capitulo>
<capitulo num="2" public="si" >
La segunda calle
```

Result is a NodeSet containing 12 elements

```
<documentRoot>
<?version="1.0"?>
<?type="text/xsl" href="evalua_xpath.xls"?>
<libro>
<titulo>Dos por tres calles</titulo>
<autor>Josefa Santos</autor>
<capitulo num="1" >
La primera calle
<parrafo>
Era una sombría noche del mes de agosto...
</parrafo>
<parrafo destacar="si" >
Ella, inocente cual
<enlace href="http://www.enlace.es" >mariposa</enlace>
que surca el cielo en busca de libaciones...
</parrafo>
</capitulo>
<capitulo num="2" public="si" >
La segunda calle
```

Xpath: XML Path Language

Location Paths: Predicado

eje::pruebanodo[predicado]

- Un predicado permite restringir el conjunto de nodos seleccionados a aquellos que cumplen cierta condición.
 - Dicha condición es una expresión XPath y se especifica entre corchetes.
 - En una **Expresión Booleana**: un nodo del conjunto seleccionado la cumplirá, o no la cumplirá.
- Dada la prueba de nodo, y dado el eje (supongamos “es hijo de”), del conjunto de nodos resultante quedan sólo los que cumplan el predicado.
- En el predicado pueden intervenir **funciones Xpath**.

List of XPaths
XPath as filesystem addressing
Start with //
All elements: *
Further conditions inside []
Attributes
Attribute values
* Nodes counting
Playing with names of selected elements
Length of string
Combining XPaths with
Child axis
Descendant axis
Parent axis
Ancestor axis
Following-sibling axis
Preceding-sibling axis
Following axis
Preceding axis
Descendant-or-self axis

http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Further_conditions_inside_%5B%5D

Xpath: XML Path Language

Location Paths: Predicado

■ Ejemplos para poder **probarlos con nuestro fichero libro.xml**:

- *Ejemplo:* Seleccionar todos los **capítulo** que tengan un **parrafo** que tenga algún elemento con atributo **href**:
 - **//capítulo[parrafo/*[@href]]**

➤ Serán muchas las ocasiones en que deseemos que los nodos a seleccionar no cumplan una sino varias condiciones. En estos casos, los predicados se pueden suceder uno a otro haciendo el efecto de la operación AND. Como en el siguiente ejemplo:

- *Ejemplo:* Seleccionar todos los **capítulo** que tengan un **parrafo** que tenga algún elemento con atributo **href** y que ellos mismos (los **capítulo**) tengan el atributo **public** a valor **si**:
 - **//capítulo [parrafo/*[@href]] [@public='si']**

Xpath: XML Path Language

Location Paths: Predicado

- También se puede hacer uso del operador **and** encerrando entre paréntesis los distintos predicados lógicos. **Ejemplo** similar al anterior:
 - `//capítulo[(parrafo/*[@href]) and (@public='si')]`
 - También se puede hacer uso de la operación **OR** en vez de **and**.
- Existe otro tipo de operación **or** que utiliza la barra vertical: **|** separando no dos predicados, sino dos expresiones XPath.
- **Ejemplo** de **|**: Seleccionar todos los **capítulo** que tengan un **parrafo** que tenga algún elemento con atributo **href** o todos los **apendice**:
 - `//capítulo[parrafo/*[@href]] | //apendice`

http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Combining_XPaths_with_%7C

Xpath: XML Path Language

Location Paths: Predicado

- Por último, también podemos especificar con **not** la negación de alguna de las negaciones del predicado.
- **Ejemplo:** Seleccionar todos los **capítulo** que no tengan el atributo **public**: `//capítulo[not(@public)]`
- **Predicados con funciones de cardinalidad:**
 - Existen funciones que nos van a servir para restringir los nodos basándose en la posición del elemento devuelto.
 - ❖ **position()**, → **Ejemplo:** `//capítulo[position()=2]` ó `//capítulo[2]`
 - ❖ **last()** → **Ejemplo:** `//capítulo[not(position()=last())]` ó

Xpath: XML Path Language

Location Paths: Predicado

XPath:

output to own window

```
//capítulo[position()=2]
```

XML:

[Test!](#) [Format!](#) [Validate!](#) [Fix!](#) [Color!](#) [Save!](#) [Transform!](#)

```
<capítulo num="2" public="si">La segunda calle
<parrafo>Era una obscura noche del mes de septiembre...</parrafo>
<parrafo>Ella, inocente cual
  <enlace href="http://www.abejilla.es">abejilla</enlace> que surca el viento en busca del nectar de las flores...
</parrafo>
</capítulo>
```

XPath:

output to own window

```
//capítulo[not(position()=last())]
```

XML:

[Test!](#) [Format!](#) [Validate!](#) [Fix!](#) [Color!](#) [Save!](#) [Transform!](#)

```
<?xml version="1.0" encoding="UTF-8"?>

<root>
  <capítulo num="1">La primera calle
    <parrafo>Era una sombría noche del mes de agosto...</parrafo>
    <parrafo destacar="si">Ella, inocente cual
      <enlace href="http://www.enlace.es">mariposa</enlace> que surca el cielo en busca de libaciones...
    </parrafo>
  </capítulo>
  <capítulo num="2" public="si">La segunda calle
    <parrafo>Era una obscura noche del mes de septiembre...</parrafo>
    <parrafo>Ella, inocente cual
      <enlace href="http://www.abejilla.es">abejilla</enlace> que surca el viento en busca del nectar de las flores...
    </parrafo>
  </capítulo>
</root>
```

He añadido un 3º capítulo

Xpath: XML Path Language

Location Paths: Predicado

- Ejercicio: crear una hoja para el horario en la que salgan las tareas después del miércoles (día 3 en adelante): horario2.xsl

```
?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="ISO-8859-1"/>
<xsl:template match="horario">
    <html>
        <head><title>Horario</title></head>
        <body>
            <xsl:apply-templates select="dia[numdia &gt;= 3]"/>
        </body>
    </html>
</xsl:template>
<xsl:template match="dia">
    <p>Día: [<xsl:value-of select="numdia"/>]</p>
    <xsl:for-each select="tarea">
        <p>Tarea: [<xsl:value-of select=".//>]</p>
    </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```



Xpath: XML Path Language

Location Paths: Predicado

- Hay una gran variedad de **funciones** que podemos usar en el predicado:
 - **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.
 - Ver http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Nodes_counting
 - **name()**: Devuelve el nombre de un nodo (su etiqueta). local-name(), namespace-uri()
 - Ver http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Playing_with_names_of_selected_elements
 - Biblioteca de strings:
 - **normalize-space()**, **string()**, **concat()**, **stringlength()**
 - Ver http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Length_of_string
 - **sum()**
 - Recuerda también las de **cardinalidad**: id(), last(), position()

Xpath: XML Path Language

Location Paths: Predicado

- Escribir una hoja que muestre en HTML todos los nodos de un documento, como listas no numeradas, indicando el número de orden de cada nodo y el número de hijos que contiene (cada elemento irá, además, numerado): horario3.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="windows-1252" indent="yes"/>
<xsl:template match="horario">
    <HTML>
        <HEAD><TITLE>Horario</TITLE></HEAD>
        <BODY>
            <xsl:apply-templates select="*"/>
        </BODY>
    </HTML>
</xsl:template>
<xsl:template match="*"
    <LI>
        <STRONG><xsl:value-of select="position()"/>
        <xsl:text> </xsl:text><xsl:value-of select="name()"/>
    </STRONG>
    - Hijos: <xsl:value-of select="count(*)"/>
</LI>
<UL>
    <xsl:apply-templates select="*"/> <!-- Llamada recursiva -->
</UL>
</xsl:template>
</xsl:stylesheet>
```



- 1 dia - Hijos: 2
 - 1 numdia - Hijos: 0
 - 2 tarea - Hijos: 3
 - 1 hora-ini - Hijos: 0
 - 2 hora-fin - Hijos: 0
 - 3 nombre - Hijos: 0
- 2 dia - Hijos: 2
 - 1 numdia - Hijos: 0
 - 2 tarea - Hijos: 3
 - 1 hora-ini - Hijos: 0
 - 2 hora-fin - Hijos: 0
 - 3 nombre - Hijos: 0
- 3 dia - Hijos: 3
 - 1 numdia - Hijos: 0
 - 2 tarea - Hijos: 3
 - 1 hora-ini - Hijos: 0
 - 2 hora-fin - Hijos: 0
 - 3 nombre - Hijos: 0
- 3 tarea - Hijos: 3
 - 1 hora-ini - Hijos: 0
 - 2 hora-fin - Hijos: 0
 - 3 nombre - Hijos: 0
- 4 dia - Hijos: 2
 - 1 numdia - Hijos: 0
 - 2 tarea - Hijos: 3
 - 1 hora-ini - Hijos: 0
 - 2 hora-fin - Hijos: 0
 - 3 nombre - Hijos: 0
- 5 dia - Hijos: 2
 - 1 numdia - Hijos: 0
 - 2 tarea - Hijos: 3
 - 1 hora-ini - Hijos: 0
 - 2 hora-fin - Hijos: 0
 - 3 nombre - Hijos: 0

Xpath: XML Path Language

Location Paths: Predicado

`//*[count(BBB)=2]`

Select elements which have two children BBB

```
<AAA>
  <CCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </CCC>
  <DDD>
    <BBB/>
    <BBB/>
  </DDD>
  <EEE>
    <CCC/>
    <DDD/>
  </EEE>
</AAA>
```

`//*[count(*)=2]`

Select elements which have 2 children

```
<AAA>
  <CCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </CCC>
  <DDD>
    <BBB/>
    <BBB/>
  </DDD>
  <EEE>
    <CCC/>
    <DDD/>
  </EEE>
</AAA>
```

`//*[count(*)=3]`

Select elements which have 3 children

```
<AAA>
  <CCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </CCC>
  <DDD>
    <BBB/>
    <BBB/>
  </DDD>
  <EEE>
    <CCC/>
    <DDD/>
  </EEE>
</AAA>
```

Xpath: XML Path Language

Location Paths: Predicado

`//*[name()='BBB']`

Select all elements with name BBB, equivalent with `//BBB`

```
<AAA>
  <BCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </BCC>
  <DDB>
    <BBB/>
    <BBB/>
  </DDB>
  <BEC>
    <CCC/>
    <DBD/>
  </BEC>
</AAA>
```

`//*[starts-with(name(),'B')]`

Select all elements name of which starts with letter B

```
<AAA>
  <BCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </BCC>
  <DDB>
    <BBB/>
    <BBB/>
  </DDB>
  <BEC>
    <CCC/>
    <DBD/>
  </BEC>
</AAA>
```

`//*[contains(name(),'C')]`

Select all elements name of which contain letter C

```
<AAA>
  <BCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </BCC>
  <DDB>
    <BBB/>
    <BBB/>
  </DDB>
  <BEC>
    <CCC/>
    <DBD/>
  </BEC>
</AAA>
```

Xpath: XML Path Language

Location Paths: Predicado

```
//*[string-length(name()) = 3]
```

Select elements with three-letter name

```
<AAA>
  <Q/>
  <SSSS/>
  <BB/>
  <CCC/>
  <DDDDDDDD/>
  <EEEE/>
</AAA>
```

```
//*[string-length(name()) < 3]
```

Select elements name of which has one or two characters

```
<AAA>
  <Q/>
  <SSSS/>
  <BB/>
  <CCC/>
  <DDDDDDDD/>
  <EEEE/>
</AAA>
```

```
//*[string-length(name()) > 3]
```

Select elements with name longer than three characters

```
<AAA>
  <Q/>
  <SSSS/>
  <BB/>
  <CCC/>
  <DDDDDDDD/>
  <EEEE/>
</AAA>
```

Xpath: XML Path Language

Location Paths: Predicado

//BBB[@id='b1']

Select BBB elements which have attribute id with value b1

```
<AAA>
  <BBB id = "b1"/>
  <BBB name = "bbb "/>
  <BBB name = "bbb"/>
</AAA>
```

//BBB[@name='bbb']

Select BBB elements which have attribute name with value 'bbb'

```
<AAA>
  <BBB id = "b1"/>
  <BBB name = "bbb "/>
  <BBB name = "bbb"/>
</AAA>
```

//BBB[normalize-space(@name)='bbb']

Select BBB elements which have attribute name with value bbb,
leading and trailing spaces are removed before comparison

```
<AAA>
  <BBB id = "b1"/>
  <BBB name = "bbb "/>
  <BBB name = "bbb"/>
</AAA>
```

//@id

Select all attributes @id

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

//BBB[@id]

Select BBB elements which have attribute id

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

//BBB[@name]

Select BBB elements which have attribute name

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

//BBB[@*]

Select BBB elements which have any attribute

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

//BBB[not(@*)]

Select BBB elements without an attribute

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

Xpath: XML Path Language

Location Paths: Predicado

- Ejercicio: Generar una versión del horario que para cada día muestra la lista de tareas (sus nombres) y su prioridad, y también la hora de inicio y fin: horario4.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="windows-1252" indent="yes"/>
<xsl:template match="horario">
    <HTML>
        <HEAD><TITLE>Horario</TITLE></HEAD>
        <BODY>
            <xsl:apply-templates select="dia"/>
        </BODY>
    </HTML>
</xsl:template>
<xsl:template match="dia">
    <P>Día <xsl:value-of select="numdia"/></P>
    <UL>
        <xsl:apply-templates select="tarea"/>
    </UL>
</xsl:template>
<xsl:template match="tarea">
    <LI><STRONG><xsl:value-of select="nombre"/></STRONG>-
        Prioridad:<xsl:value-of select=".//@prioridad"/><BR></BR>
        De <xsl:value-of select="hora-ini"/> a <xsl:value-of select="hora-fin"/>
    </LI>
</xsl:template>
</xsl:stylesheet>
```



Dia 1

- Tutorías- Prioridad:media
De 12 a 14

Dia 2

- Autómatas- Prioridad:alta
De 12 a 14

Dia 4

- Procesadores de lenguajes- Prioridad:alta
De 9 a 11
- EDI- Prioridad:
De 16 a 17

Dia 3

- Procesadores de lenguajes- Prioridad:alta
De 9 a 11

Dia 5

- Ver la tele- Prioridad:baja
De 17 a 18

Xpath: XML Path Language

Location Paths: Ejes (axis)

eje::pruebanodo[predicado]

- El eje denota la relación de un Localization Paths 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**.
- Equivale a “que es un”, pero sus argumentos se leen de derecha a izquierda
- **child** está **implícito** y casi nunca se pone.
 - Pero para el nodo raíz, está implícito self (self denota al nodo de contexto)
- Ejemplos:
 - **/universidad/euitio**
 - Equivale de manera implícita a **/self::universidad/child::euitio**

Xpath: XML Path Language

Location Paths: Ejes (axis)

child::

- Es el eje utilizado por defecto. Se corresponde con la barra, / (aunque tiene una forma más larga que es: /child::).
- **Ejemplo:** Seleccionar todos los **titulo** de un **libro**:
 - /libro/titulo -> /libro/child::titulo

Xpath: XML Path Language

Location Paths: Ejes (axis)

attribute::

- Se corresponde con el signo de la arroba, @ (o en su forma larga que es: attribute::). Mediante este operador podemos seleccionar aquellos nodos atributos que deseemos, indicando el nombre del atributo en cuestión.
- **Ejemplo:** Seleccionar el atributo num que posean los elementos **capítulo**
 - /libro/capítulo/@num -> /libro/capítulo/attributte::num
- **Ejemplo:** Seleccionar todos los elementos hijo de los **capítulo** que posean el atributo **public** (sin importar el valor asignado al mismo):
 - /libro/capítulo[@public]/* -> /libro/capítulo[attributte::public]/*
- **Ejemplo:** Seleccionar todos los elementos hijo de **parrafo** cuyo atributo **destacar** sea igual a "si".
 - /libro/título/parrafo[@destacar="si"] -> /libro/título/parrafo[attributte::destacar="si"]

Xpath: XML Path Language

Location Paths: Ejes (axis)

■ Recuerda que se puede acceder a un elemento atributo gracias al eje attribute::

- Una abreviatura de esto es la arroba @
- Más ejemplos:
 - Ver http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Attributes
 - Ver http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Attribute_valu_es

The screenshot shows a user interface for evaluating XPath expressions. At the top, there is a text input field containing the XPath query `//capítulo[attribute::num=1]`. To the right of the input field is a button labeled "Evaluate". Below the input field, there is a section titled "History" which also contains the query `//capítulo[attribute::num=1]`. Next to the history section are two buttons: "Clear History" and "Evaluate". Further down, there is a section titled "Load your own source file" with three buttons: "Seleccionar archivo" (Select file), "No se ha ...n archivo" (No file selected), and "Upload File". The main result area is titled "Result is a NodeSet containing 1 element" and shows the XML document structure. The XML code is as follows:

```
<documentRoot>
<?version="1.0"?>
<!-<?xmlstylesheet type="text/xsl" href="evalua_xpath.xsl"?->-->
<libro>
<titulo>Dos por tres calles</titulo>
<autor>Josefa Santos</autor>
<capítulo num="1" >
La primera calle
<parrafo>
Era una sombría noche del mes de agosto...
</parrafo>
<parrafo destacar="si" >
Ella, inocente cual
<enlace href="http://www.enlace.es" >mariposa</enlace>
que surca el cielo en busca de libaciones...
</parrafo>
</capítulo>
<capítulo num="2" public="si" >
La segunda calle
<parrafo>
Era una obscura noche del mes de septiembre...
</parrafo>
<parrafo>
Ella, inocente cual
<enlace href="http://www.enlace.es" >ahotilla</enlace>
```

Xpath: XML Path Language

Location Paths: Ejes (axis)

- No haremos ejercicios con los ejes, pero si os propongo ver más ejemplos.
- Ejes posibles: ***ancestor, ancestor-or-self, attribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, self***. Ejemplos:
 - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Child_axis
 - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Descendant_axis
 - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Parent_axis
 - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Ancestor_axis
 - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Following-sibling_axis
 - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Preceding-sibling_axis
 - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Following_axis
 - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Preceding_axis
 - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Descendant-or-self_axis
- Para practicar: http://www.zvon.org/comp/tests/r/test-xlab.html#XPath_1~Axes

Xpath: XML Path Language

Acceso a elementos de otro documento XML

- Se puede acceder a datos de otro fichero XML:
 - Mediante la función **document()**
- Ejercicio: usando el fichero “literales.xml” generar una versión del horario (sobre horario4.xsl) que en vez de “Día 1” muestre “Lunes” y así sucesivamente
 - horario5.xsl.
 - NOTA: habrá que usar concat(), current() [nodo contexto] y normalize-space() [normalizar espacios dejando un espacio entre los componentes del texto y elimina los espacios al principio y al final]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<literales>
    <dia1>Lunes</dia1>
    <dia2>Martes</dia2>
    <dia3>Miércoles</dia3>
    <dia4>Jueves</dia4>
    <dia5>Viernes</dia5>
    <dia6>Sábado</dia6>
    <dia7>Domingo</dia7>
</literales>
```

Xpath: XML Path Language

Acceso a elementos de otro documento XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="horario">
    <HTML>
        <HEAD><TITLE>Horario</TITLE></HEAD>
        <BODY>
            <xsl:apply-templates select="dia"/>
        </BODY>
    </HTML>
</xsl:template>
<xsl:template match="dia"> <!-- por cada /horario/dia se realiza el select de value-of con el current() correspondiente-->
    <P><xsl:value-of
        select="document('literales.xml')/literales/*[name()='concat('dia', normalize-
        space(current()/numdia))]" /> </P>
        <UL>
            <xsl:apply-templates select="tarea"/>
        </UL>
</xsl:template>
<xsl:template match="tarea">
    <LI><STRONG><xsl:value-of select="nombre"/></STRONG>-
        Prioridad:<xsl:value-of select="./@prioridad"/><BR></BR>
        De <xsl:value-of select="hora-ini"/> a <xsl:value-of select="hora-fin"/>
    </LI>
</xsl:template>
</xsl:stylesheet>
```

Lunes

- Tutorias- Prioridad:media
De 12 a 14

Martes

- Autómatas- Prioridad:alta
De 12 a 14

Jueves

- Procesadores de lenguajes- Prioridad:alta
De 9 a 11
- EDI- Prioridad:
De 16 a 17

Miércoles

- Procesadores de lenguajes- Prioridad:alta
De 9 a 11

Viernes

- Ver la tele- Prioridad:baja
De 17 a 18

Representa todos los <dia> y de ellos se selecciona el texto de la etiqueta que cumpla el select

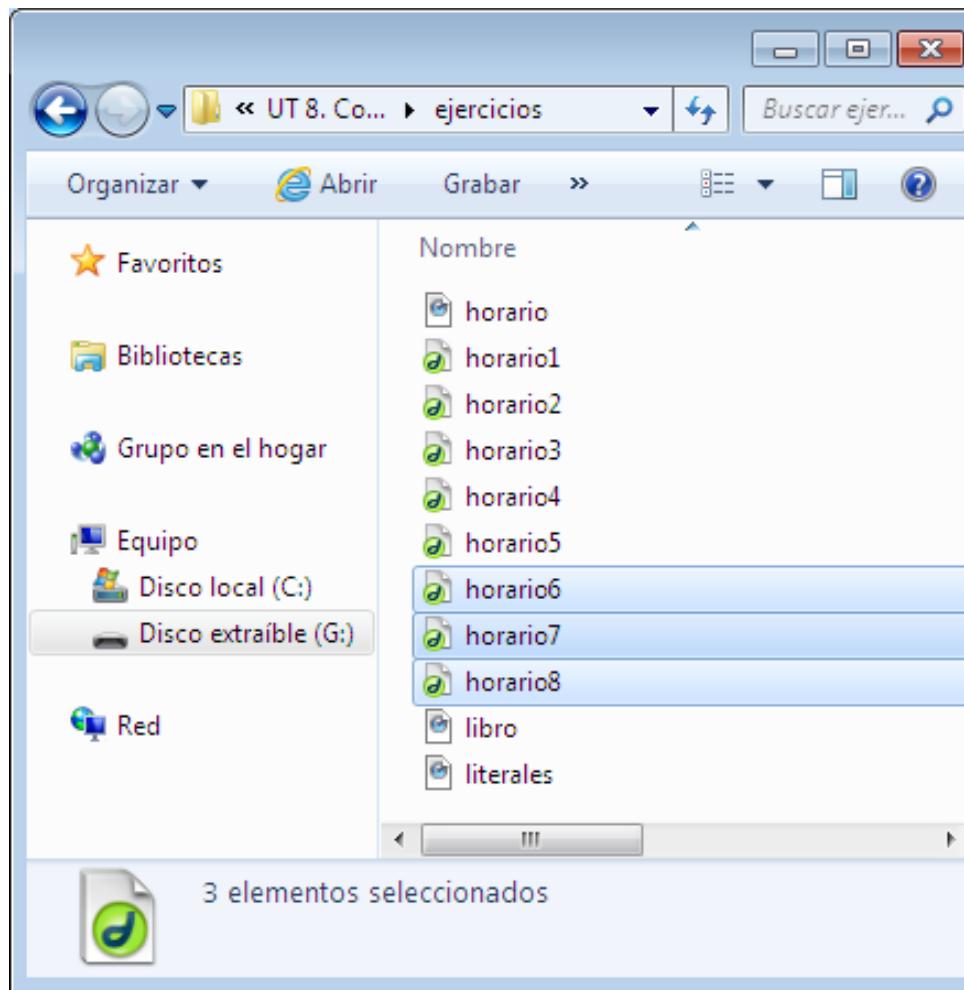
Xpath: XML Path Language

Acceso a elementos de otro documento XML: horario5.xls

- En vez de obtener el número de dia tal y como lo haciamos en horario4.xls:
 - <P>Día <xsl:value-of select="numdia"/></P>
tenemos que obtener su equivalente en literal (así, en vez de devolver “Día 1”, devolveremos “Lunes”, y asi sucesivamente).
- Los literales estan en literal.xml, luego la expresión que buscamos tendra este aspecto:
`<xsl:value-of select="document('literales.xml') ... />`
- Los literales están dentro del nodo literales:
`select="document('literales.xml')/literales/*"`
- Pero mediante un predicado hay que seleccionar el texto del nodo literal cuyo nombre corresponda al numero de dia que se esta procesando:
`select="document('literales.xml')/literales/*[name()= ...]"/>`
- Los nombres de los nodos literales son como “diaX”, por lo que habrá que implementar la condición de igualdad entre una cadena del tipo “diaX”, siendo X el valor de numdia, y el nombre del nodo hijo de literal diaX:
`[name()=concat('dia', normalize-space(current()/numdia))]`
- ¿Por que “current()/numdia” y “./numdia”?
Caundo se usan funciones anidadas, debemos referirnos al nodo contexto con current(), en vez de con .

Xpath: XML Path Language

Acceso a elementos de otro documento XML: Otros horarios.xls



Xpath: XML Path Language

Resumen: Con Xpath podemos

Seleccionar los nodos para la aplicación de templates.

Obtener valores (bastante elaborados)

La selección de nodos puede basarse en similitud de nombres, en el eje y/o en ciertas condiciones (predicado)

Xpath: XML Path Language

Ejercicios

- Realizar un resumen en forma de tutorial de la UT8.
- Engloba en un solo documento los ejercicios, ejemplos .. que se han planteado en la UT8
- Boletín de ejercicios “EJERCICIOS XPATH.doc” (incluye instrucciones precisas): Componer 27 expresiones xpath que respondan a cada una de las 27 cuestiones sobre datos de un XML dado (*tareas finales xpath.xml*)

Xpath: XML Path Language

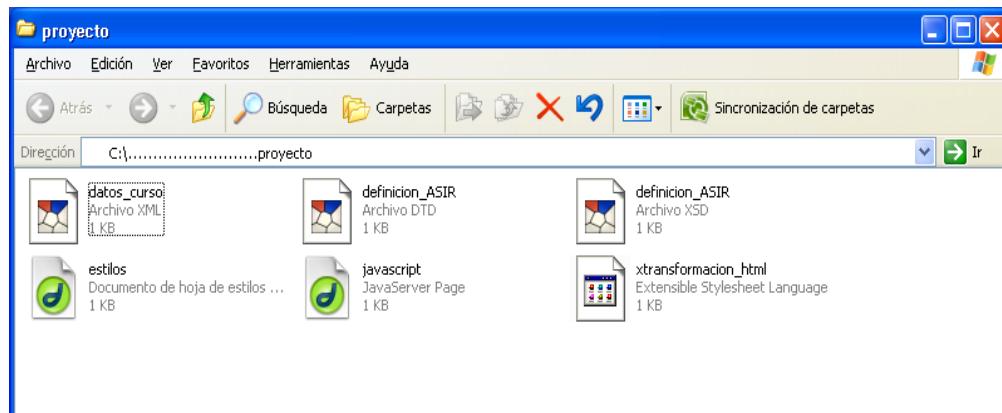
Proyecto Grupal



Ejemplo de Examen
View FireFox

Objetivos

- ◆ Dominar los conceptos fundamentales eXtensible Markup Language.
- ◆ (XHTML-CSS-XML-DTD-XSD-XSLT-XPATH-CSS-JAVASCRIPT-DOM....).
- ◆ Definir datos en formato XML que recoja la información necesaria para implementar este sistema de información.



Descripción General

Desarrolla un DTD/XSD que represente un entorno SI. Deberá crearse el XML bien formado y válido respecto a la definición creada, que mediante un XSLT se transformará en un HTML. Este HTML tendrá asociado un CSS que permita proporcionar una visión adecuada de la información que contiene el XML, así como los elementos JAVASCRIPT/DOM que permitan añadir la funcionalidad que el grupo considere oportuno.