

Unidad 5 – Parte 2

Conversión y adaptación de documentos XML

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

ÍNDICE

1. XSLT
 1. ¿Qué es XSLT?
 2. Elementos en XSLT
2. Transformando con XSLT
 1. Referenciar un fichero XSLT desde un fichero XML
 2. XSLT: <xsl:stylesheet>
 3. XSLT: <xsl:template> y plantillas vacías
 4. XSLT: <xsl:template> y manipulación de plantillas
 5. XSLT: <xsl:output>
 6. XSLT: <xsl:value-of>
 7. XSLT: <xsl:for-each>
 8. XSLT: <xsl:sort>
 9. XSLT: <xsl:if>
 10. XSLT: <xsl:choose>, <xsl:when> y <xsl:otherwise>
 11. XSLT: <xsl:copy> y <xsl:copy-of>
 12. XSLT: <xsl:include> y <xsl:import>
 13. XSLT: <xsl:variable> y <xsl:param>
 14. XSLT: <xsl:element> y <xsl:attribute>
 15. XSLT: <xsl:key> y función key()
 16. XSLT: <xsl:comment>
 17. XSLT: Referencia
3. Bibliografía

ÍNDICE

1. XSLT

1. ¿Qué es XSLT?
2. Elementos en XSLT

2. Transformando con XSLT

1. Referenciar un fichero XSLT desde un fichero XML
2. XSLT: <xsl:stylesheet>
3. XSLT: <xsl:template> y plantillas vacías
4. XSLT: <xsl:template> y manipulación de plantillas
5. XSLT: <xsl:output>
6. XSLT: <xsl:value-of>
7. XSLT: <xsl:for-each>
8. XSLT: <xsl:sort>
9. XSLT: <xsl:if>
10. XSLT: <xsl:choose>, <xsl:when> y <xsl:otherwise>
11. XSLT: <xsl:copy> y <xsl:copy-of>
12. XSLT: <xsl:include> y <xsl:import>
13. XSLT: <xsl:variable> y <xsl:param>
14. XSLT: <xsl:element> y <xsl:attribute>
15. XSLT: <xsl:key> y función key()
16. XSLT: <xsl:comment>
17. XSLT: Referencia

3. Bibliografía

1. XSLT

1.1. ¿Qué es XSLT?

- ✓ Las hojas de transformación (o estilo) XSLT realizan la transformación del documento utilizando una o varias reglas de plantilla.
- ✓ Estas reglas de plantilla unidas al documento fuente a transformar alimentan un procesador de XSLT que realiza las transformaciones deseadas poniendo el resultado en un archivo de salida.
- ✓ Actualmente, XSLT es muy usado en la edición web, generando páginas HTML a partir de documentos XML. La unión de XML y XSLT permite **separar contenido y presentación**, aumentando así la productividad.

1. XSLT

1.1. ¿Qué es XSLT?



1. XSLT

1.1. ¿Qué es XSLT?

What is XSLT?

- XSLT stands for XSL Transformations
- XSLT is the most important part of XSL
- XSLT transforms an XML document into another XML document
- XSLT uses XPath to navigate in XML documents
- XSLT is a W3C Recommendation

XSLT = XSL Transformations

XSLT is the most important part of XSL.

XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.

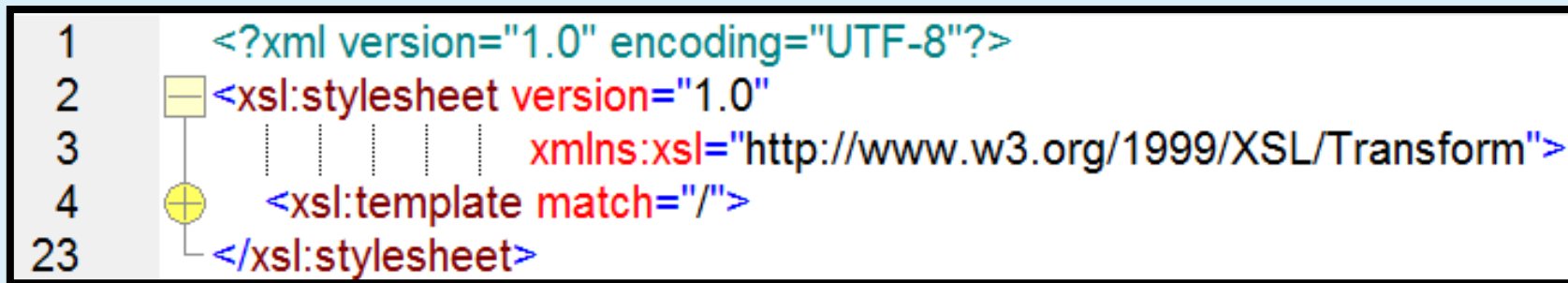
With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.

A common way to describe the transformation process is to say that **XSLT transforms an XML source-tree into an XML result-tree.**

1. XSLT

1.2. Elementos en XSLT

- ✓ En XSLT tenemos elementos de alto nivel, que afectan a toda la hoja(<xsl:stylesheet>), y elementos de bajo nivel, que afectan a nivel de plantilla (<xsl:template>).



The diagram shows a snippet of XSLT code with line numbers on the left. Line 1 is the XML declaration. Line 2 starts the <xsl:stylesheet> element, marked with a yellow box icon. Line 3 is the xmlns:xsl attribute, with dotted lines connecting it to the start of the stylesheet element on line 2. Line 4 starts the <xsl:template> element, marked with a yellow circle icon. Line 23 ends the <xsl:stylesheet> element. The code is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet version="1.0"
3   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4   <xsl:template match="/">
23 </xsl:stylesheet>
```

- ✓ A continuación veremos una clasificación de elementos de XSLT de acuerdo a estos dos niveles, aunque posteriormente profundizaremos en ellos según los vayamos aplicando.

1. XSLT

1.2. Elementos en XSLT

- ✓ Los **elementos de alto nivel** más utilizados y que veremos aquí son:
 - ✓ `<xsl:template>`
 - ✓ `<xsl:variable>` y `<xsl:param>`
 - ✓ `<xsl:output>`
 - ✓ `<xsl:import>` y `<xsl:include>`
 - ✓ `<xsl:key>`
- ✓ **Otros elementos de alto nivel** son:
 - ✓ `<xsl:strip-space>` y `<xsl:preserve-space>` Normalización de espacios.
 - ✓ `<xsl:decimal-format>` Definición de formato a utilizar para conversión de números en cadenas de caracteres.
 - ✓ `<xsl:namespace-alias>` Definición de prefijo alternativo para un espacio de nombres.
 - ✓ `<xsl:attribute.set>` Creación de grupos de atributos que se pueden reutilizar en `<xsl:element>` y `<xsl:copy>` mediante el atributo *use-attribute-sets*.

1. XSLT

1.2. Elementos en XSLT

- ✓ Los **elementos de bajo nivel** afectan a nivel de plantilla. Los más importantes y de los que veremos en su mayoría son:
 - ✓ Instrucciones de manipulación de plantillas
 - ✓ `<xsl:applytemplates>`
 - ✓ `<xsl:call-templates>`
 - ✓ `<xsl:with-param>`
 - ✓ Instrucciones de control
 - ✓ `<xsl:for-each>`, `<xsl:sort>` y `<xsl:if>`
 - ✓ `<xsl:choose>`, `<xsl:when>` y `<xsl:otherwise>`
- ✓ Instrucciones de salida
 - ✓ `<xsl:value-of>`
 - ✓ `<xsl:number>`
 - ✓ `<xsl:element>`
 - ✓ `<xsl:attribute>`
 - ✓ `<xsl:text>`
 - ✓ `<xsl:comment>`
 - ✓ `<xsl:processing-instruction>`
- ✓ Otras instrucciones
 - ✓ `<xsl:copy>` y `<xsl:copy-of>`
 - ✓ `<xsl:apply-imports>`

ÍNDICE

1. XSLT
 1. ¿Qué es XSLT?
 2. Elementos en XSLT
2. **Transformando con XSLT**
 1. Referenciar un fichero XSLT desde un fichero XML
 2. XSLT: <xsl:stylesheet>
 3. XSLT: <xsl:template> y plantillas vacías
 4. XSLT: <xsl:template> y manipulación de plantillas
 5. XSLT: <xsl:output>
 6. XSLT: <xsl:value-of>
 7. XSLT: <xsl:for-each>
 8. XSLT: <xsl:sort>
 9. XSLT: <xsl:if>
 10. XSLT: <xsl:choose>, <xsl:when> y <xsl:otherwise>
 11. XSLT: <xsl:copy> y <xsl:copy-of>
 12. XSLT: <xsl:include> y <xsl:import>
 13. XSLT: <xsl:variable> y <xsl:param>
 14. XSLT: <xsl:element> y <xsl:attribute>
 15. XSLT: <xsl:key> y función key()
 16. XSLT: <xsl:comment>
 17. XSLT: Referencia
3. Bibliografía

2. Transformando con XSLT

2.1. Referenciar un fichero XSLT desde un fichero XML

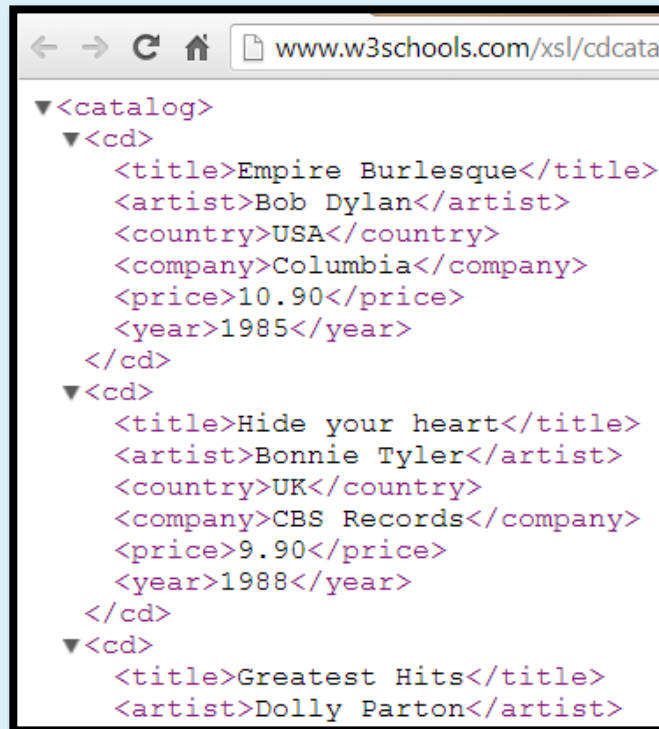
- ✓ Dado el siguiente fichero XML que tiene asociado un DTD:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE catalog SYSTEM "catalog.dtd">
3  <catalog>
4    <cd>
5      <title>Empire Burlesque</title>
6      <artist>Bob Dylan</artist>
7      <country>USA</country>
8      <company>Columbia</company>
9      <price>10.90</price>
10     <year>1985</year>
11   </cd>
12   <cd>
13     <title>Hide your heart</title>
14     <artist>Bonnie Tyler</artist>
15     <country>UK</country>
16     <company>CBS Records</company>
17     <price>9.90</price>
18     <year>1988</year>
19   </cd>
20   <cd>
21     <title>Greatest Hits</title>
22     <artist>Dolly Parton</artist>
23     <country>USA</country>
24     <company>RCA</company>
25     <price>9.90</price>
26     <year>1982</year>
27   </cd>
```

2. Transformando con XSLT

2.1. Referenciar un fichero XSLT desde un fichero XML

- ✓ Si abrimos dicho fichero XML directamente sobre el navegador:



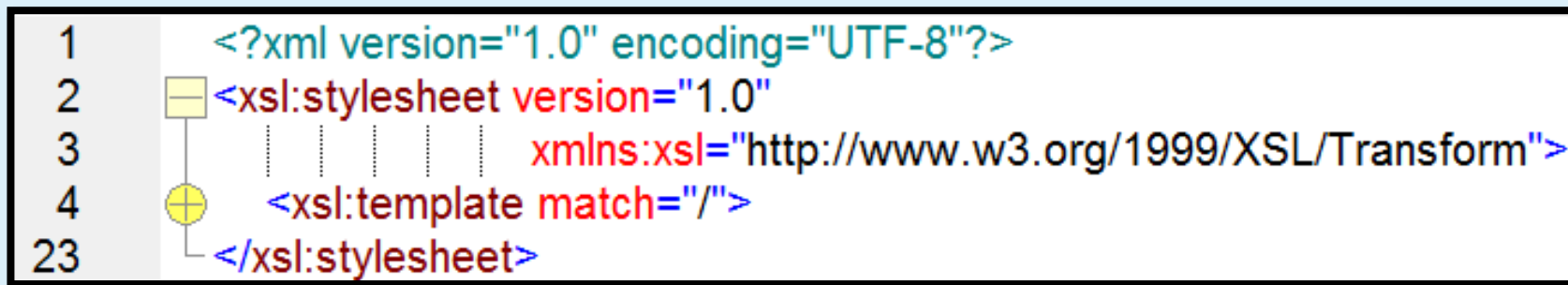
A screenshot of a web browser window displaying an XML document. The address bar shows the URL `www.w3schools.com/xsl/cdcata`. The XML content is as follows:

```
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
    <year>1988</year>
  </cd>
  <cd>
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
```

2. Transformando con XSLT

2.1. Referenciar un fichero XSLT desde un fichero XML

- ✓ Se desea crear un fichero XSLT que permita transformar ese fichero XML en un fichero XHTML. De esta forma cuando el fichero XML (junto con su XSLT) se abra en un navegador, este sea capaz de presentar la información de forma más amigable a como se ha visto en la diapositiva anterior.
- ✓ Dicho fichero tendrá por nombre: "cdcatalog.xml" y ya lo tenemos creado:



The image shows a snippet of XML code within an editor window. On the left, a vertical list of line numbers is displayed: 1, 2, 3, 4, and 23. To the right of these numbers, the corresponding XML code is shown. Line 1 contains the XML declaration: `<?xml version="1.0" encoding="UTF-8"?>`. Line 2 starts an `<xsl:stylesheet>` element with `version="1.0"`. Line 3 continues the `<xsl:stylesheet>` element with `xmlns:xsl="http://www.w3.org/1999/XSL/Transform">`. Line 4 starts an `<xsl:template match="/">` element. Line 23 closes the `<xsl:stylesheet>` element with `</xsl:stylesheet>`. In the left margin, next to line 2, there is a yellow square icon with a black border. Next to line 4, there is a yellow circle icon with a black border and a plus sign inside.

```
1      <?xml version="1.0" encoding="UTF-8"?>
2      <xsl:stylesheet version="1.0"
3          xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4          <xsl:template match="/">
23     </xsl:stylesheet>
```

2. Transformando con XSLT

2.1. Referenciar un fichero XSLT desde un fichero XML

- ✓ Ahora sólo queda referenciar dicho XSLT desde nuestro fichero XML:

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



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE catalog SYSTEM "catalog.dtd">
3 <?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
4 <catalog>
5   <cd>
6     <title>Empire Burlesque</title>
7     <artist>Bob Dylan</artist>
8     <country>USA</country>
9     <company>Columbia</company>
10    <price>10.90</price>
11    <year>1985</year>
12  </cd>
13  <cd>
```

2. Transformando con XSLT

2.1. Referenciar un fichero XSLT desde un fichero XML

- ✓ Cuando abramos dicho fichero XML en el navegador se visualizará así:



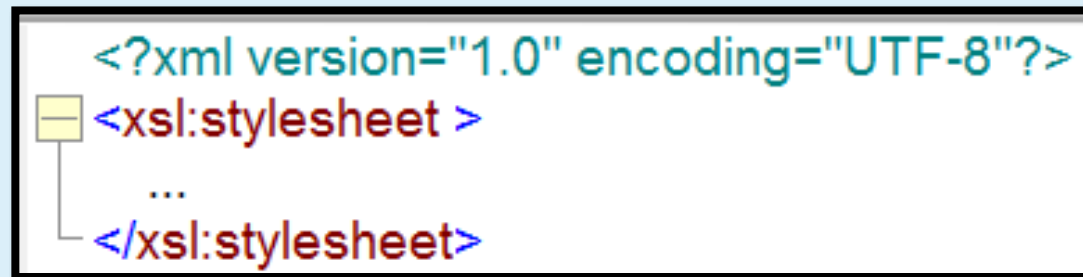
The image shows a 3D-rendered browser window displaying a table titled "My CD Collection". The table has two columns: "Title" and "Artist". The data rows are as follows:

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti

2. Transformando con XSLT

2.2. XSLT: <xsl:stylesheet>

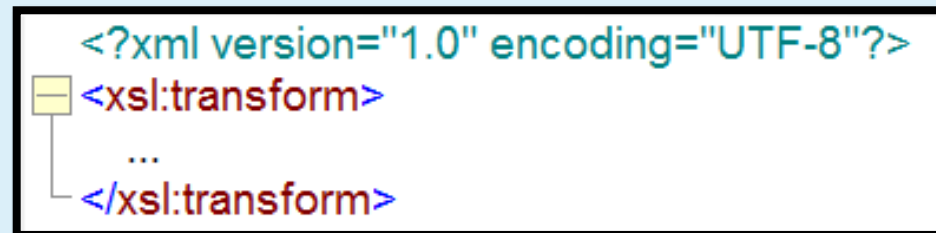
- ✓ Una hoja de transformación XSLT es un fichero XML cuyo nodo raíz es: <xsl:stylesheet>



The diagram shows an XML structure for an XSLT stylesheet. It starts with a root node <?xml version="1.0" encoding="UTF-8"?>. Below this is a yellow box containing the text <xsl:stylesheet>. To the left of this box is a small yellow square with a horizontal line, connected to the root node by a vertical line. Below the <xsl:stylesheet> tag are three dots (...) and then the closing tag </xsl:stylesheet>.

```
<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet>  
...  
</xsl:stylesheet>
```

- ✓ También puede crearse así (pero en clase usaremos la forma anterior):



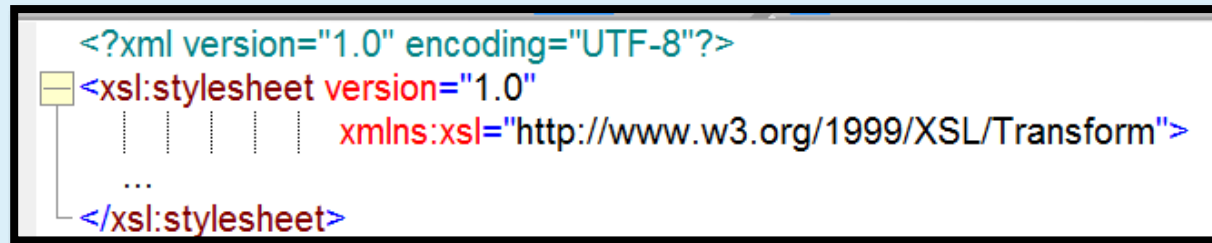
The diagram shows an alternative XML structure for an XSLT transformation. It starts with a root node <?xml version="1.0" encoding="UTF-8"?>. Below this is a yellow box containing the text <xsl:transform>. To the left of this box is a small yellow square with a horizontal line, connected to the root node by a vertical line. Below the <xsl:transform> tag are three dots (...) and then the closing tag </xsl:transform>.

```
<?xml version="1.0" encoding="UTF-8"?>  
<xsl:transform>  
...  
</xsl:transform>
```


2. Transformando con XSLT

2.2. XSLT: <xsl:stylesheet>

- ✓ El nodo raíz de XSLT suele ir acompañado de una serie de atributos interesantes:

A screenshot of an XML editor showing an XSLT stylesheet. On the left, a tree view shows the root element 'xsl:stylesheet' expanded. The main area displays the following code:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  ...
</xsl:stylesheet>
```

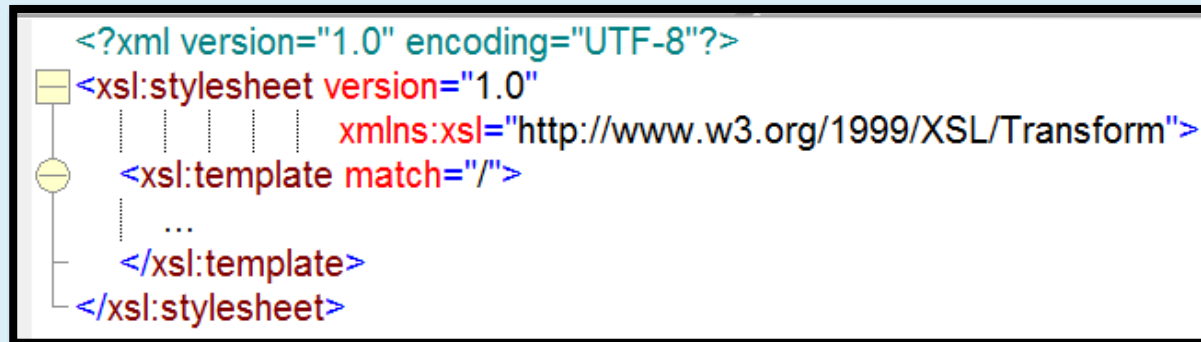
- version --> Especifica la versión de XSLT utilizada en la hoja de estilos.
- xmlns:xsl --> Indica el espacio de nombres utilizado para los elementos de XSLT.

Nota: En clase en principio trabajaremos con la versión 1.0 de XSLT.

2. Transformando con XSLT

2.3. XSLT: <xsl:template> y plantillas vacías

- ✓ Justo después del nodo raíz debe aparecer el elemento <xsl:template>. De lo contrario, el procesador recorrerá todos los nodos del XML recopilando todos los datos del mismo, sin incluir los metadatos de atributos.
- ✓ Este elemento nos permite definir un elemento plantilla dentro del XSL. Dicha plantilla nos permitirá generar la salida formateada.



```
<?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:template>
</xsl:stylesheet>
```

The diagram shows a tree view on the left with a root node (square) and a child node (circle). The code on the right corresponds to the tree structure, with the root node being the <xsl:stylesheet> element and the child node being the <xsl:template> element.

- ✓ El atributo "**match**" indica mediante XPath el nodo al que se aplica la plantilla.

2. Transformando con XSLT

2.3. XSLT: <xsl:template> y plantillas vacías

- ✓ A modo de ejemplo, contamos con un XML que contiene datos de CD musicales, tal como se ve en la imagen.
- ✓ Veamos a continuación el resultado que obtenemos en función del uso que hagamos con las plantillas.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE catalog SYSTEM "DTD/cdcatalog.dtd">

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

<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
```

2. Transformando con XSLT

2.3. XSLT: <xsl:template> y plantillas vacías

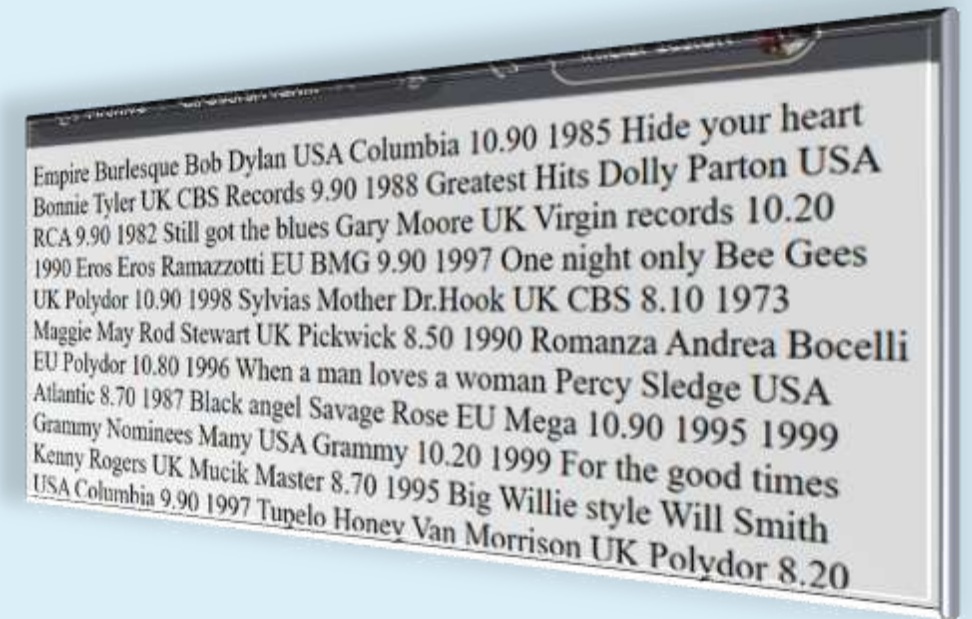
- ✓ Si no aparece ninguna plantilla, el procesador recorre todos los nodos y recopila todos los datos para el archivo de salida, sin incluir los metadatos de atributos. El resultado es un archivo que contiene dichos datos. En este caso un HTML:

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

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

  <xsl:output method="html"/>

</xsl:stylesheet>
```



2. Transformando con XSLT

2.3. XSLT: <xsl:template> y plantillas vacías

- ✓ Si aparece una plantilla que no es el nodo raíz, el procesador hace lo mismo que en el caso anterior, pero al llegar al nodo para el que hemos puesto una plantilla intenta seguir las reglas que contiene. Si esta plantilla no contiene reglas, no recopila nada de ese nodo. En el ejemplo vemos que han desaparecido los datos de “title”:

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

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

  <xsl:template match="title"/>

</xsl:stylesheet>
```



2. Transformando con XSLT

2.3. XSLT: <xsl:template> y plantillas vacías

- ✓ Si aparecen varias plantillas, el procesador sólo atiende a la primera, salvo que especifiquemos lo contrario en la primera plantilla. A continuación, un ejemplo prácticamente igual que el anterior, pero añadiendo una segunda plantilla que es ignorada por el procesador. Vemos que no aparecen los datos de “title” pero siguen apareciendo los datos de “artist”:

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

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

  <xsl:template match="title"/>

  <xsl:template name="artist"/>

</xsl:stylesheet>
```



2. Transformando con XSLT

2.3. XSLT: <xsl:template> y plantillas vacías

- ✓ Otro ejemplo del mismo caso. Ahora tenemos una plantilla para el nodo raíz y algunas más para otros nodos del árbol.
- ✓ El procesador no atenderá a las siguientes plantillas.
- ✓ Dado que la plantilla del nodo raíz no contiene ninguna regla, el procesador no recorrerá nodos ni recopilará ningún dato.

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

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

  <xsl:output method="html"/>

  <xsl:template match="/" />

  <xsl:template match="cd">
    <p>
      Hola mundo
    </p>
  </xsl:template>
```


2. Transformando con XSLT

2.4. XSLT: <xsl:template> y manipulación de plantillas

- ✓ Ahora vamos a añadir algo de contenido a la plantilla del nodo raíz y, lo que es más importante, vamos a aplicar las plantillas existentes para **todos los nodos hijos** con **<xsl:apply-templates />**. Para cada “cd”, se inserta un párrafo con la palabra “Hola”:

```
<xsl:template match="/">
  <html>
    <head><title>cdcatalog.xml</title></head>
    <body>
      <h2>Mi colección de CDs</h2>
      <xsl:apply-templates />
    </body>
  </html>
</xsl:template>

<xsl:template match="cd">
  <p>
    Hola
  </p>
</xsl:template>
```



2. Transformando con XSLT

2.4. XSLT: <xsl:template> y manipulación de plantillas

- ✓ Ahora aplicaremos las plantillas existentes **exclusivamente para el nodo hijo indicado** por el atributo “select” en <xsl:apply-templates />. Si no lo encuentra, no hace nada:

```
<xsl:template match="/">
  <html>
    <head><title>cdcatalog.xml</title></head>
    <body>
      <h2>Mi colección de CDs</h2>
      <xsl:apply-templates select="libro"/>
    </body>
  </html>
</xsl:template>

<xsl:template match="cd">
  <p>
    Hola
  </p>
</xsl:template>
```



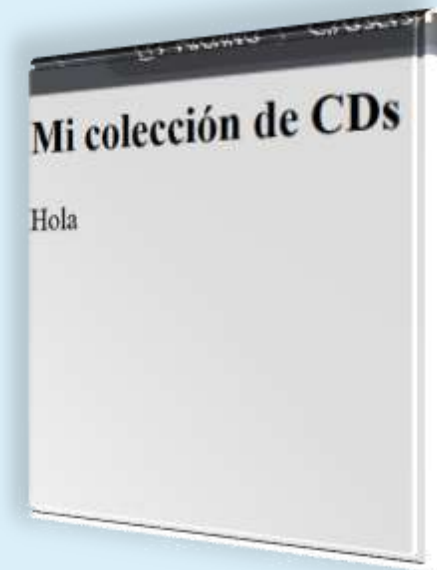
2. Transformando con XSLT

2.4. XSLT: <xsl:template> y manipulación de plantillas

- ✓ Finalmente, podemos invocar una plantilla mediante el elemento `<xsl:call-template name=""/>`. La plantilla invocada contendrá el atributo “name” en lugar de “select”:

```
<xsl:template match="/">
  <html>
    <head><title>cdcatalog.xml</title></head>
    <body>
      <h2>Mi colección de CDs</h2>
      <xsl:call-template name="plantilla1"/>
    </body>
  </html>
</xsl:template>

<xsl:template name="plantilla1">
  <p>
    Hola
  </p>
</xsl:template>
```



2. Transformando con XSLT

2.5. XSLT: <xsl:output>

- ✓ Permite indicar cómo quieres que sea la salida generada.

```
<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
```

```
<xsl:output method="html" version="5.0" encoding="UTF-8" indent="yes"/>
```

- ✓ Es muy útil cuando la salida generada es otro fichero XML.
- ✓ El elemento <xsl:output> tiene distintos atributos aunque los más importantes son los que se muestran a continuación.

2. Transformando con XSLT

2.5. XSLT: <xsl:output>

Attribute	Value	Description
method	xml html text name	Optional. Defines the output format. The default is XML (but if the first child of the root node is <html> and there are no preceding text nodes, then the default is HTML) Netscape 6 only supports "html" and "xml"
version	string	Optional. Sets the W3C version number for the output format (only used with method="html" or method="xml")
encoding	string	Optional. Sets the value of the encoding attribute in the output
doctype-public	string	Optional. Sets the value of the PUBLIC attribute of the DOCTYPE declaration in the output
doctype-system	string	Optional. Sets the value of the SYSTEM attribute of the DOCTYPE declaration in the output
indent	yes no	Optional. "yes" indicates that the output should be indented according to its hierarchic structure. "no" indicates that the output should not be indented according to its hierarchic structure. This attribute is not supported by Netscape 6

2. Transformando con XSLT

2.5. XSLT: <xsl:output>

- ✓ En el ejemplo anterior:



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3   <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
4   <xsl:template match="/">
5     <minicatalog>
6       <xsl:for-each select="catalog/cd[price > 10]">
7         <cd>
8           <xsl:attribute name="year">
9             <xsl:value-of select="year" />
10          </xsl:attribute>
11          <title>
12            <xsl:value-of select="title"/>
13          </title>
14          <artist>
15            <xsl:value-of select="artist"/>
16          </artist>
17        </cd>
18      </xsl:for-each>
19    </minicatalog>
20  </xsl:template>
21 </xsl:stylesheet>
```

2. Transformando con XSLT

2.6. XSLT: <xsl:value-of>

- ✓ Extrae el valor del nodo XML seleccionado.
- ✓ El nodo se indica en el atributo "select" usando XPath.

```
<xsl:value-of select="..." />
```

2. Transformando con XSLT

2.6. XSLT: <xsl:value-of>

- ✓ Vemos un ejemplo aplicado sobre el mismo XML que hemos visto en apartados anteriores.
- ✓ Deseamos seleccionar algunos datos del primer CD y el último CD:

```
<xsl:value-of select="/catalog/cd[1]/title"/>  
<xsl:value-of select="/catalog/cd[1]/artis"/>  
<xsl:value-of select="/catalog/cd[1]/year"/>  
  
<xsl:value-of select="/catalog/cd[last()]/title"/>  
<xsl:value-of select="/catalog/cd[last()]/artis"/>  
<xsl:value-of select="/catalog/cd[last()]/year"/>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>  
2 <!DOCTYPE catalog SYSTEM "catalog.dtd">  
3 <catalog>  
4   <cd>  
5     <title>Empire Burlesque</title>  
6     <artist>Bob Dylan</artist>  
7     <country>USA</country>  
8     <company>Columbia</company>  
9     <price>10.90</price>  
10    <year>1985</year>  
11  </cd>  
12  <cd>  
13    <title>Hide your heart</title>  
14    <artist>Bonnie Tyler</artist>  
15    <country>UK</country>  
16    <company>CBS Records</company>  
17    <price>9.90</price>  
18    <year>1988</year>  
19  </cd>  
20  <cd>  
21    <title>Greatest Hits</title>  
22    <artist>Dolly Parton</artist>  
23    <country>USA</country>  
24    <company>RCA</company>  
25    <price>9.90</price>  
26    <year>1982</year>  
27  </cd>
```

2. Transformando con XSLT

2.6. XSLT: <xsl:value-of>

- ✓ Integramos estos elementos en esta plantilla de ejemplo para ver el resultado:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0"
3      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4      <xsl:template match="/">
5          <html>
6              <head>
7                  <title>CD</title>
8                  <style>
9                      table, th, td { border: 1px solid blue;}
10                 </style>
11             </head>
12             <body>
13                 <h2>My CD Collection</h2>
14                 <table>
15                     <tr>
16                         <th>Title</th>
17                         <th>Artist</th>
18                     </tr>
19
20                     <!-- Aquí pondré instrucciones XSLT para
21                          obtener los datos del XML -->
22
23                 </table>
24             </body>
25         </html>
26     </xsl:template>
27 </xsl:stylesheet>
```


2. Transformando con XSLT

2.6. XSLT: <xsl:value-of>

- ✓ Integramos estos elementos en esta plantilla de ejemplo para ver el resultado:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3   <xsl:template match="/">
4     <html>
5       <head>
6         <title>CD</title>
7         <style>
8           table, th, td { border: 1px solid blue;}
9         </style>
10      </head>
11      <body>
12        <h2>My CD Collection</h2>
13        <table>
14          <tr bgcolor="#9acd32">
15            <th>Title</th>
16            <th>Artist</th>
17            </tr>
```

```
18   <tr>
19     <td>
20       <xsl:value-of select="/catalog/cd[1]/title"/>
21     </td>
22     <td>
23       <xsl:value-of select="/catalog/cd[1]/artist"/>
24     </td>
25   </tr>
26   <tr>
27     <td>
28       <xsl:value-of select="/catalog/cd[last()]/title"/>
29     </td>
30     <td>
31       <xsl:value-of select="/catalog/cd[last()]/artist"/>
32     </td>
33   </tr>
34 </table>
35 </body>
36 </html>
37 </xsl:template>
38 </xsl:stylesheet>
```

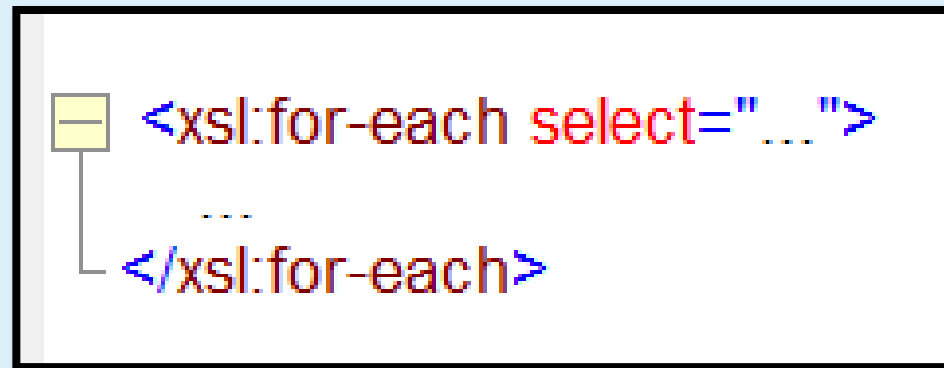


Title	Artist
Empire Burlesque	Bob Dylan
Unchain my heart	Joe Cocker

2. Transformando con XSLT

2.7. XSLT: <xsl:for-each>

- ✓ Permite recorrer un fichero XML usando un bucle for.
- ✓ En el atributo "select" usando XPath se indica a través de qué nodo queremos recorrer.




The diagram shows an XSLT `<xsl:for-each>` element. On the left, a yellow square with a horizontal line represents the element's structure. A vertical line extends from the bottom of this square, branching into two horizontal lines. The top horizontal line points to the opening tag `<xsl:for-each select="...">`, where `select="..."` is highlighted in red. The bottom horizontal line points to the closing tag `</xsl:for-each>`, which is also highlighted in red. The text is displayed in a monospaced font with blue and red color coding for tags and attributes.

2. Transformando con XSLT

2.7. XSLT: <xsl:for-each>


✓ Un para de ejemplos:

a) Para cada “title” obtenemos el valor del nodo actual (“title”).



```
<xsl:for-each select="/catalog/cd/title">  
  <xsl:value-of select="."/>  
</xsl:for-each>
```

b) Para cada “cd”, obtenemos el valor de los nodos “title” y “artist”.



```
<xsl:for-each select="/catalog/cd">  
  <xsl:value-of select="title"/>  
  <xsl:value-of select="artist"/>  
</xsl:for-each>
```

2. Transformando con XSLT

2.7. XSLT: <xsl:for-each>

- ✓ A medida que vamos recorriendo, podemos aplicar distintos tipos de filtros. En este ejemplo seleccionamos el valor de los nodos “title” y “artist” para cada “cd” cuyo valor en el nodo “artist” sea igual a ‘Bob Dylan’:

```
<xsl:for-each select="catalog/cd[artist='Bob Dylan']">
  <tr>
    <td><xsl:value-of select="title"/></td>
    <td><xsl:value-of select="artist"/></td>
  </tr>
</xsl:for-each>
```

Legal filter operators are:

- = (equal)
- != (not equal)
- < less than
- > greater than

2. Transformando con XSLT

2.7. XSLT: <xsl:for-each>

- ✓ Integramos esto en la plantilla anterior:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0"
3      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4      <xsl:template match="/">
5          <html>
6              <head>
7                  <title>CD</title>
8                  <style>
9                      table, th, td { border: 1px solid blue;}
10                 </style>
11             </head>
12             <body>
13                 <h2>My CD Collection</h2>
14                 <table>
15                     <tr>
16                         <th>Title</th>
17                         <th>Artist</th>
18                     </tr>
19
20                     <!-- Aquí pondré instrucciones XSLT para
21                          obtener los datos del XML -->
22
23                 </table>
24             </body>
25         </html>
26     </xsl:template>
27 </xsl:stylesheet>
```

2. Transformando con XSLT

2.7. XSLT: <xsl:for-each>

- ✓ Integramos esto en la plantilla anterior y observamos el resultado:



```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3    <xsl:template match="/">
4      <html>
5        <head>
6          <title>CD</title>
7          <style>
8            table, th, td { border: 1px solid blue;}
9          </style>
10         </head>
11         <body>
12           <h2>My CD Collection</h2>
13           <table>
14             <tr bgcolor="#9acd32">
15               <th>Title</th>
16               <th>Artist</th>
17             </tr>
```

```
18     <xsl:for-each select="catalog/cd">
19       <tr>
20         <td><xsl:value-of select="title"/></td>
21         <td><xsl:value-of select="artist"/></td>
22       </tr>
23     </xsl:for-each>
24   </table>
25 </body>
26 </html>
27 </xsl:template>
28 </xsl:stylesheet>
```

2. Transformando con XSLT

2.7. XSLT: <xsl:for-each>

✓ ¿Qué ocurre en este ejemplo?



CD
Empire Burlesque Bob Dylan USA Columbia 10.901985
Hide your heart Bonnie Tyler UK CBS Records 9.901988
Greatest Hits Dolly Parton USA RCA 9.901982
Still got the blues Gary Moore UK Virgin records 10.201990
Eros Eros Ramazzotti EU BMG 9.901997
One night only Bee Gees UK Polydor 10.901998
Sylvias Mother Dr Hook UK CBS 8.101973
Maggie May Rod Stewart UK Pickwick 8.501990
Romanza Andrea Bocelli EU Polydor 10.801996
When a man loves a woman Percy Sledge USA Atlantic 8.701987

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3    <xsl:template match="/">
4      <html>
5        <head>
6          <title>CD</title>
7          <style>
8            table, th, td { border: 1px solid blue;}
9          </style>
10         </head>
11         <body>
12           <h2>My CD Collection</h2>
13           <table>
14             <tr bgcolor="#9acd32">
15               <th>CD</th>
16             </tr>
17             <xsl:for-each select="catalog/cd">
18               <tr>
19                 <td><xsl:value-of select="."/></td>
20               </tr>
21             </xsl:for-each>
22           </table>
23         </body>
24       </html>
25     </xsl:template>
26   </xsl:stylesheet>
```



2. Transformando con XSLT

2.8. XSLT: <xsl:sort>

- ✓ Permite mostrar al usuario de forma ordenada los datos de salida obtenidos a partir de un elemento <xsl:for-each select="...">.
- ✓ En el atributo "select" usando XPath se indica el nodo a través del cual queremos establecer el orden.

```
<xsl:sort select="..." />
```

```
<xsl:for-each select="catalog/cd">  
  <xsl:sort select="artist"/>  
  <tr>  
    <td><xsl:value-of select="title"/></td>  
    <td><xsl:value-of select="artist"/></td>  
  </tr>  
</xsl:for-each>
```


2. Transformando con XSLT

2.8. XSLT: <xsl:sort>

- ✓ La etiqueta <xsl:sort> soporta los siguiente atributos:

Attribute	Value	Description
select	XPath-expression	Optional. Specifies which node/node-set to sort on
lang	language-code	Optional. Specifies which language is to be used by the sort
data-type	text number qname	Optional. Specifies the data-type of the data to be sorted. Default is "text"
order	ascending descending	Optional. Specifies the sort order. Default is "ascending"
case-order	upper-first lower-first	Optional. Specifies whether upper- or lowercase letters are to be ordered first

```
<xsl:sort select="..." order="descending"/>
```

2. Transformando con XSLT

2.8. XSLT: <xsl:sort>

✓ Integramos esto en la plantilla anterior:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3   <xsl:template match="/">
4     <html>
5       <head>
6         <title>CD</title>
7         <style>
8           table, th, td { border: 1px solid blue;}
9         </style>
10      </head>
11      <body>
12        <h2>My CD Collection</h2>
13        <table>
14          <tr bgcolor="#9acd32">
15            <th>Title</th>
16            <th>Artist</th>
17          </tr>
```

```
18
19   <xsl:for-each select="catalog/cd">
20     <xsl:sort select="artist"/>
21     <tr>
22       <td><xsl:value-of select="title"/></td>
23       <td><xsl:value-of select="artist"/></td>
24     </tr>
25   </xsl:for-each>
26 </table>
27 </body>
28 </html>
29 </xsl:template>
</xsl:stylesheet>
```

2. Transformando con XSLT

2.9. XSLT: <xsl:if>

- ✓ Con este elemento podemos definir sentencias condicionales sencillas con el fin de condicionar la salida de datos obtenida a partir de un elemento <xsl:for-each select="...">.

```
<xsl:if test="expresión condicional">  
  ...  
</xsl:if>
```

Legal operators are:

- = (equal)
- != (not equal)
- < less than
- > greater than

```
<xsl:for-each select="catalog/cd">  
  <xsl:if test="price &gt; 10">  
    <tr>  
      <td>  
        <xsl:value-of select="title"/>  
      </td>  
      <td>  
        <xsl:value-of select="artist"/>  
      </td>  
      <td>  
        <xsl:value-of select="price"/>  
      </td>  
    </tr>  
  </xsl:if>  
</xsl:for-each>
```

2. Transformando con XSLT

2.9. XSLT: <xsl:if>

- ✓ Siguiendo con el ejemplo anterior, para cada “cd”,
- ✓ si se cumple la condición de que el valor de “price” es mayor que 10,
- ✓ se añade a la tabla una fila con los datos de “title”, “artist” y “price”.



Title	Artist	Price
Empire Burlesque	Bob Dylan	10.90
Still got the blues	Gary Moore	10.20
One night only	Bee Gees	10.90

```
<table border="1">
  <tr bgcolor="#9acd32">
    <th>Title</th>
    <th>Artist</th>
    <th>Price</th>
  </tr>
  <xsl:for-each select="catalog/cd">
    <xsl:if test="price &gt; 10">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
        <td><xsl:value-of select="price"/></td>
      </tr>
    </xsl:if>
  </xsl:for-each>
</table>
```

2. Transformando con XSLT

2.9. XSLT: <xsl:if>

- ✓ Siguiendo con el ejemplo anterior:

```
<table border="1">
  <tr bgcolor="#9acd32">
    <th>Title</th>
    <th>Artist</th>
    <th>Price</th>
  </tr>
  <xsl:for-each select="catalog/cd[price > 10]">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
      <td><xsl:value-of select="price"/></td>
    </tr>
  </xsl:for-each>
</table>
```

¿Y en este último ejemplo qué obtenemos?

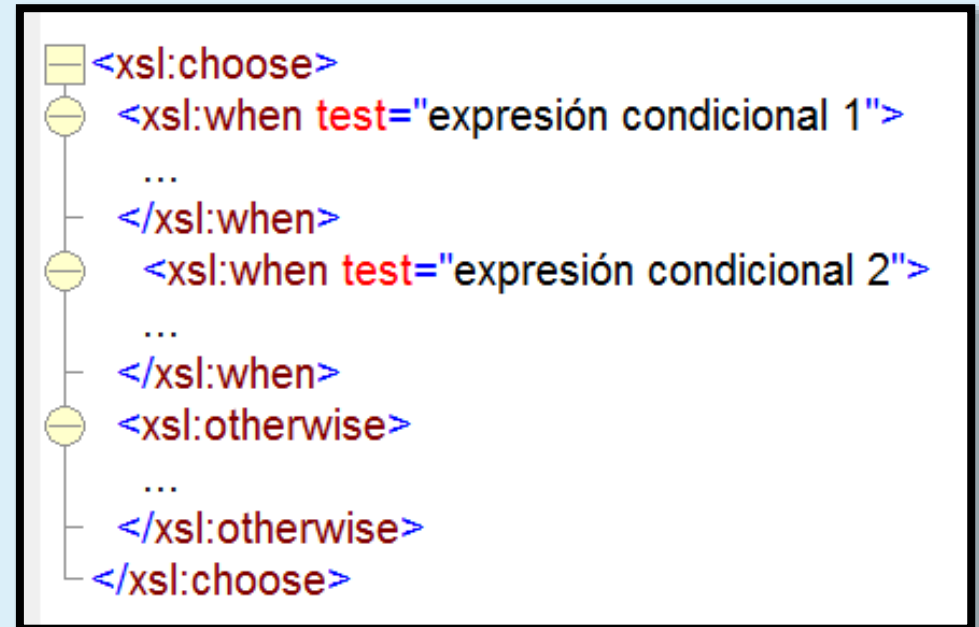
2. Transformando con XSLT

2.10. XSLT: <xsl:choose>, <xsl:when> y <xsl:otherwise>

- ✓ El elemento <xsl:choose> se utiliza junto con los elementos <xsl:when> y <xsl:otherwise> para expresar múltiples sentencias condicionales sencillas, con el fin de condicionar la salida de datos.
- ✓ El procesador recorrerá cada nodo comprobando cada una de las sentencias condicionales.
- ✓ En el momento en que se cumple alguna, ejecuta su contenido y descarta las demás.
- ✓ Si no se cumple ninguna, hace lo indicado en <xsl:otherwise>.

Legal operators are:

- = (equal)
- != (not equal)
- < (less than)
- > (greater than)



2. Transformando con XSLT

2.10. XSLT: <xsl:choose>, <xsl:when> y <xsl:otherwise>

✓ Siguiendo con el ejemplo anterior:

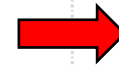
```
<table border="1">
  <tr bgcolor="#9acd32">
    <th>Title</th>
    <th>Artist</th>
  </tr>
  <xsl:for-each select="catalog/cd">
```

```
    <tr>
      <td><xsl:value-of select="title"/></td>
      <xsl:choose>
```

```
        <xsl:when test="price > 10">
          <td bgcolor="#ff00ff">
            <xsl:value-of select="artist"/></td>
          </xsl:when>
```

```
        <xsl:when test="price > 9">
          <td bgcolor="#cccccc">
            <xsl:value-of select="artist"/></td>
          </xsl:when>
```

```
        <xsl:otherwise>
          <td><xsl:value-of select="artist"/></td>
        </xsl:otherwise>
      </xsl:choose>
```



```
    </tr>
```

```
  </xsl:for-each>
```

```
</table>
```

```
</xsl:output>
```



Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr. Hook
Maggie May	Rod Stewart

2. Transformando con XSLT

2.11. XSLT: <xsl:copy> y <xsl:copy-of>

- ✓ Estos elementos permiten copiar información al árbol de resultado.
- ✓ <xsl:copy> hasta XSLT 3.0 solo soporta el atributo “use-attribute-sets” (opcional). Realiza una copia superficial del nodo actual (etiquetas de apertura y cierre), es decir, sin incluir nodos hijos, atributos o contenido del mismo, salvo que incluyamos <xsl:apply-templates/> como hijo:
- ✓ Más información y ejemplos en:
 - ✓ [O'Reilly](#)
 - ✓ [XSLTDEV](#)
 - ✓ [SAXONICA](#)
 - ✓ [w3schools](#)

```
<xsl:template match="message">  
  <xsl:copy>  
    <xsl:apply-templates/>  
  </xsl:copy>  
</xsl:template>
```


2. Transformando con XSLT

2.11. XSLT: <xsl:copy> y <xsl:copy-of>

- ✓ Estos elementos permiten copiar información al árbol de resultado.
- ✓ **<xsl:copy-of>** hasta XSLT 3.0 solo soporta el atributo “select” (obligatorio), el cual puede apuntar a:
 - ✓ Un fragmento de árbol → Copia el fragmento completo.
 - ✓ Un conjunto de nodos → Copia todos los nodos íntegros, incluyendo nodos hijos y nodos atributo.
 - ✓ Si “select” contiene otra cosa distinta de las dos anteriores, se convierte a una string que se incorpora al árbol de resultado.
- ✓ El siguiente ejemplo copia el nodo actual (documento de entrada) al árbol de salida:
- ✓ Más información y ejemplos en:
 - ✓ [O'Reilly](#)
 - ✓ [XSLTDEV](#)
 - ✓ [SAXONICA](#)
 - ✓ [w3schools](#)

```
<xsl:template match="/">
  <xsl:copy-of select="."/>
</xsl:template>
```

2. Transformando con XSLT

2.12. XSLT: <xsl:include> y <xsl:import>

- ✓ Estos elementos sirven para incluir, en una hoja, estilos procedentes de otra hoja.
- ✓ **<xsl:include>** permite separar las transformaciones comunes en una hoja separada e incluir las plantillas que contiene en cualquier punto de otras hojas de estilos.
- ✓ Todas las plantillas incluidas con **<xsl:include>** tienen la misma prioridad que las plantillas ya existentes en la hoja de estilos receptora.
- ✓ [Aquí](#) un ejemplo de aplicación de <xsl:include> en cualquier parte del XSLT.
- ✓ **<xsl:import>** también permite importar las plantillas encontradas en otra hoja de estilos. Sin embargo, sólo puede aparecer al principio del documento.
- ✓ Las plantillas incorporadas con **<xsl:import>** tienen una prioridad inferior a las plantillas ya existentes en la hoja de estilos receptora.
- ✓ [Aquí](#) un ejemplo de aplicación de <xsl:import> al principio del XSLT.

2. Transformando con XSLT

2.13. XSLT: <xsl:variable> y <xsl:param>

- ✓ Estos elementos permiten crear una variable y asignarle un contenido.
- ✓ La única diferencia entre ellos es <xsl:variable> no permite cambiar su valor o contenido, mientras que <xsl:param> si permite cambiarlo con <xsl:with-param>, como veremos a continuación.
- ✓ El atributo name contendrá el nombre de la variable o parámetro, mientras que el atributo select es opcional y permite asignar un valor de acuerdo a una expresión.

```
<xsl:variable  
  name="name"  
  select="expression">  
  
  <!-- Content:template -->  
  
</xsl:variable>
```

```
<xsl:param  
  name="name"  
  select="expression">  
  
  <!-- Content:template -->  
  
</xsl:param>
```

2. Transformando con XSLT

2.13. XSLT: <xsl:variable> y <xsl:param>

- ✓ Si aparece el atributo “select”, no se permite incluir contenido.
- ✓ Si contiene una cadena literal, irá insertada entre comillas distintas a las que encierran el valor del atributo:

```
<xsl:variable name="color" select="'red'"/>
```

```
<xsl:variable name="color" select='"red"'/>
```

- ✓ Si no aparece el atributo “select” ni hay contenido, el valor será una cadena vacía Ejemplo:

```
<xsl:variable name="j" />
```

2. Transformando con XSLT

2.13. XSLT: <xsl:variable> y <xsl:param>

- ✓ Ejemplo de uso de una variable para contener HTML e insertarlo posteriormente.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:variable name="header">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
  </xsl:variable>

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

2. Transformando con XSLT

2.13. XSLT: <xsl:variable> y <xsl:param>

- ✓ Ahora se inserta el contenido de la variable por medio del elemento <xsl:copy-of>.
- ✓ El procesador recuperará el valor de la variable y lo insertará dentro de la tabla del HTML.

```
</xsl:variable>
```

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

```
  <html>
```

```
  <body>
```

```
    <table border="1">
```

```
    <xsl:copy-of select="$header" />
    <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
```

```
      </tr>
```

```
    </xsl:for-each>
```

```
  </table>
```

```
  </body>
```

```
  </html>
```

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

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

2. Transformando con XSLT

2.13. XSLT: <xsl:variable> y <xsl:param>

- ✓ Al igual que <xsl:variable>, <xsl:param> puede almacenar tanto datos como una estructura de datos

```
<xsl:param name="bgColor" select="'blue'"/>
<xsl:param name="width" select="150"/>
<xsl:param name="content"/>
```

```
<xsl:param name="bgColor">
  <xsl:text>blue</xsl:text>
</xsl:param>
<xsl:param name="width">
  <xsl:value-of select="7+8"/>
  <xsl:text>0</xsl:text>
</xsl:param>
<xsl:param name="content"/>
```

- ✓ Sin embargo, <xsl:param> permite cambiar la información después de haberlo definido, motivo por el cual también tienen definirlo vacío, sin contener información.

```
<xsl:param name="content"/>
```

2. Transformando con XSLT

2.13. XSLT: <xsl:variable> y <xsl:param>

- ✓ A modo de ejemplo, podemos definir 4 parámetros para almacenar coordenadas de varios tipos de cajas, asignando o no un valor por defecto:

```
<xsl:param name="startX" select="0"/>
<xsl:param name="startY" select="0"/>
<xsl:param name="endX" select="0"/>
<xsl:param name="endY" select="0"/>
```

- ✓ Posteriormente, podemos cambiar los valores de los parámetros mediante <xsl:with-param> en el momento de invocar una plantilla con <xsl:call-template>, o al aplicar plantillas con <xsl:apply-templates>.

```
<xsl:call-template name="draw-box">
  <xsl:with-param name="startX" select="50"/>
  <xsl:with-param name="startY" select="50"/>
  <xsl:with-param name="endX" select="97"/>
  <xsl:with-param name="endY" select="144"/>
</xsl:call-template>
```


2. Transformando con XSLT

2.13. XSLT: <xsl:variable> y <xsl:param>

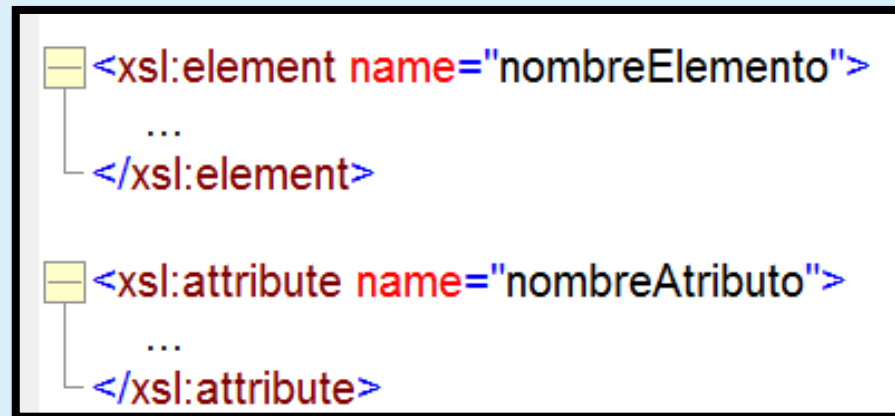
Observaciones sobre <xsl:param>:

- ✓ Si se definen parámetros <xsl:param> dentro de una plantilla, deben ser lo primero que aparezca en ella.
- ✓ Si se aplican o invocan plantillas que no asignan valores a los parámetros, se utilizan los valores por defecto. Si no están aplicados esos valores por defecto, se toma como valor por defecto una string vacía ("").

2. Transformando con XSLT

2.14. XSLT: <xsl:element> y <xsl:attribute>

- ✓ Permiten crear nuevos elementos y nuevos atributos XML



The diagram illustrates the syntax for creating new XML elements and attributes using XSLT. It is divided into two sections, each with a yellow square icon containing a minus sign. The first section shows the declaration of a new element: `<xsl:element name="nombreElemento">` followed by an ellipsis and then `</xsl:element>`. The second section shows the declaration of a new attribute: `<xsl:attribute name="nombreAtributo">` followed by an ellipsis and then `</xsl:attribute>`. The text is color-coded: 'xsl:' is blue, 'element' or 'attribute' is red, 'name=' is blue, and the attribute value is in quotes.

```
<xsl:element name="nombreElemento">
...
</xsl:element>

<xsl:attribute name="nombreAtributo">
...
</xsl:attribute>
```

2. Transformando con XSLT

2.14. XSLT: <xsl:element> y <xsl:attribute>

- ✓ Partiendo del fichero XML anterior vamos a crear un nuevo fichero XML:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE catalog SYSTEM "catalog.dtd">
3 <?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
4 <catalog>
5   <cd>
6     <title>Empire Burlesque</title>
7     <artist>Bob Dylan</artist>
8     <country>USA</country>
9     <company>Columbia</company>
10    <price>10.90</price>
11    <year>1985</year>
12  </cd>
13  <cd>
```



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <minicatalog>
3   <cd year="1985">
4     <title>Empire Burlesque</title>
5     <artist>Bob Dylan</artist>
6   </cd>
7   <cd year="1990">
8     <title>Still got the blues</title>
9     <artist>Gary Moore</artist>
10  </cd>
11  ...
12 </minicatalog>
```

2. Transformando con XSLT

2.14. XSLT: <xsl:element> y <xsl:attribute>

- ✓ Partiendo del fichero XML anterior vamos a crear un nuevo fichero XML:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3    <xsl:template match="/">
4      <xsl:element name="minicatalog">
5        <xsl:for-each select="catalog/cd[price > 10]">
6          <xsl:element name="cd">
7            <xsl:attribute name="year">
8              <xsl:value-of select="year" />
9            </xsl:attribute>
10           <xsl:element name="title">
11             <xsl:value-of select="title" />
12           </xsl:element>
13           <xsl:element name="artist">
14             <xsl:value-of select="artist" />
15           </xsl:element>
16         </xsl:element>
17       </xsl:for-each>
18     </xsl:element>
19   </xsl:template>
20 </xsl:stylesheet>
```

2. Transformando con XSLT

2.14. XSLT: <xsl:element> y <xsl:attribute>

✓ ¿Otra forma?

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3    <xsl:template match="/">
4      <minicatalog>
5        <xsl:for-each select="catalog/cd[price > 10]">
6          <cd>
7            <xsl:attribute name="year">
8              <xsl:value-of select="year" />
9            </xsl:attribute>
10           <title>
11             <xsl:value-of select="title"/>
12           </title>
13          <artist>
14            <xsl:value-of select="artist"/>
15          </artist>
16        </cd>
17      </xsl:for-each>
18    </minicatalog>
19  </xsl:template>
20 </xsl:stylesheet>
```

2. Transformando con XSLT

2.15. XSLT: <xsl:key> y función key()

- ✓ <xsl:key> es un elemento de alto nivel cuya función es declarar un elemento como clave, es decir, indexarlo
 - ✓ un atributo “name” para el nombre de la clave,
 - ✓ un atributo “match” para seleccionar los nodos a indexar mediante Xpath
 - ✓ y un atributo “use” para especificar mediante Xpath la propiedad de los nodos indexados que va a ser utilizada para recuperar los nodos del índice.

```
<xsl:key  
  name="name"  
  match="pattern"  
  use="expression"/>
```

2. Transformando con XSLT

2.15. XSLT: <xsl:key> y función key()

- ✓ Suponiendo que tenemos un archivo persons.xml:

```
<persons>
  <person name="Tarzan" id="050676"/>
  <person name="Donald" id="070754"/>
  <person name="Dolly" id="231256"/>
</persons>
```

- ✓ Podemos definir una clave en un XSL de la siguiente manera:

```
<xsl:key name="preg" match="person" use="@id"/>
```

2. Transformando con XSLT

2.15. XSLT: <xsl:key> y función key()

- ✓ La función **key(string, object)** devuelve un conjunto de nodos del documento a partir de un índice creado mediante <xsl:key>. Para ello se basa en dos parámetros obligatorios:
 - ✓ string. Indica el nombre de la clave,
 - ✓ object. Especifica el valor que debe tener el parámetro de búsqueda utilizado para indexar la clave (atributo “use”).

```
node-set key(string, object)
```


2. Transformando con XSLT

2.15. XSLT: <xsl:key> y función key()

- ✓ Para encontrar a la persona con id = "050676":

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" />
  <xsl:template match="/">
    <xsl:for-each select="key('preg','050676')">
      <p>
        Id: <xsl:value-of select="@id"/><br />
        Name: <xsl:value-of select="@name"/>
      </p>
    </xsl:for-each>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

2. Transformando con XSLT

2.16. XSLT: <xsl:comment>

✓ Los comentarios en XSLT puede hacer de dos formas:

a) Usando los comentarios de XML:

```
<!-- Esto es un comentario -->
```

b) Usando el elemento <xsl:comment>:

```
<xsl:comment>Esto es otro comentario</xsl:comment>
```

2. Transformando con XSLT

2.16. XSLT: Referencia

- ✓ XSLT define más elementos de los explicados aquí. Puedes acceder a la referencia completa desde:

https://www.w3schools.com/xml/xsl_elementref.asp

- ✓ Asimismo, XSLT define una serie de funciones predefinidas para realizar operaciones avanzadas sobre los nodos. Puedes acceder a la referencia completa desde:

https://www.w3schools.com/xml/xsl_functions.asp

ÍNDICE

1. XSLT
 1. ¿Qué es XSLT?
 2. Elementos en XSLT
2. Transformando con XSLT
 1. Referenciar un fichero XSLT desde un fichero XML
 2. XSLT: <xsl:stylesheet>
 3. XSLT: <xsl:template> y plantillas vacías
 4. XSLT: <xsl:template> y manipulación de plantillas
 5. XSLT: <xsl:output>
 6. XSLT: <xsl:value-of>
 7. XSLT: <xsl:for-each>
 8. XSLT: <xsl:sort>
 9. XSLT: <xsl:if>
 10. XSLT: <xsl:choose>, <xsl:when> y <xsl:otherwise>
 11. XSLT: <xsl:copy> y <xsl:copy-of>
 12. XSLT: <xsl:include> y <xsl:import>
 13. XSLT: <xsl:variable> y <xsl:param>
 14. XSLT: <xsl:element> y <xsl:attribute>
 15. XSLT: <xsl:key> y función key()
 16. XSLT: <xsl:comment>
 17. XSLT: Referencia

3. Bibliografía

3. Bibliografía

- ✓ **Lenguajes de Marcas y Sistemas de Gestión de Información**
Sánchez, F. J. y otros. Editorial: RA-MA.
- ✓ **w3schools.com: XML Tutorial**
<https://www.w3schools.com/xml/>
- ✓ **<iXML>: Curso de introducción a XML de la Universidad de Alicante**
<http://ixml.uaedf.ua.es/course>

Unidad 5 – Parte 2

Conversión y adaptación de documentos XML

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN