

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).

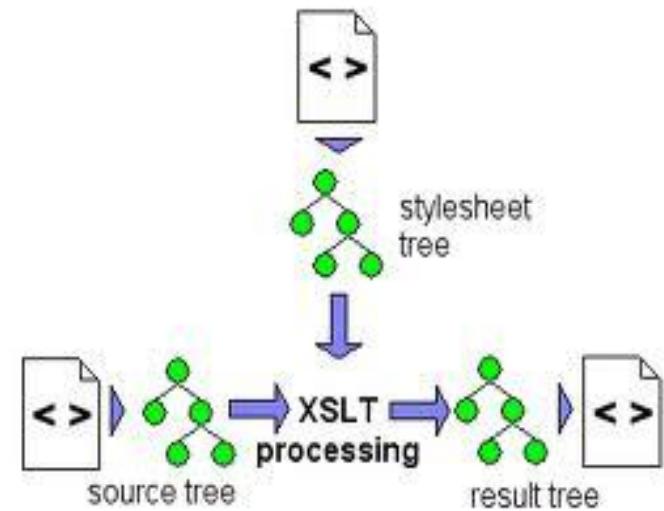
XSLT es pues, un lenguaje que se usa para convertir documentos XML en otros documentos XML; puede convertir un documento XML que obedezca a un DTD a otro que obedezca otro diferente, un documento XML bien formado a otro que siga un DTD, o, lo más habitual, convertirlo a "formatos finales", tales como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

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>
```



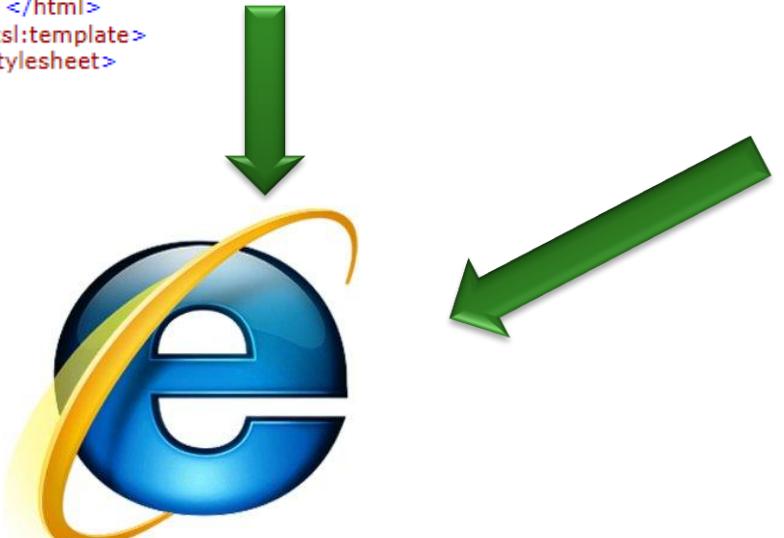
```
c:\tmp> xt documento1.xml documento1.xsl doc1.xml
```

```
<?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>
```



The screenshot shows a Microsoft WordPad window titled "libros - WordPad". The XML code displays a list of books:

```
<?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>
```

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

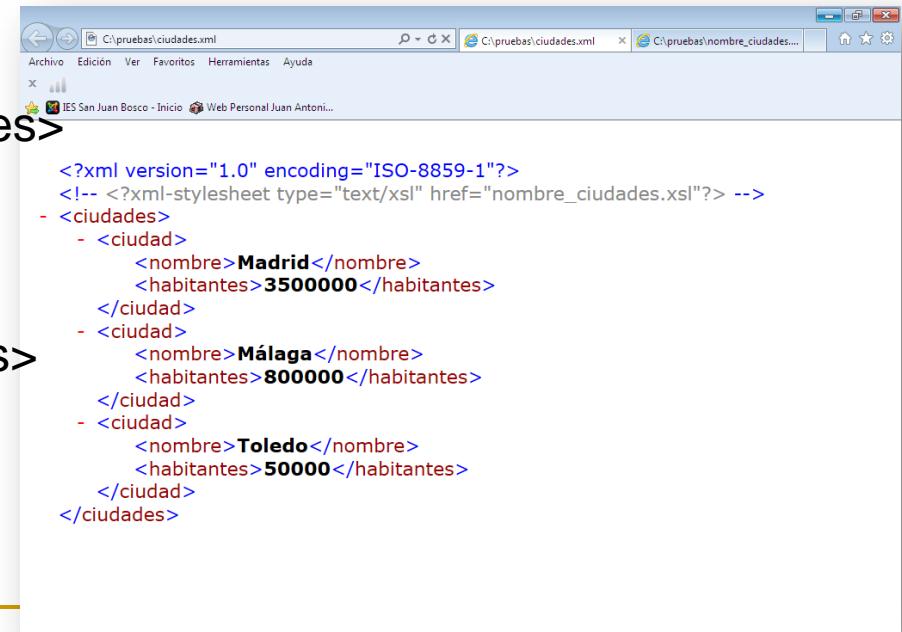
- La regla se aplicará a todas las instancias del elemento nombre que se encuentre en el xml al que se va a aplicar la transformación. Esto se indica mediante el atributo match que acompaña al elemento xsl:template.
- Entre las etiquetas de inicio y de fin del elemento xsl:template se escribe la transformación 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

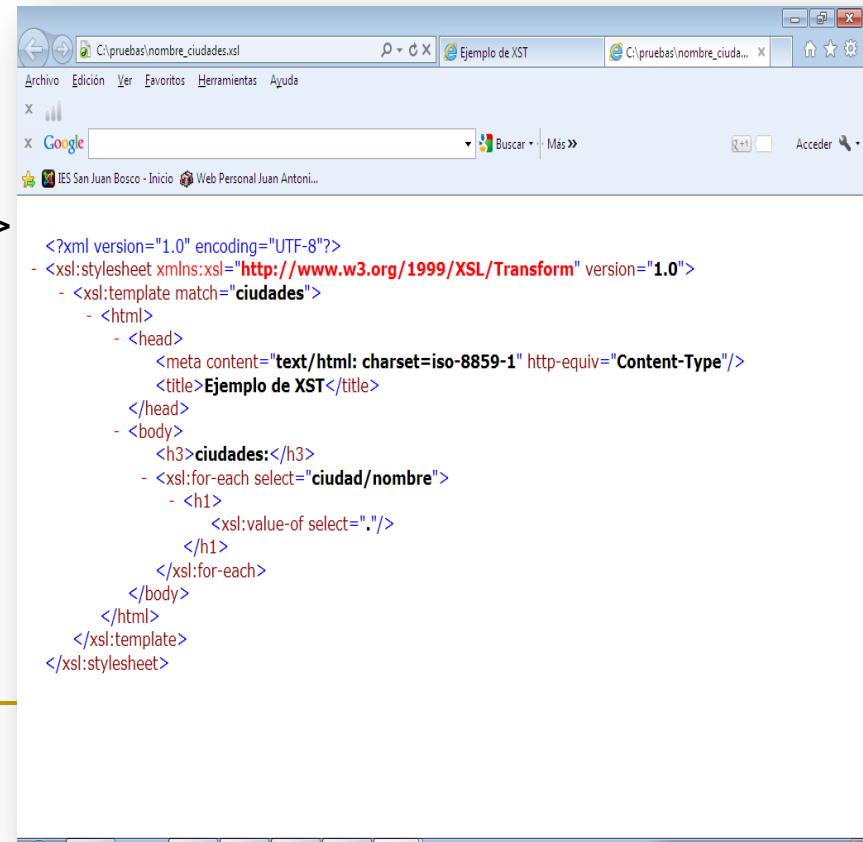


Introducción a la conversión de doc XML

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>
    <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

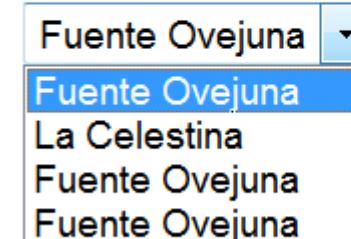
- La pretensión de este ejemplo es ver como podemos cargar un combo (desplegable) mediante una hoja XSLT. Es decir, utilizar los datos de un XML para que aparezcan como opciones de un combo.
- Para conseguir la carga del combo tenemos que definir una plantilla. Dicha plantilla lo que tendrá que hacer 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



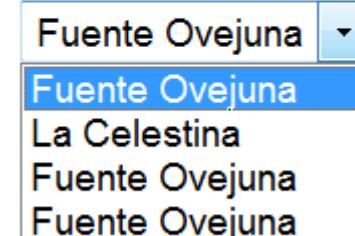
Introducción a la conversión de doc XML

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

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: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>
```