

Unidad 6

Conversión y adaptación de documentos XML

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

ÍNDICE

1. Introducción
2. XPath: Conceptos esenciales
 1. Antes de nada
 2. Sintaxis XPath
 3. Operadores XPath
 4. Funciones XPath
 5. Más ejemplos
3. XSLT
 1. Referenciar un fichero XSLT desde un fichero XML
 2. XSLT: <xsl:stylesheet>
 3. XSLT <xsl:template>
 4. XSLT <xsl:value-of>
 5. XSLT <xsl:for-each>
 6. XSLT <xsl:sort>
 7. XSLT <xsl:if>
 8. XSLT <xsl:choose>
 9. XSLT <xsl:apply-templates>
 10. XSLT: <xsl:element> y <xsl:attribute>
 11. XSLT: <xsl:output>
 12. XSLT: <xsl:comment>
 13. XSLT: Referencia
4. Bibliografía

ÍNDICE

1. **Introducción**
2. **XPath: Conceptos esenciales**
 1. Antes de nada
 2. Sintaxis XPath
 3. Operadores XPath
 4. Funciones XPath
 5. Más ejemplos
3. **XSLT**
 1. Referenciar un fichero XSLT desde un fichero XML
 2. XSLT: <xsl:stylesheet>
 3. XSLT <xsl:template>
 4. XSLT <xsl:value-of>
 5. XSLT <xsl:for-each>
 6. XSLT <xsl:sort>
 7. XSLT <xsl:if>
 8. XSLT <xsl:choose>
 9. XSLT <xsl:apply-templates>
 10. XSLT: <xsl:element> y <xsl:attribute>
 11. XSLT: <xsl:output>
 12. XSLT: <xsl:comment>
 13. XSLT: Referencia
4. **Bibliografía**

1. Introducción

- ✓ XSL (*Extensible Stylesheet Language*) es **una familia de lenguajes** basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera, debe ser transformada o formateada para su presentación en un medio (pantalla, impresora, etc.). Puede considerarse que XSL es el CSS de XML.
- ✓ XSL permite tomar pleno control sobre los datos, pudiendo establecer criterios sobre qué datos ver y en qué orden visualizarlos, estableciendo filtros y definiendo formatos de salida para su presentación.
- ✓ Extensible Stylesheet Language (XSL) es una recomendación del W3C:

<https://www.w3.org/TR/xsl/>

1. Introducción

CSS = Style Sheets for HTML

HTML uses predefined tags, and the meaning of each tag is **well understood**.

The <table> tag in HTML defines a table - and a browser knows **how to display it**.

Adding styles to HTML elements are simple. Telling a browser to display an element in a special font or color, is easy with CSS.

XSL = Style Sheets for XML

XML does not use predefined tags (we can use any tag-names we like), and therefore the meaning of each tag is **not well understood**.

A <table> tag could mean an HTML table, a piece of furniture, or something else - and a browser **does not know how to display it**.

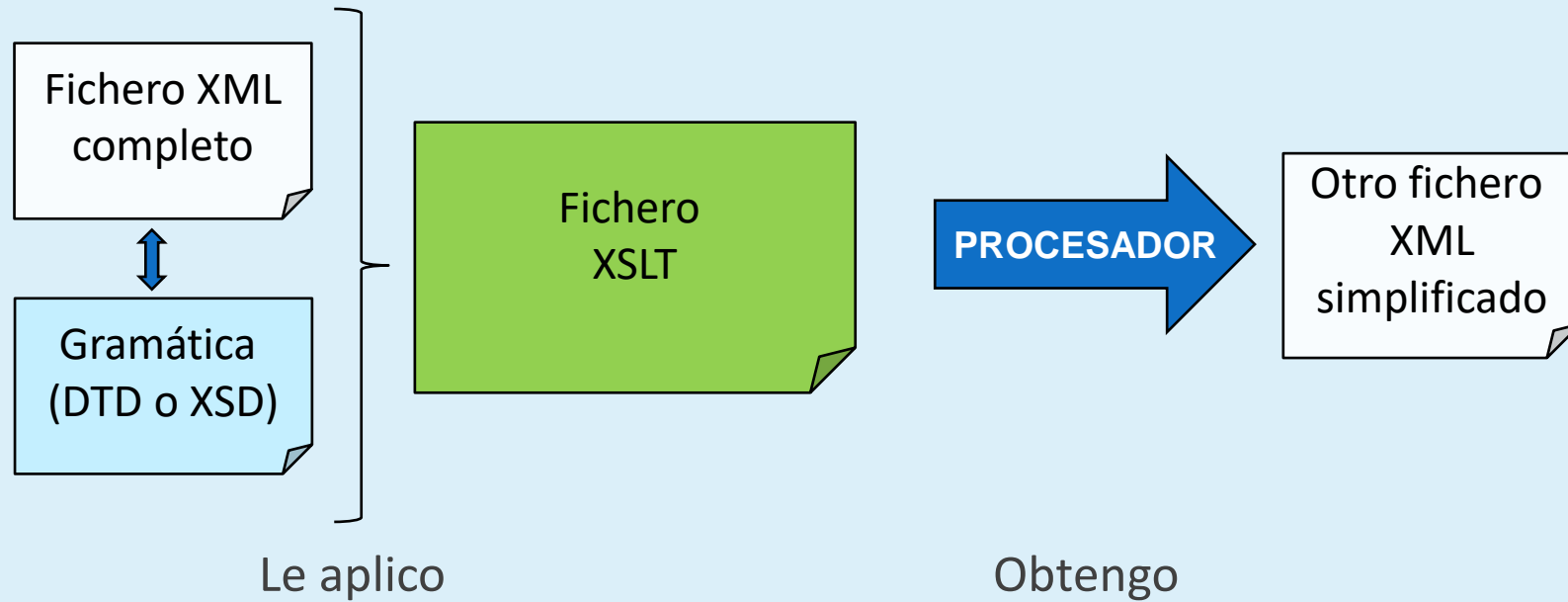
XSL describes how the XML document should be displayed!

1. Introducción

- ✓ XSL es una familia de lenguajes formada por:
 - **XSLT:** permite convertir documentos XML de una sintaxis a otra (por ejemplo, de un XML a otro XML simplificado o de un XML a un documento HTML).
 - **XSL-FO:** permite especificar el formato visual con el cual se quiere presentar un documento XML, es usado principalmente para generar documentos PDF o documento que serán enviado a la impresora directamente.
- ✓ XSLT y XSL-FO usan internamente a su vez:
 - **XPath:** permite acceder o referirse a porciones de un documento XML.

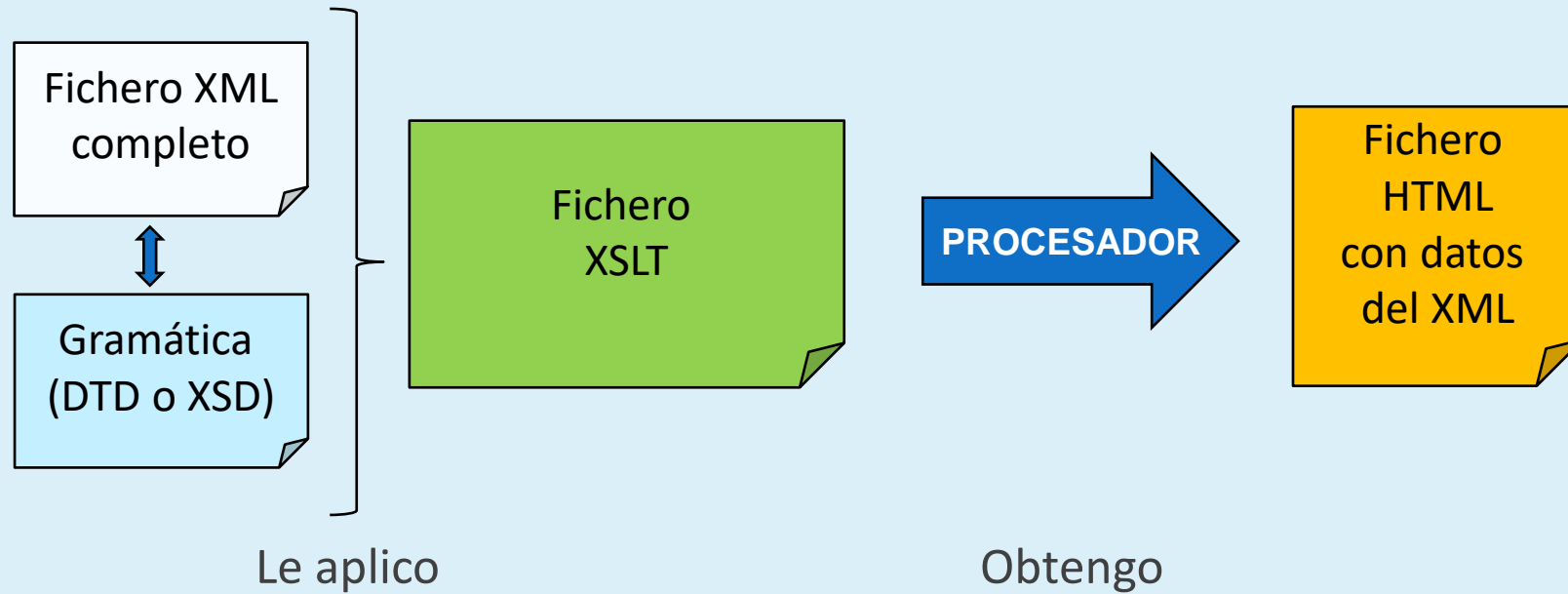
1. Introducción

- ✓ XSLT y XPath: XML Simplificado



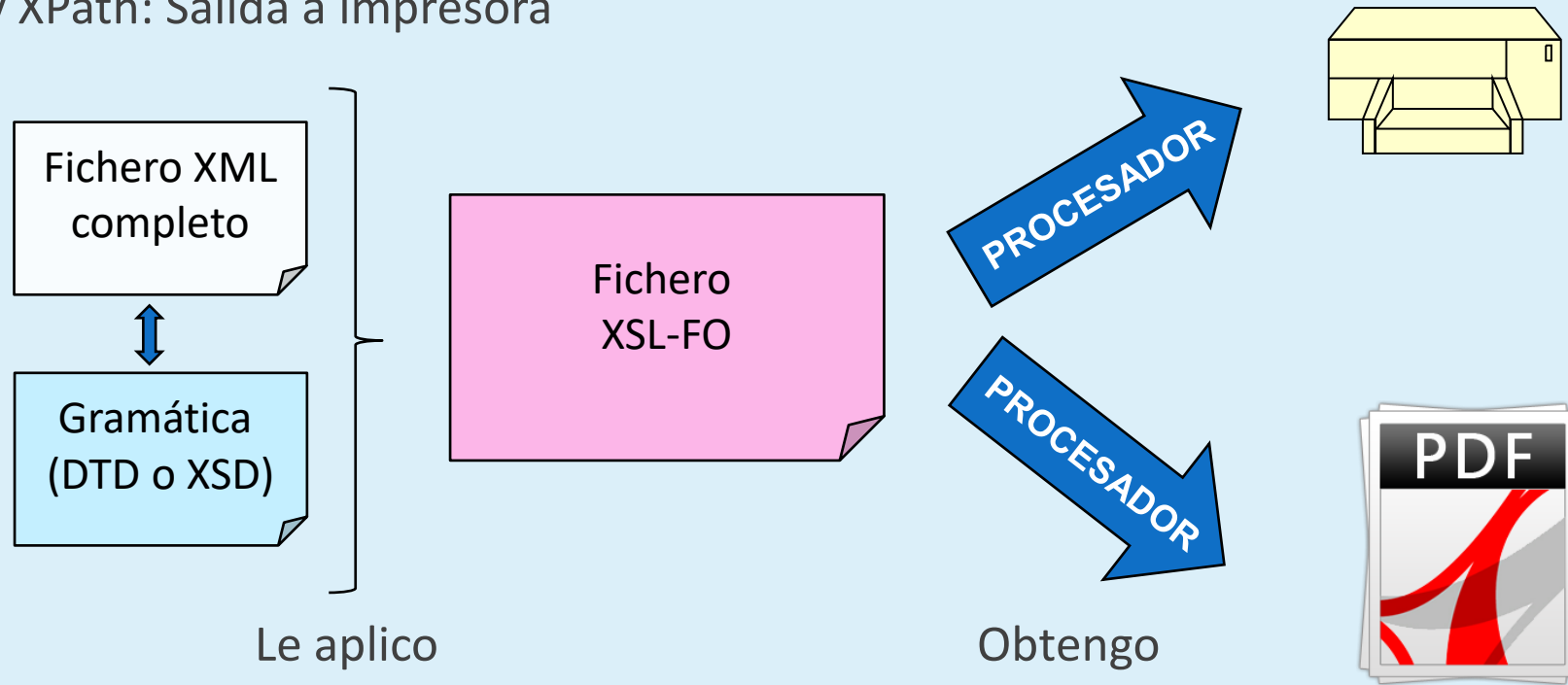
1. Introducción

- ✓ XSLT y XPath: HTML



1. Introducción

- ✓ XSL-FO y XPath: Salida a impresora



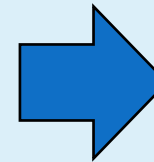
1. Introducción

✓ Ejemplo 1: XSLT y XPath

```
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>
```



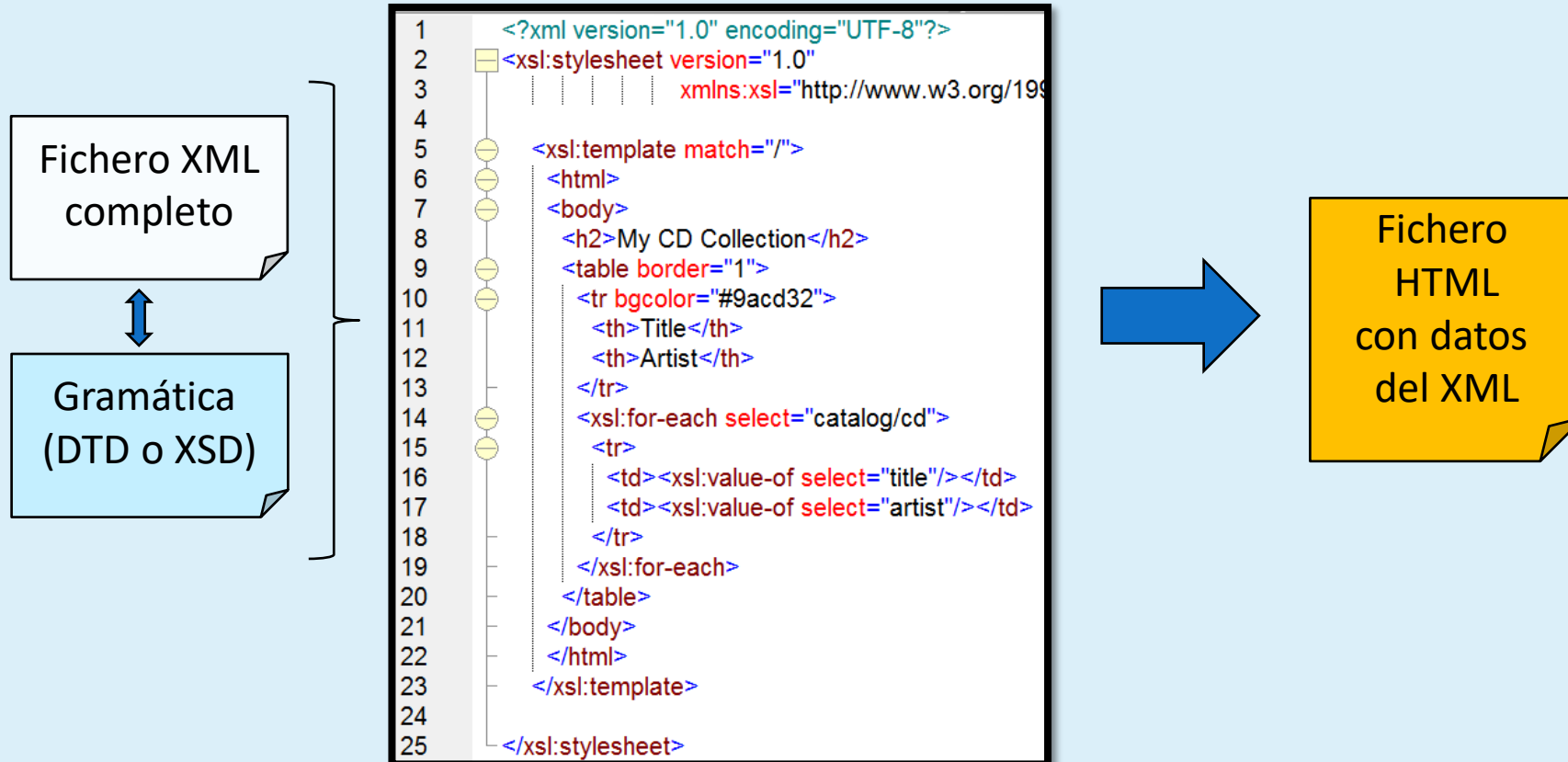
Fichero
XSLT



Fichero
HTML
con datos
del XML

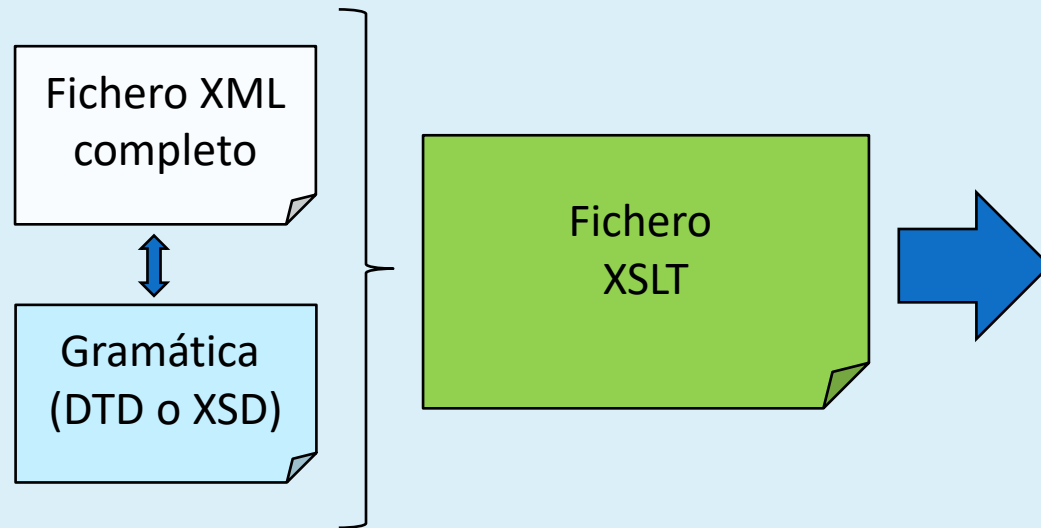
1. Introducción

✓ Ejemplo 1: XSLT y XPath



1. Introducción

✓ Ejemplo 1: XSLT y XPath



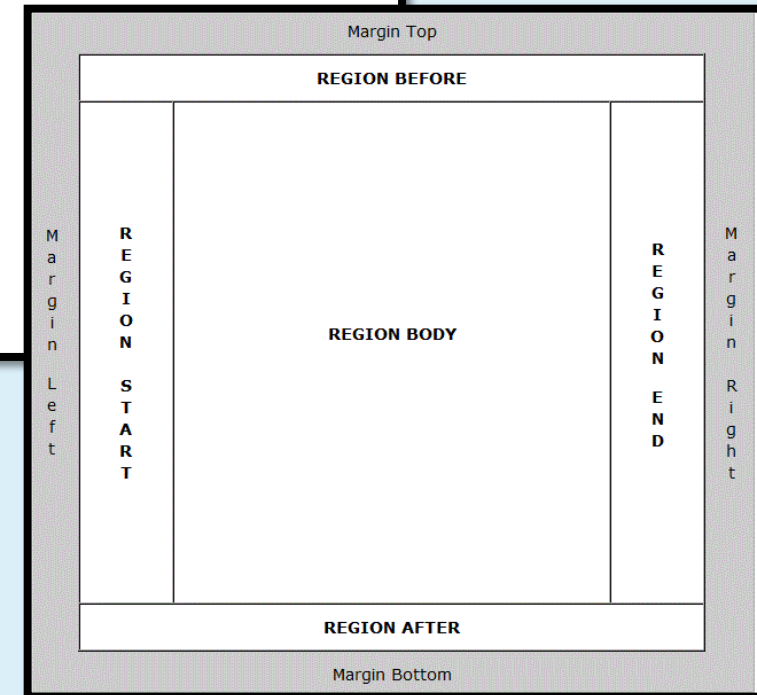
My CD Collection

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
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers

1. Introducción

✓ Ejemplo 2: XSL-FO y XPath

```
1 <fo:simple-page-master master-name="A4" page-width="297mm"
2   page-height="210mm" margin-top="1cm" margin-bottom="1cm"
3   margin-left="1cm" margin-right="1cm">
4   <fo:region-body margin="3cm"/>
5   <fo:region-before extent="2cm"/>
6   <fo:region-after extent="2cm"/>
7   <fo:region-start extent="2cm"/>
8   <fo:region-end extent="2cm"/>
9 </fo:simple-page-master>
```

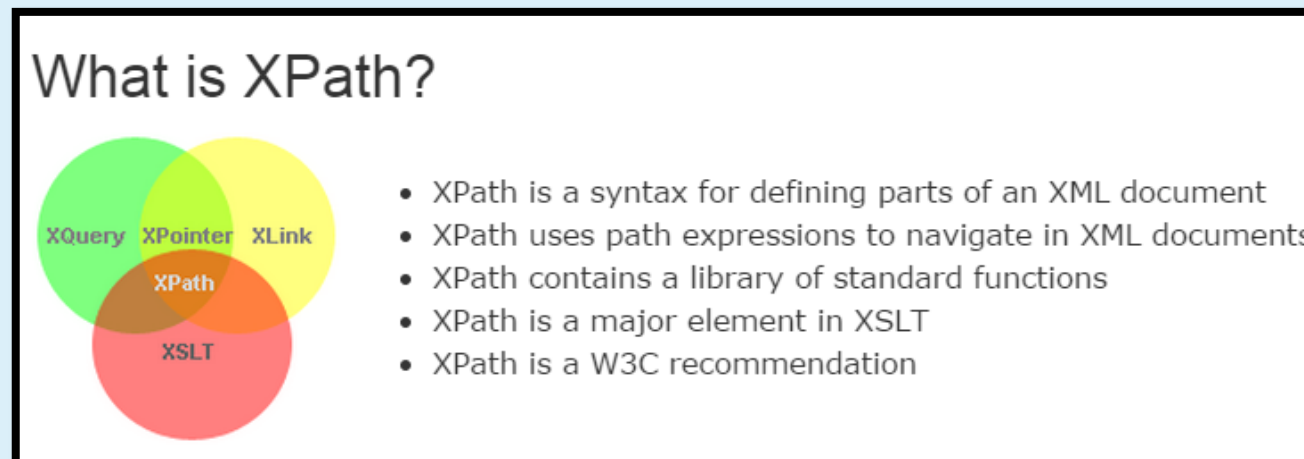


ÍNDICE

1. Introducción
2. **XPath: Conceptos esenciales**
 1. Antes de nada
 2. Sintaxis XPath
 3. Operadores XPath
 4. Funciones XPath
 5. Más ejemplos
3. XSLT
 1. Referenciar un fichero XSLT desde un fichero XML
 2. XSLT: <xsl:stylesheet>
 3. XSLT <xsl:template>
 4. XSLT <xsl:value-of>
 5. XSLT <xsl:for-each>
 6. XSLT <xsl:sort>
 7. XSLT <xsl:if>
 8. XSLT <xsl:choose>
 9. XSLT <xsl:apply-templates>
 10. XSLT: <xsl:element> y <xsl:attribute>
 11. XSLT: <xsl:output>
 12. XSLT: <xsl:comment>
 13. XSLT: Referencia
4. Bibliografía

2. XPATH: Conceptos esenciales

- ✓ XPath es un lenguaje de expresión utilizado para referenciar nodos de información en un conjunto de datos XML.
- ✓ Con XPath podremos seleccionar y hacer referencia a texto, elementos, atributos y cualquier otra información contenida dentro de un fichero XML.
- ✓ XPath por sí solo no tiene "sentido". Siempre se usa en combinación con otra tecnología. Nosotros lo usaremos con XSLT y XQuery.



2. XPATH: Conceptos esenciales

- ✓ XPath es una recomendación del W3C:

<https://www.w3.org/TR/xpath/all/>


- ✓ Existen varias versiones de XPath:
 - noviembre de 1999: [XML Path Language \(XPath\) 1.0](#)
 - enero de 2007: [XML Path Language \(XPath\) 2.0](#)
 - diciembre de 2010: [XML Path Language \(XPath\) 2.0 \(2ª edición\)](#)
 - abril de 2014: [XML Path Language \(XPath\) 3.0](#)
 - marzo de 2017: [XML Path Language \(XPath\) 3.1](#)

2. XPATH: Conceptos esenciales

2.1. Antes de nada

- ✓ Dado el siguiente fichero XML:

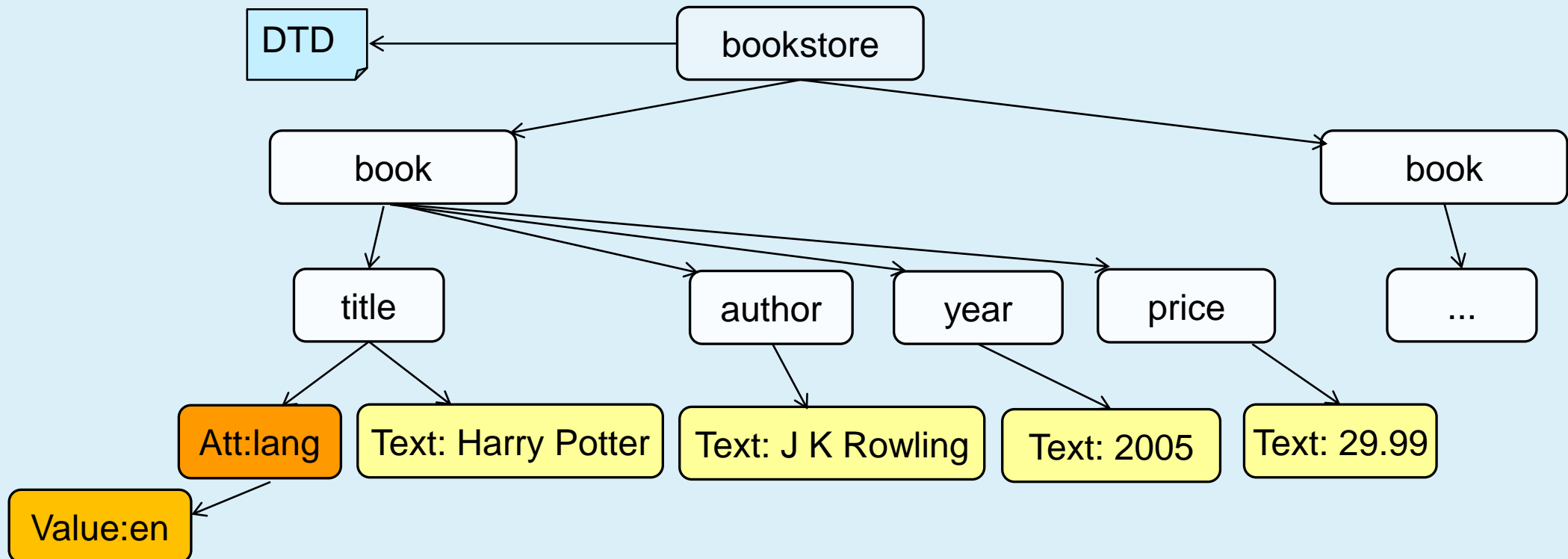
```
1      <?xml version="1.0" encoding="UTF-8"?>
2      <!DOCTYPE bookstore SYSTEM "dtd/bookstore.dtd">
3      <bookstore>
4          <book>
5              <title lang="en">Harry Potter</title>
6              <author>J K Rowling</author>
7              <year>2005</year>
8              <price>29.99</price>
9          </book>
10         <book>
11             <title lang="en">LOR</title>
12             <author>J R R Tolkien</author>
13             <year>2004</year>
14             <price>50.55</price>
15         </book>
16     </bookstore>
```



2. XPATH: Conceptos esenciales

2.1. Antes de nada

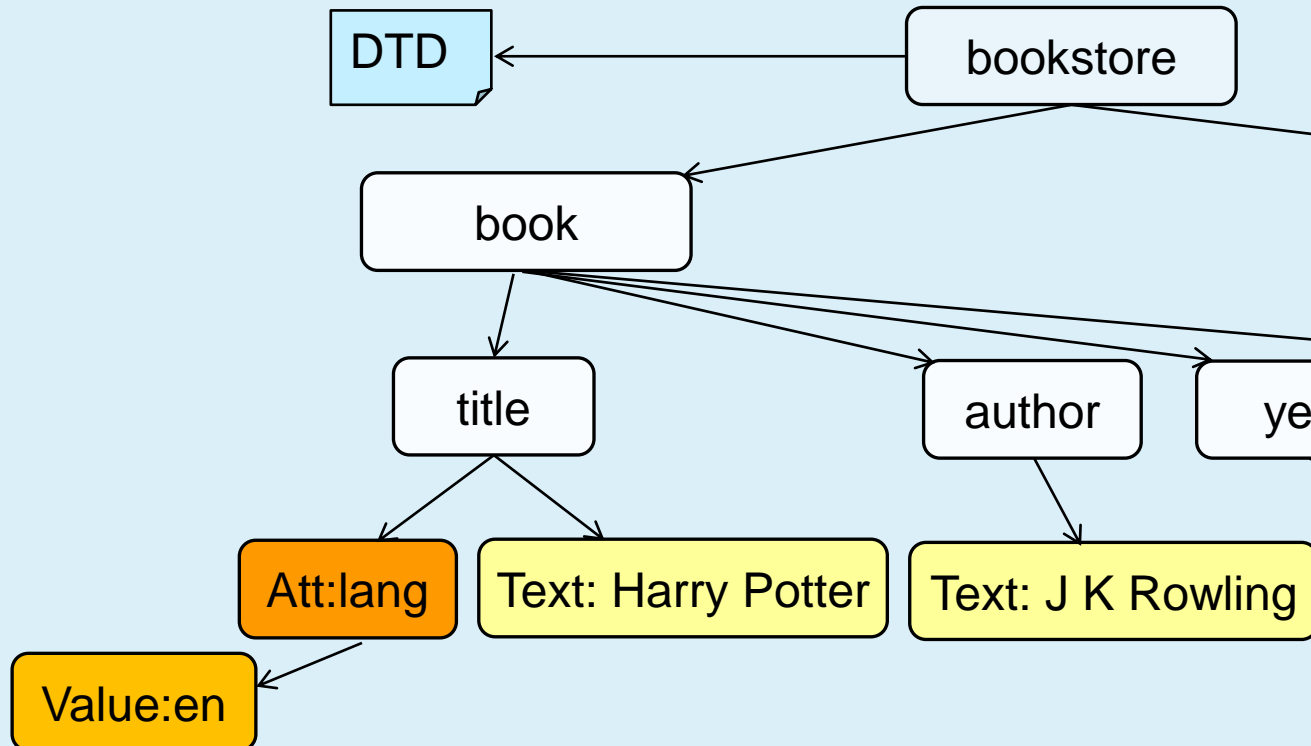
- ✓ Dicho fichero XML genera el siguiente árbol:



2. XPATH: Conceptos esenciales

2.1. Antes de nada

- ✓ Dicho fichero XML genera el siguiente árbol:



CONCEPTOS:

- ✓ Nodo raíz
- ✓ Nodos elemento
- ✓ Nodos atributos
- ✓ Valores atómicos (text y value)
- ✓ Padre
- ✓ Hijo
- ✓ Hermanos
- ✓ ...

¡¡XPath trabaja con nodos!!

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath

- ✓ Selección de nodos:

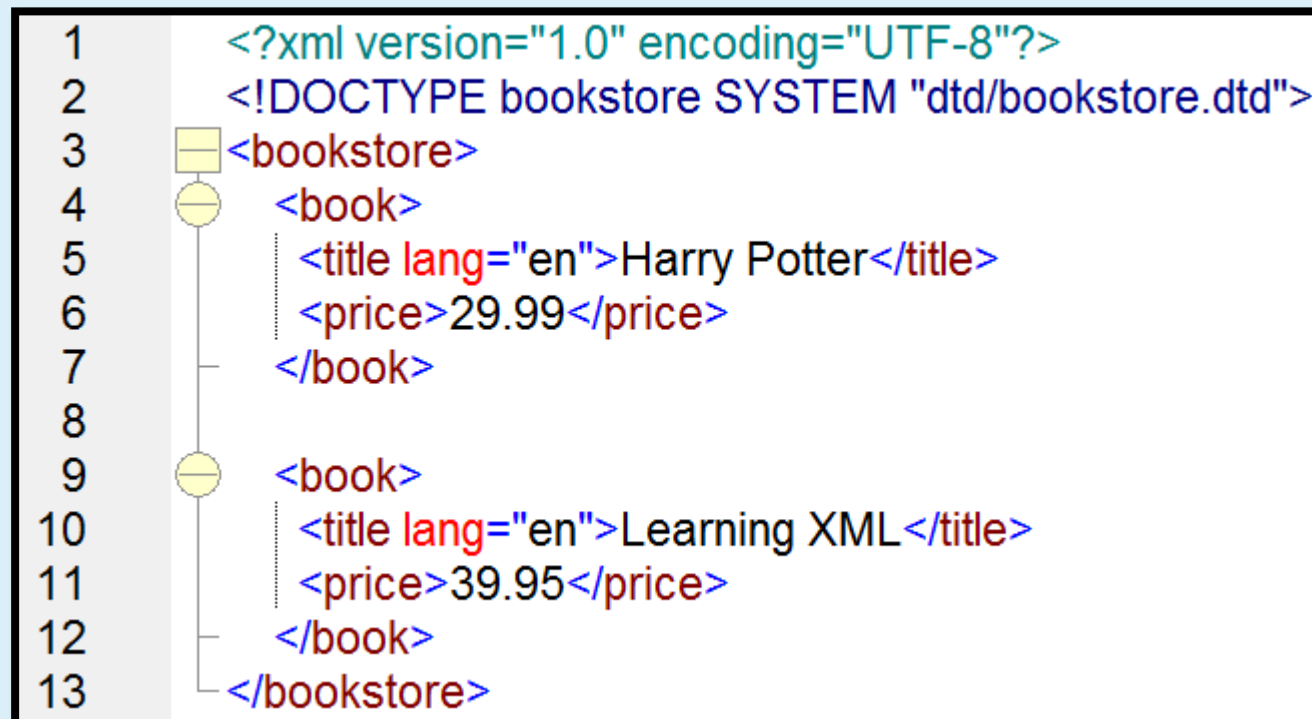
Expression	Description
<i>nodename</i>	Selects all nodes with the name " <i>nodename</i> "
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Nota: No confundir la raíz / con el nodo raíz.
El nodo raíz cuelga de / y por tanto tiene un padre, la raíz /.

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath

- ✓ Dado el siguiente fichero XML:



2. XPATH: Conceptos esenciales

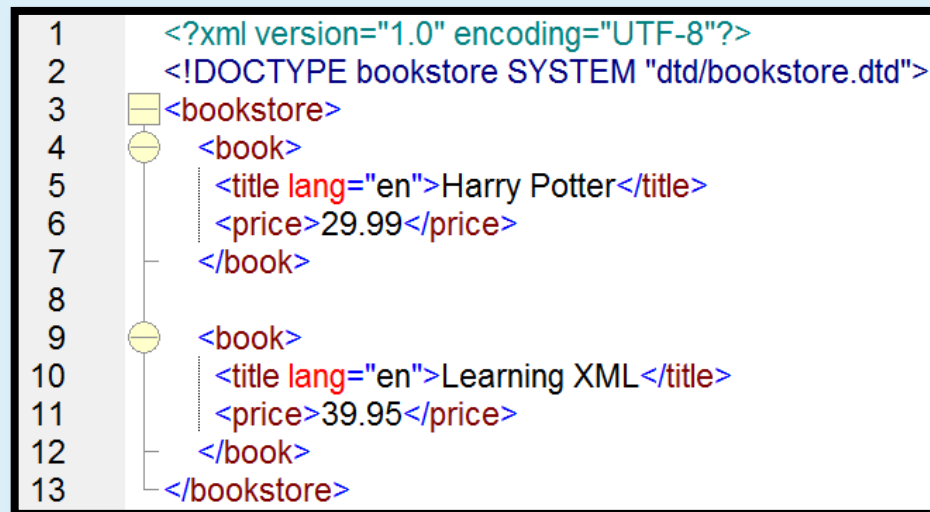
2.2. Sintaxis XPath

1	<?xml version="1.0" encoding="UTF-8"?>
2	<!DOCTYPE bookstore SYSTEM "dtd/bookstore.dtd">
3	<bookstore>
4	<book>
5	<title lang="en">Harry Potter</title>
6	<price>29.99</price>
7	</book>
8	
9	<book>
10	<title lang="en">Learning XML</title>
11	<price>39.95</price>
12	</book>
13	</bookstore>

Path Expression	Result
bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore Note: If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath



Las rutas pueden ser:

- Absolutas:
- Relativas:

book	Selects all nodes with the name "book"
/bookstore	Selects the root element bookstore Note: If the path starts with a slash (/) it always represents an absolute path to an element!

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath

- ✓ **Predicados:** Se utilizan para encontrar un nodo específico o un nodo que contiene un valor específico. Siempre están incrustados entre corchetes.

Path Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element. Note: In IE 5,6,7,8,9 first node is[0], but according to W3C, it is [1]. To solve this problem in IE, set the SelectionLanguage to XPath: <i>In JavaScript:</i> <code>xml.setProperty("SelectionLanguage","XPath");</code>
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath

✓ Predicados

1	<code><?xml version="1.0" encoding="UTF-8"?></code>
2	<code><!DOCTYPE bookstore SYSTEM "dtd/bookstore.dtd"></code>
3	 <code><bookstore></code>
4	<code><book></code>
5	<code><title lang="en">Harry Potter</title></code>
6	<code><price>29.99</price></code>
7	<code></book></code>
8	
9	<code><book></code>
10	<code><title lang="en">Learning XML</title></code>
11	<code><price>39.95</price></code>
12	<code></book></code>
13	<code></bookstore></code>

Path Expression
<code>/bookstore/book[1]</code>
<code>/bookstore/book[last()]</code>
<code>/bookstore/book[last()-1]</code>
<code>/bookstore/book[position()<3]</code>

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath

✓ Predicados

<code>//title[@lang]</code>	Selects all the title elements that have an attribute named lang
<code>//title[@lang='en']</code>	Selects all the title elements that have a "lang" attribute with a value of "en"
<code>/bookstore/book[price>35.00]</code>	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
<code>/bookstore/book[price>35.00]/title</code>	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath

✓ Predicados

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE bookstore SYSTEM "dtd/bookstore.dtd">
3  <bookstore>
4  <book>
5    <title lang="en">Harry Potter</title>
6    <price>29.99</price>
7  </book>
8
9  <book>
10   <title lang="en">Learning XML</title>
11   <price>39.95</price>
12 </book>
13 </bookstore>
```

//title[@lang]

//title[@lang='en']

/bookstore/book[price>35.00]

/bookstore/book[price>35.00]/title

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath

- ✓ Selección de nodos desconocidos

Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath

✓ Selección de nodos desconocidos

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE bookstore SYSTEM "dtd/bookstore.dtd">
3  <bookstore>
4  <book>
5      <title lang="en">Harry Potter</title>
6      <price>29.99</price>
7  </book>
8
9  <book>
10     <title lang="en">Learning XML</title>
11     <price>39.95</price>
12 </book>
13 </bookstore>
```

Path Expression	Result
/bookstore/*	Selects all the child element nodes of the bookstore element
//*	Selects all elements in the document
//title[@*]	Selects all title elements which have at least one attribute of any kind

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath

- ✓ Selección de varios elementos

Path Expression	Result
<code>//book/title //book/price</code>	Selects all the title AND price elements of all book elements
<code>//title //price</code>	Selects all the title AND price elements in the document
<code>/bookstore/book/title //price</code>	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath

✓ Selección de varios elementos

1	<?xml version="1.0" encoding="UTF-8"?>
2	<!DOCTYPE bookstore SYSTEM "dtd/bookstore.dtd">
3	<bookstore>
4	<book>
5	<title lang="en">Harry Potter</title>
6	<price>29.99</price>
7	</book>
8	
9	<book>
10	<title lang="en">Learning XML</title>
11	<price>39.95</price>
12	</book>
13	</bookstore>

Path Expression	Result
//book/title //book/price	Selects all the title AND price elements of all book elements
//title //price	Selects all the title AND price elements in the document
/bookstore/book/title //price	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document

2. XPATH: Conceptos esenciales

2.2. Sintaxis XPath

✓ Otros

- Existen otros selectores de nodos más avanzados (XPath Axes) que permiten navegar por el árbol y realizar selecciones más avanzadas. Los puedes encontrar en:

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

2. XPATH: Conceptos esenciales

2.3. Operadores XPath

- ✓ Algunos de los operadores que usaremos en clase:

=	Equal	price=9.80
!=	Not equal	price!=9.80
<	Less than	price<9.80
<=	Less than or equal to	price<=9.80
>	Greater than	price>9.80
>=	Greater than or equal to	price>=9.80

Más operadores en: https://www.w3schools.com/xml/xpath_operators.asp


2. XPATH: Conceptos esenciales

2.4. Funciones XPath

- ✓ XPath también define una serie de funciones muy útiles, aunque estas funciones las usaremos principalmente cuando estudiemos XQuery

Functions Reference

- [Accessor](#)
- [Error and Trace](#)
- [Numeric](#)
- [String](#)
- [AnyURI](#)
- [Boolean](#)
- [Duration/Date/Time](#)
- [QName](#)
- [Node](#)
- [Sequence](#)
- [Context](#)

 The default prefix for the function namespace is fn:
The URI of the function namespace is: <http://www.w3.org/2005/xpath-functions>

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

2. XPATH: Conceptos esenciales

2.5. Más ejemplos

- ✓ Queremos seleccionar todos los nodos `<h1>` que son hijos de un nodo `<body>` que, a su vez, es hijo de un nodo `<html>` que es el nodo raíz del árbol XML. ¿Cómo lo hacemos?
- ✓ Queremos seleccionar todos los nodos `<h1>` que aparezcan en cualquier posición del árbol XML. ¿Cómo lo hacemos?
- ✓ Queremos seleccionar todos los nodos `<h1>` que sean hijos directos de cualquier nodo `<div>`. Este último puede aparecer en cualquier posición del árbol XML. ¿Cómo lo hacemos?

2. XPATH: Conceptos esenciales

2.5. Más ejemplos

- ✓ Queremos seleccionar todos los nodos <h1> que son hijos de un nodo <body> que, a su vez, es hijo de un nodo <html> que es el nodo raíz del árbol XML. ¿Cómo lo hacemos?

```
/html/body/h1
```

- ✓ Queremos seleccionar todos los nodos <h1> que aparezcan en cualquier posición del árbol XML. ¿Cómo lo hacemos?

```
//h1
```

- ✓ Queremos seleccionar todos los nodos <h1> que sean hijos directos de cualquier nodo <div>. Este último puede aparecer en cualquier posición del árbol XML. ¿Cómo lo hacemos?

```
//div/h1
```

2. XPATH: Conceptos esenciales

2.5. Más ejemplos

- ✓ Queremos todos los nodos <libro> que aparezcan en el documento en cualquier posición y que tengan un nodo hijo <autor> con el valor "Hunter". ¿Cómo lo hacemos?
- ✓ Queremos todos los nodos <libro> que aparezcan en cualquier posición y que tengan un atributo "año" con un valor superior a "1999". ¿Cómo lo hacemos?
- ✓ Queremos todos los nodos <libro> que aparezcan en cualquier posición y que tengan un atributo "paginas", independientemente del valor de ese atributo. ¿Cómo lo hacemos?

2. XPATH: Conceptos esenciales

2.5. Más ejemplos

- ✓ Queremos todos los nodos <libro> que aparezcan en el documento en cualquier posición y que tengan un nodo hijo <autor> con el valor "Hunter". ¿Cómo lo hacemos?

```
//libro[autor = "Hunter"]
```

- ✓ Queremos todos los nodos <libro> que aparezcan en cualquier posición y que tengan un atributo "año" con un valor superior a "1999". ¿Cómo lo hacemos?

```
//libro[@año > 1999]
```

- ✓ Queremos todos los nodos <libro> que aparezcan en cualquier posición y que tengan un atributo "paginas", independientemente del valor de ese atributo. ¿Cómo lo hacemos?

```
//libro[@paginas]
```

2. XPATH: Conceptos esenciales

2.5. Más ejemplos

- ✓ Queremos solo el primero nodo <autor> que encuentre para cada nodo <libro> el cual debe ser un hijo del nodo <bib> que es el nodo raíz. ¿Cómo lo hacemos?
- ✓ Queremos solo el primer <autor> de todos. ¿Cómo lo hacemos?

2. XPATH: Conceptos esenciales

2.5. Más ejemplos

- ✓ Queremos solo el primero nodo <autor> que encuentre para cada nodo <libro> el cual debe ser un hijo del nodo <bib> que es el nodo raíz. ¿Cómo lo hacemos?

```
/bib/libro/autor[1]
```

- ✓ Queremos solo el primer <autor> de todos. ¿Cómo lo hacemos?

```
//autor[1]
```


2. XPATH: Conceptos esenciales

2.5. Más ejemplos

- ✓ Queremos todos los nodos <titulo> de los nodos <libro>, pero solo de los libros que tengan un atributo "año" con un valor superior a "1999". ¿Cómo lo hacemos?
- ✓ Queremos todos los nodos <titulo> que sean descendientes de <libreria>, pero que no tiene por qué ser hijos directos de <libreria>. Librería es el nodo raíz. ¿Cómo lo hacemos?

2. XPATH: Conceptos esenciales

2.5. Más ejemplos

- ✓ Queremos todos los nodos <titulo> de los nodos <libro>, pero solo de los libros que tengan un atributo "año" con un valor superior a "1999". ¿Cómo lo hacemos?

```
//libro[@año>1999]/titulo
```

- ✓ Queremos todos los nodos <titulo> que sean descendientes de <libreria>, pero que no tiene por qué ser hijos directos de <libreria>. Librería es el nodo raíz. ¿Cómo lo hacemos?

```
/libreria//titulo
```

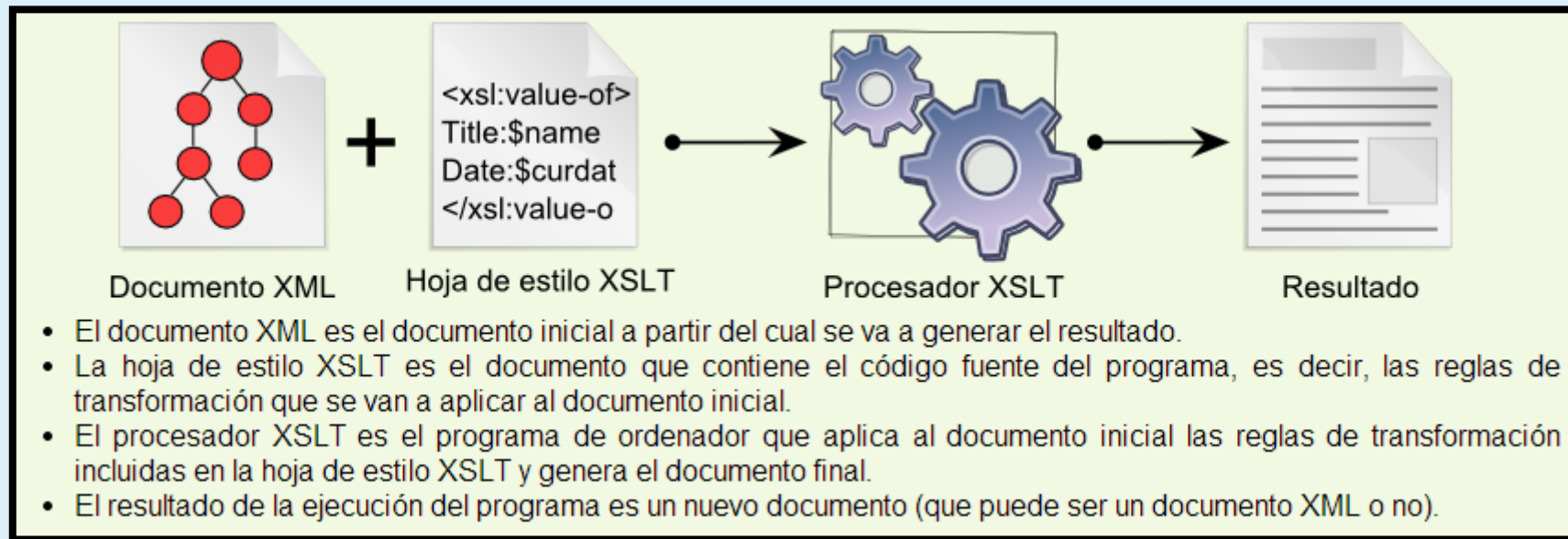
ÍNDICE

1. Introducción
2. XPath: Conceptos esenciales
 1. Antes de nada
 2. Sintaxis XPath
 3. Operadores XPath
 4. Funciones XPath
 5. Más ejemplos
3. **XSLT**
 1. Referenciar un fichero XSLT desde un fichero XML
 2. XSLT: <xsl:stylesheet>
 3. XSLT <xsl:template>
 4. XSLT <xsl:value-of>
 5. XSLT <xsl:for-each>
 6. XSLT <xsl:sort>
 7. XSLT <xsl:if>
 8. XSLT <xsl:choose>
 9. XSLT <xsl:apply-templates>
 10. XSLT: <xsl:element> y <xsl:attribute>
 11. XSLT: <xsl:output>
 12. XSLT: <xsl:comment>
 13. XSLT: Referencia
4. Bibliografía

3. 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.

3. XSLT



3. 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.**

3. XSLT

3.1. Referenciar un fichero XSLT desde un fichero XML

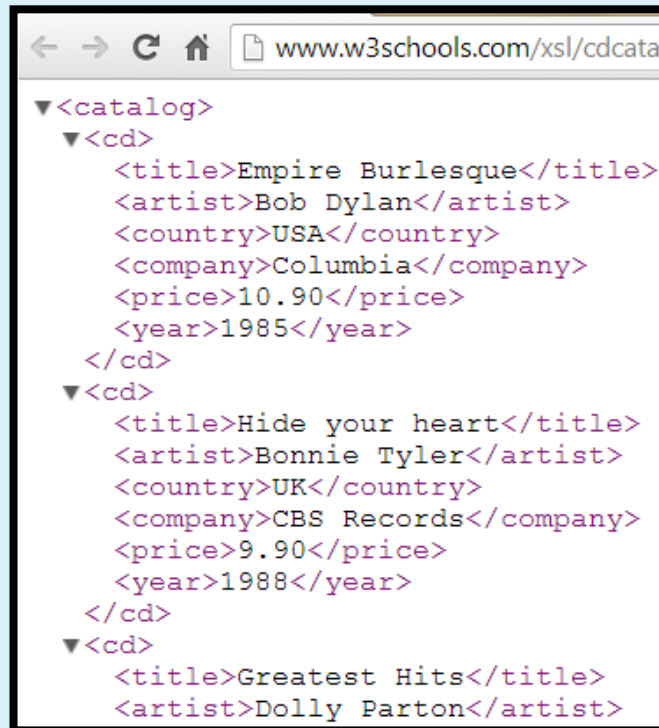
- ✓ Dado el siguiente fichero XML que tiene asociado un DTD:

```
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>
```

3. XSLT

3.1. Referenciar un fichero XSLT desde un fichero XML

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



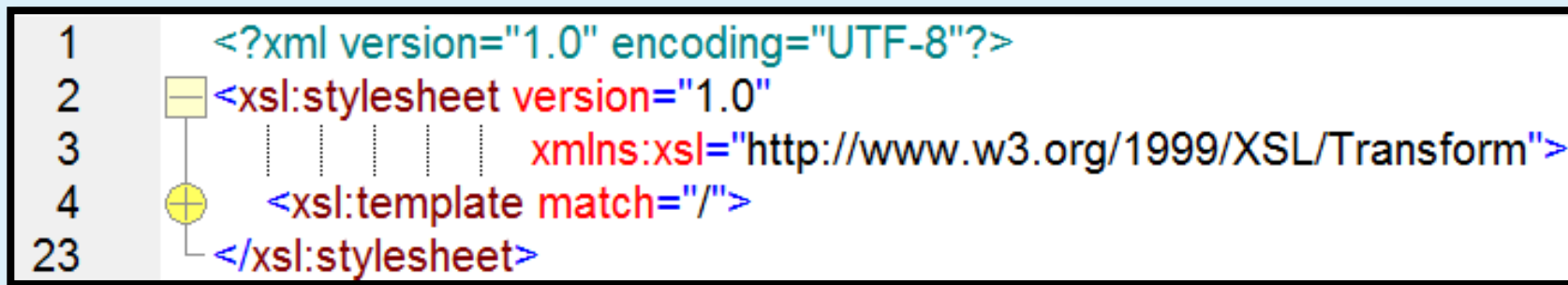
A screenshot of a web browser window displaying an XML document. The address bar shows the URL `www.w3schools.com/xsl/cdcata`. The XML content is displayed with syntax highlighting and collapsible tree view icons. The root element is `<catalog>`, which contains three `<cd>` elements. Each `<cd>` element has attributes for `title`, `artist`, `country`, `company`, `price`, and `year`.

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


3. XSLT

3.1. 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:



```
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>
```

The image shows a code editor window with a black border. On the left side, there is a vertical line of numbers: 1, 2, 3, 4, and 23. To the right of these numbers is the XML code. Line 1 contains the XML declaration. Line 2 starts the <xsl:stylesheet> tag with version="1.0". Line 3 continues the tag with xmlns:xsl="http://www.w3.org/1999/XSL/Transform". Line 4 starts the <xsl:template match="/"> tag. Line 23 ends the <xsl:stylesheet> tag. There are small yellow icons in the left margin: a rectangle on line 2 and a circle with a plus sign on line 4.

3. XSLT

3.1. 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>
```

3. XSLT

3.1. Referenciar un fichero XSLT desde un fichero XML

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

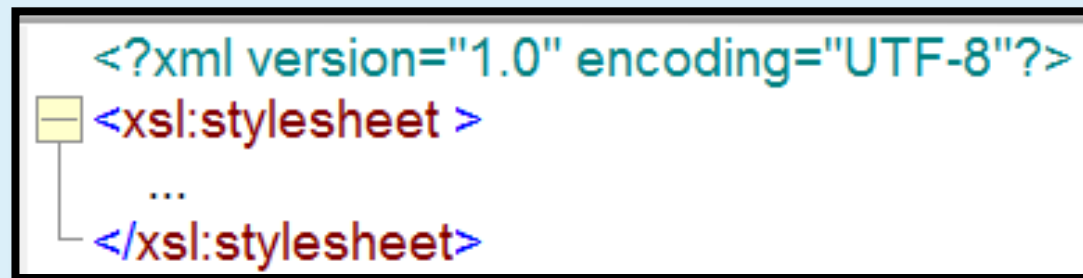
A screenshot of a web browser window. The address bar shows the URL 'www.w3schools.com/xsl/cdcatalog_with_xsl.xml'. The page title is 'My CD Collection'. Below the title is a table with two columns: 'Title' and 'Artist'. The table contains ten rows of data.

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
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge

3. XSLT

3.2. XSLT: <xsl:stylesheet>

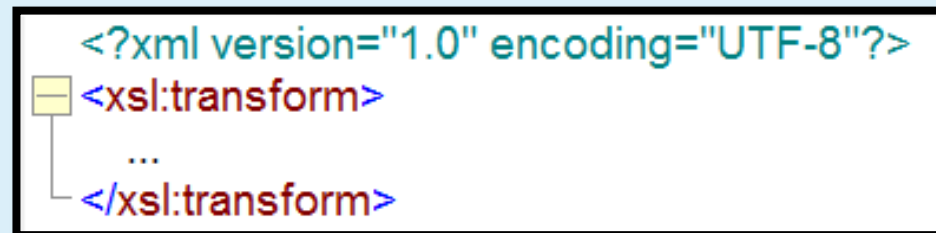
- ✓ Una hoja de transformación XSLT es un fichero XML cuyo nodo raíz es: <xsl:stylesheet>



The diagram shows an XML document 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 <xsl:stylesheet> tag. To the left of this tag is a small yellow box with a minus sign, indicating it is the root of the stylesheet. 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):



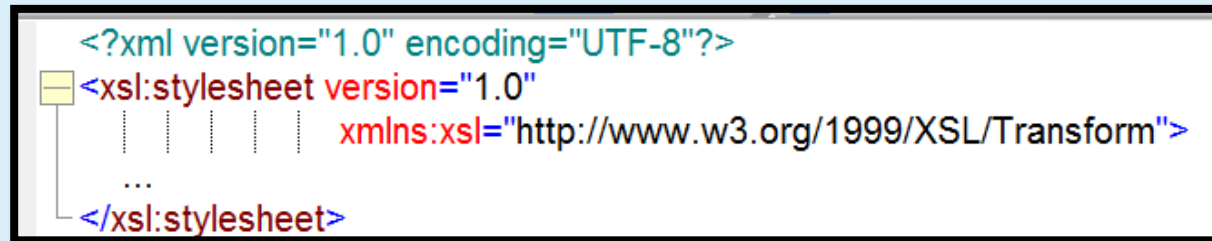
The diagram shows an alternative XML document structure for an XSLT transform. It starts with a root node <?xml version="1.0" encoding="UTF-8"?>. Below this is a yellow box containing the <xsl:transform> tag. To the left of this tag is a small yellow box with a minus sign, indicating it is the root of the transform. 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>
```

3. XSLT

3.2. XSLT: <xsl:stylesheet>

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



The diagram shows an XML root element `<?xml version="1.0" encoding="UTF-8"?>` followed by an XSLT stylesheet element `<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">`. The `version="1.0"` attribute is highlighted in red, and the `xmlns:xsl="http://www.w3.org/1999/XSL/Transform"` attribute is highlighted in blue. The element is closed with `</xsl:stylesheet>`. A small yellow box highlights the `<xsl:stylesheet` tag, and a vertical line connects it to the `version="1.0"` attribute. Dotted lines indicate other attributes that may be present.

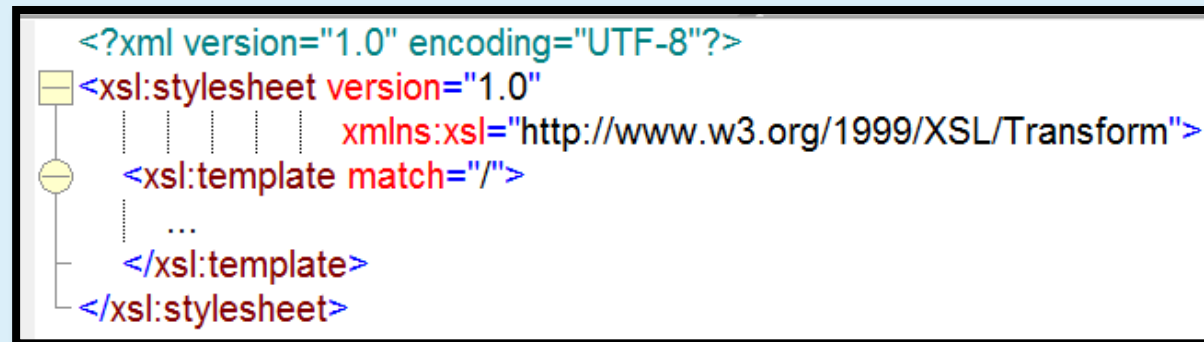
- version --> ¿Qué indica este atributo?
- xmlns:xsl --> ¿Qué indica este atributo?

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

3. XSLT

3.3. XSLT: <xsl:template>

- ✓ Justo después del nodo raíz debe aparecer el elemento <xsl:template>
- ✓ Este elemento nos permite definir un elemento plantilla dentro del XSL. Dicha plantilla nos permitirá generar la salida formateada.



- ✓ Es necesario indicar sobre qué parte del documento XML se quiere actuar utilizando el atributo "**match**".

En esta captura la plantilla ¿a quién afecta?

3. XSLT

3.3. XSLT: <xsl:template>

- ✓ En el siguiente ejemplo la hoja XSLT define una plantilla para generar un documento XHTML

```
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>
```

3. XSLT

3.4. XSLT: <xsl:value-of>

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

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


3. XSLT

3.4. XSLT: <xsl:value-of>

- ✓ Vemos un ejemplo:

```
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>
```

```
<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"/>
```

3. XSLT

3.4. XSLT: <xsl:value-of>

- ✓ 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>
```

3. XSLT

3.4. XSLT: <xsl:value-of>

- ✓ Integramos esto en la plantilla anterior:

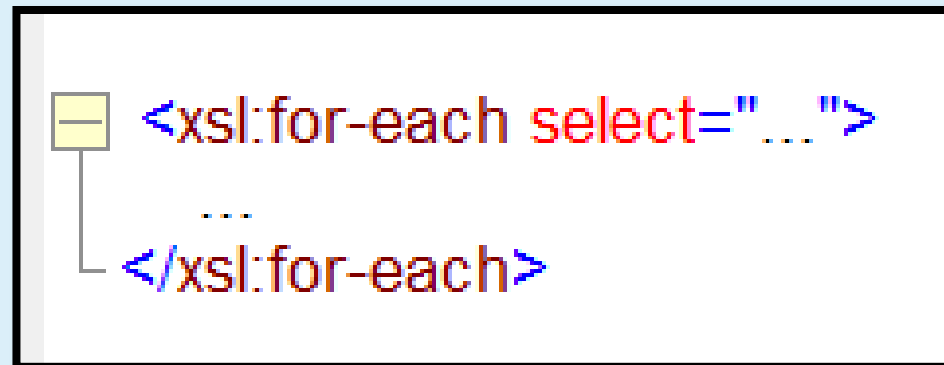
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3  <xsl:template match="/">
4    <html>
5      <head>
6        <title>CD</title>
7        <style>
8          table, th, td { border: 1px solid blue;}
9        </style>
10     </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
20   <td>
21     <xsl:value-of select="/catalog/cd[1]/title"/>
22   </td>
23   <td>
24     <xsl:value-of select="/catalog/cd[1]/artist"/>
25   </td>
26 </tr>
27 <tr>
28   <td>
29     <xsl:value-of select="/catalog/cd[last()]/title"/>
30   </td>
31   <td>
32     <xsl:value-of select="/catalog/cd[last()]/artist"/>
33   </td>
34 </tr>
35 </table>
36 </body>
37 </html>
38 </xsl:template>
</xsl:stylesheet>
```

3. XSLT

3.5. 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 elemento queremos recorrer.



```
<xsl:for-each select="...">
  ...
</xsl:for-each>
```

3. XSLT

3.5. XSLT: <xsl:for-each>

✓ ¿Qué conseguiríamos con esto?

a)

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

<xsl:value-of select="."/>
indica que se obtenga el
valor del nodo actual.

b)

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

c)

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

3. XSLT

3.5. XSLT: <xsl:for-each>

- ✓ A medida que vamos recorriendo, podemos aplicar distintos tipos de filtros:

Filtering the Output

We can also filter the output from the XML file by adding a criterion to the select attribute in the <xsl:for-each> element.

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

Legal filter operators are:

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

3. XSLT

3.5. XSLT: <xsl:for-each>

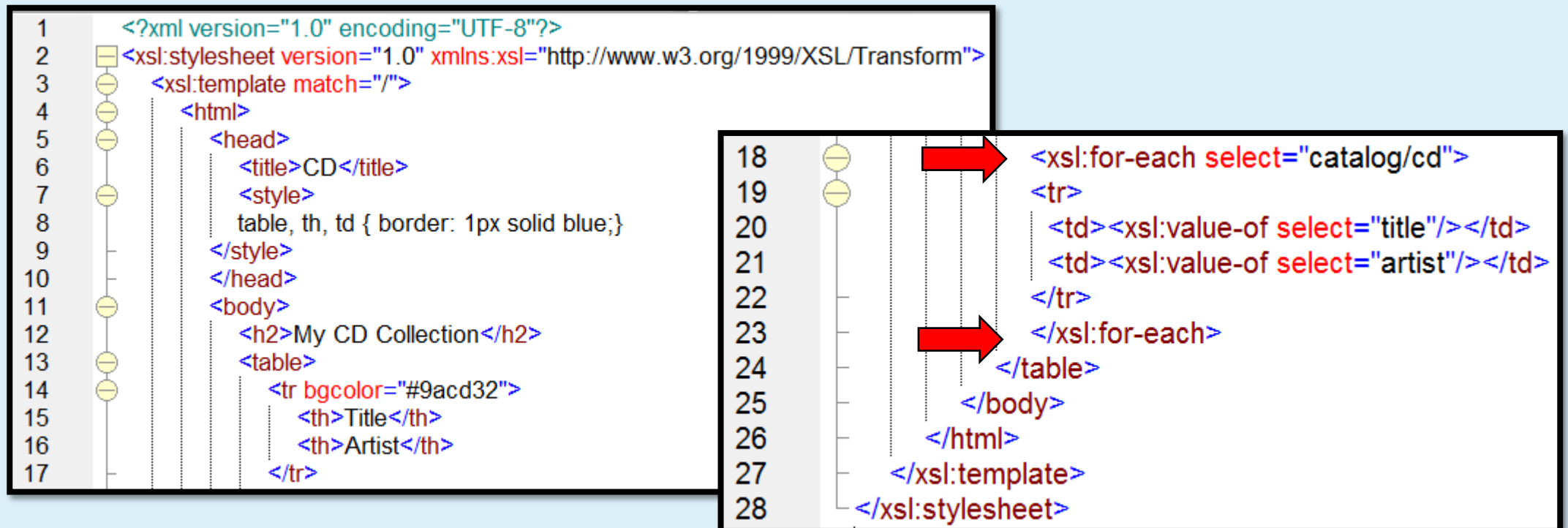
- ✓ Integramos esto en la plantilla anterior:

```
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>
```

3. XSLT

3.5. XSLT: <xsl:for-each>

- ✓ Integramos esto en la plantilla anterior:



3. XSLT

3.5. XSLT: <xsl:for-each>

✓ ¿Qué ocurre en este ejemplo?

```
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>
```



3. XSLT

3.5. XSLT: <xsl:for-each>

✓ ¿Qué ocurre en este ejemplo?

My CD Collection				
CD				
Empire Burlesque	Bob Dylan	USA	Columbia	10.901985
Hide your heart	Bonnie Tyler	UK	CBS Records	9.901988
Greatest Hits	Dolly Parton	USA	ARCA	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
Black angel	Savage Rose	EU	Mega	10.901995
1999 Grammy Nominees	Many	USA	Grammy	10.201999
For the good times	Kenny Rogers	UK	Mucik Master	8.701995
Big Willie style	Will Smith	USA	Columbia	9.901997
Tupelo Honey	Van Morrison	UK	Polydor	8.201971
Soulsville	Jorn Hoel	Norway	WEA	7.901996
The very best of	Cat Stevens	UK	Island	8.901990

3. XSLT

3.6. XSLT: <xsl:sort>

- ✓ Permite ordenar la salida mostrada al usuario.
- ✓ En el atributo "select" usando XPath se indica a través de qué elemento queremos ordenar

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

3. XSLT

3.6. 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"/>
```

3. XSLT

3.6. XSLT: <xsl:sort>

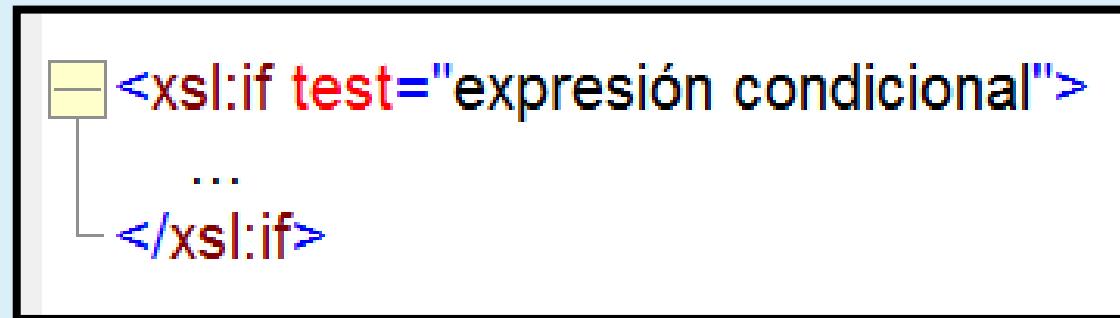
✓ Integramos esto en la plantilla anterior:

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

3. XSLT

3.7. XSLT: <xsl:if>

- ✓ Podemos definir sentencias condicionales sencillas:



The diagram shows an `<xsl:if test="expresión condicional">` tag in red. A vertical line descends from the opening tag, with three dots in the middle, and then branches to the closing tag `</xsl:if>` in blue. A small yellow box with a horizontal line is positioned to the left of the opening tag.

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

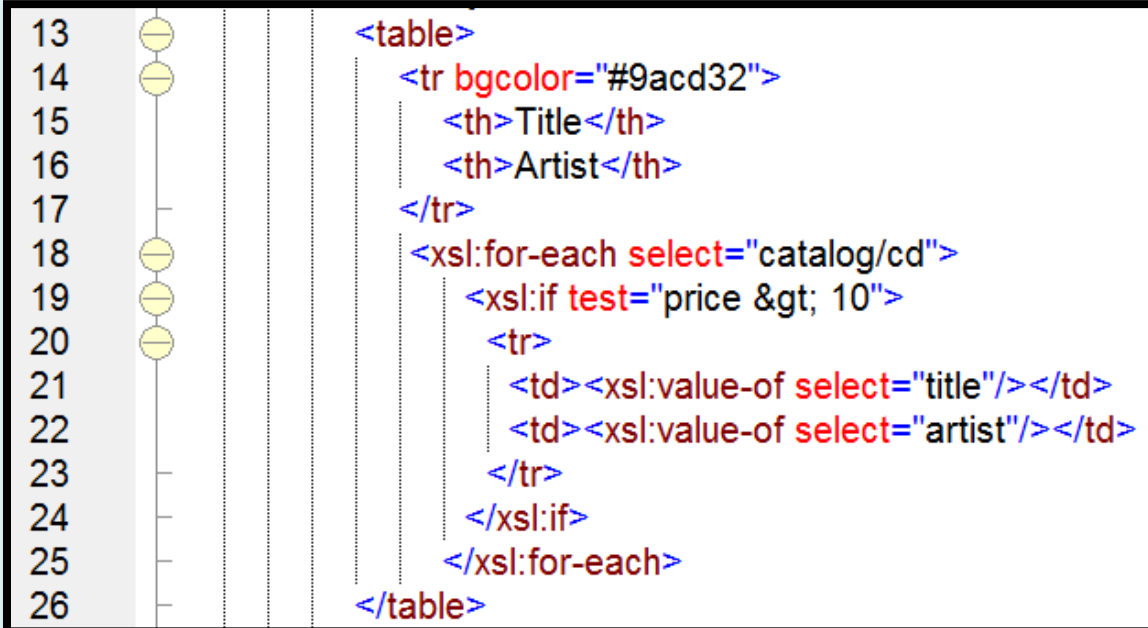
Legal operators are:

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

3. XSLT

3.7. XSLT: <xsl:if>

- ✓ Siguiendo con el ejemplo anterior:



The diagram consists of a vertical timeline on the left with yellow circles at lines 13, 14, 18, 19, and 20. To the right of the timeline is an XSLT code block. The code defines a table with a light green background, two header rows (Title and Artist), and a body that iterates over catalog/cd elements. For each element, it checks if the price is greater than 10. If true, it outputs the title and artist. The code is as follows:

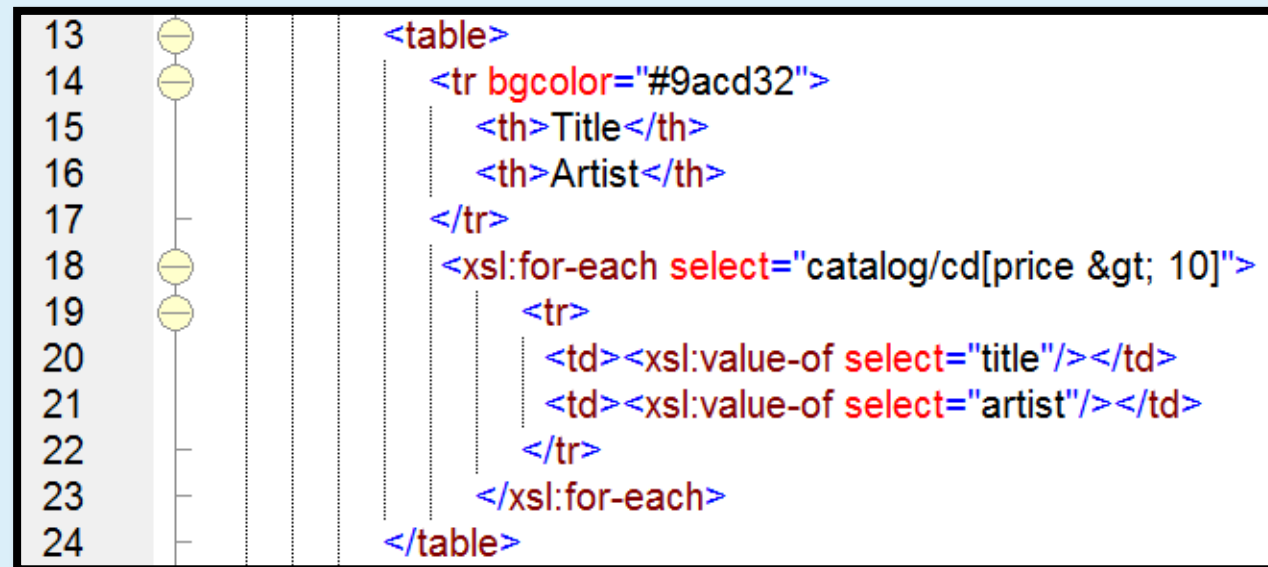
```
13 <table>
14 <tr bgcolor="#9acd32">
15 <th>Title</th>
16 <th>Artist</th>
17 </tr>
18 <xsl:for-each select="catalog/cd">
19 <xsl:if test="price > 10">
20 <tr>
21 <td><xsl:value-of select="title"/></td>
22 <td><xsl:value-of select="artist"/></td>
23 </tr>
24 </xsl:if>
25 </xsl:for-each>
26 </table>
```

¿Qué obtenemos?

3. XSLT

3.7. XSLT: <xsl:if>

- ✓ Siguiendo con el ejemplo anterior:



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

¿Y ahora qué obtenemos?

3. XSLT

3.7. XSLT: <xsl:if>

- ✓ Siguiendo con el ejemplo anterior:

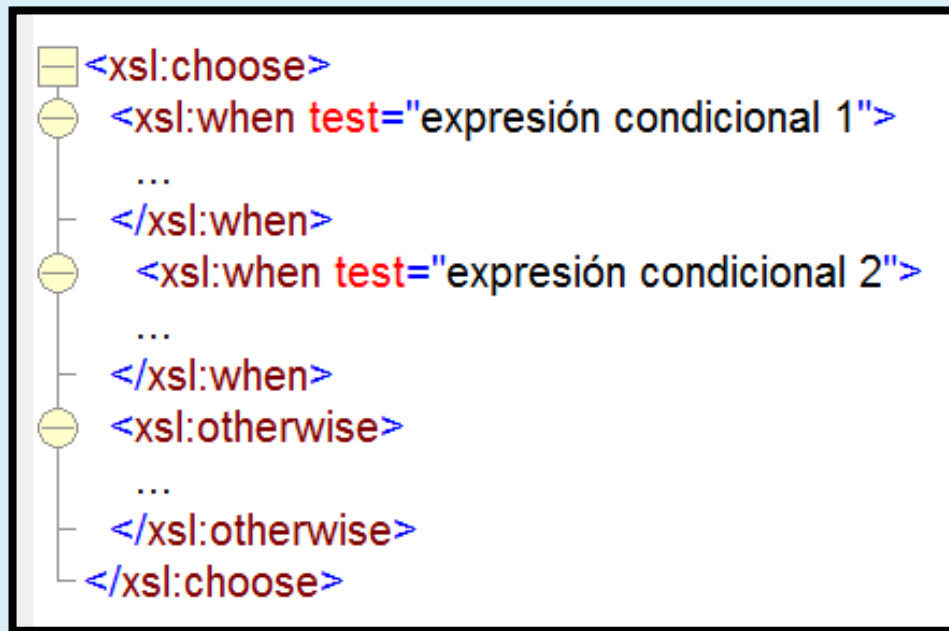
```
13 <table>
14 <tr bgcolor="#9acd32">
15 <th>Title</th>
16 <th>Artist</th>
17 <th>Price</th>
18 </tr>
19 <xsl:for-each select="catalog/cd[artist='Bee Gees']">
20 <xsl:if test="price > 10">
21 <tr>
22 <td><xsl:value-of select="title"/></td>
23 <td><xsl:value-of select="artist"/></td>
24 <td><xsl:value-of select="price"/></td>
25 </tr>
26 </xsl:if>
27 </xsl:for-each>
28 </table>
```

¿Y en este último ejemplo qué obtenemos?

3. XSLT

3.8. XSLT: <xsl:choose>

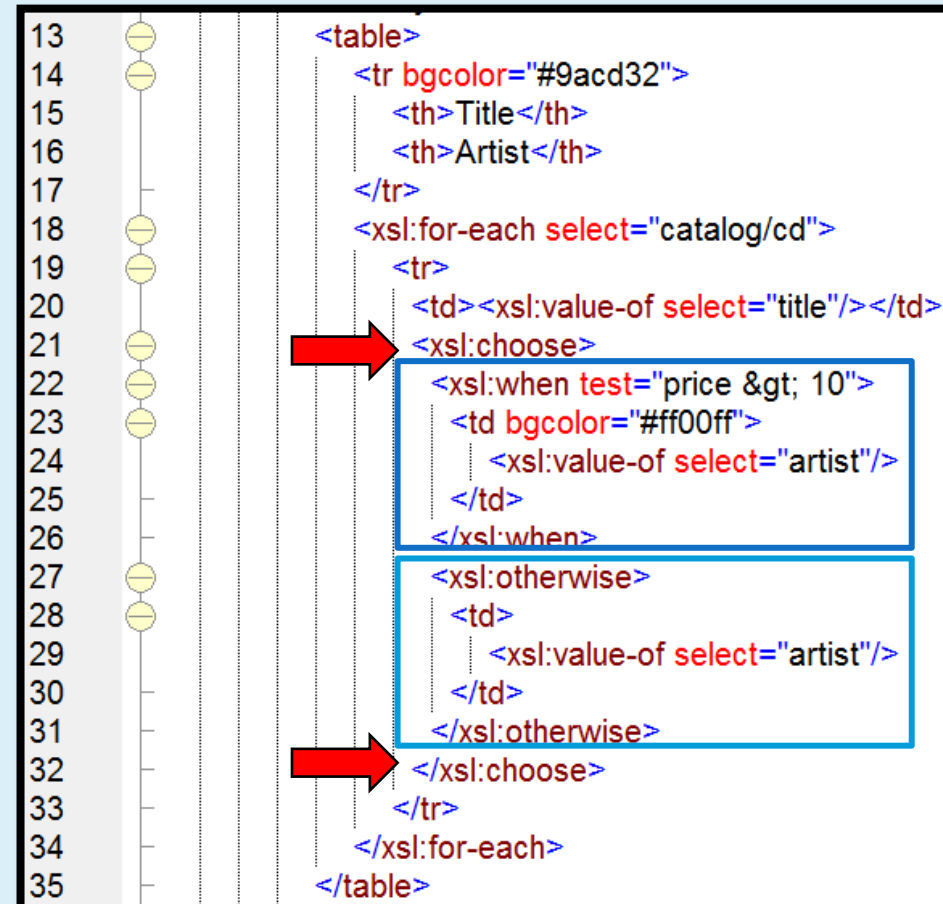
- ✓ Podemos definir sentencias condicionales múltiples.



3. XSLT

3.8. XSLT: <xsl:choose>

- ✓ Siguiendo con el ejemplo anterior



The diagram illustrates an XSLT transformation for generating an HTML table. On the left, a vertical timeline from line 13 to 35 shows the execution flow with yellow circles and horizontal lines. The XSLT code on the right uses `<xsl:choose>` to conditionally format table rows based on the price. A red arrow points from line 21 to the start of the `<xsl:choose>` block, and another red arrow points from line 32 to the end of the `</xsl:choose>` block.

```
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       <xsl:choose>
22         <xsl:when test="price > 10">
23           <td bgcolor="#ff00ff">
24             <xsl:value-of select="artist"/>
25           </td>
26         </xsl:when>
27         <xsl:otherwise>
28           <td>
29             <xsl:value-of select="artist"/>
30           </td>
31         </xsl:otherwise>
32       </xsl:choose>
33     </tr>
34   </xsl:for-each>
35 </table>
```

3. XSLT

3.9. XSLT: <xsl:apply-templates>

- ✓ En los ejemplos anteriores solo hemos utilizado una plantilla para todo el documento XML. Para ello hemos usado el elemento <xsl:template> con el atributo "match" referenciando a la raíz del documento XML.
- ✓ Sin embargo podemos aplicar diferentes "plantillas" a los nodos que queramos usando el elemento: <xsl:apply-templates>

The <xsl:apply-templates> Element

The <xsl:apply-templates> element applies a template to the current element or to the current element's child nodes.

If we add a select attribute to the <xsl:apply-templates> element it will process only the child element that matches the value of the attribute. We can use the select attribute to specify the order in which the child nodes are processed.

3. XSLT

3.9. XSLT: <xsl:apply-templates>

```
2 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3   <xsl:template match="/">
4     <html>
5       <body>
6         <h2>My CD Collection</h2>
7         <xsl:apply-templates/>
8       </body>
9     </html>
10  </xsl:template>
11
12  <xsl:template match="cd">
13    <p>
14      <xsl:apply-templates select="title"/>
15      <xsl:apply-templates select="artist"/>
16    </p>
17  </xsl:template>
18
19  <xsl:template match="title">
20    Title:
21    <span style="color:#ff0000">
22      <xsl:value-of select="."/>
23    </span>
24    <br />
25  </xsl:template>
26
27  <xsl:template match="artist">
28    Artist:
29    <span style="color:#00ff00">
30      <xsl:value-of select="."/>
31    </span>
32    <br />
33  </xsl:template>
34 </xsl:stylesheet>
```

<xsl:apply-templates/>: Indica que se apliquen el resto de las plantillas definidas en cuanto se cumpla el patrón match de cada uno.

<xsl:apply-templates select="XXXX"/>: indica que se aplique en ese momento, el contenido de la plantilla "XXXX" definido en el documento XSL.

select=".": indica el elemento actual.

3. XSLT

3.9. XSLT: <xsl:apply-templates>

My CD Collection

Title: Empire Burlesque
Artist: Bob Dylan

Title: Hide your heart
Artist: Bonnie Tyler

Title: Greatest Hits
Artist: Dolly Parton

Title: Still got the blues
Artist: Gary Moore

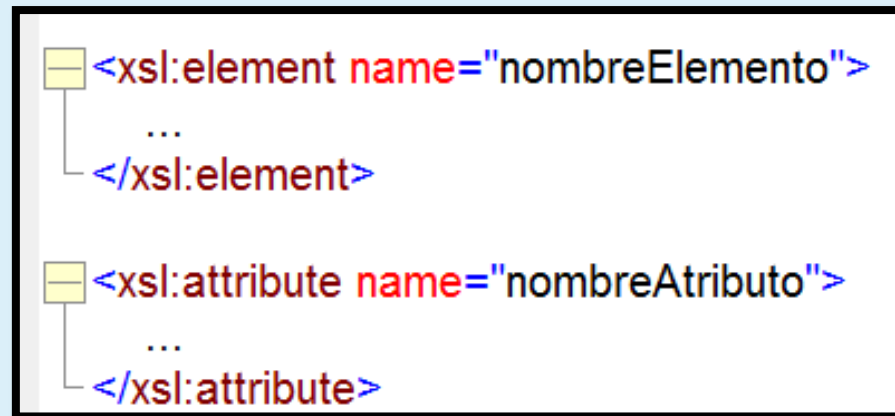
Title: Eros
Artist: Eros Ramazzotti

Title: One night only
Artist: Bee Gees

3. XSLT

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

- ✓ Permiten crear nuevos elementos y nuevos atributos XML



The diagram illustrates the structure of XSLT element and attribute declarations. It consists of two separate blocks, each enclosed in a box with a yellow header bar. The first block shows the declaration of an element: `<xsl:element name="nombreElemento">` followed by an ellipsis `...` and then the closing tag `</xsl:element>`. The second block shows the declaration of an attribute: `<xsl:attribute name="nombreAtributo">` followed by an ellipsis `...` and then the closing tag `</xsl:attribute>`. In both cases, the opening tag is on the first line, the ellipsis is on the second line, and the closing tag is on the third line. A vertical line connects the opening tag to the closing tag, indicating the scope of the declaration.

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

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

3. XSLT

3.10. 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>
```


3. XSLT

3.10. 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>
```

3. XSLT

3.10. 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>
```

3. XSLT

3.11. 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.

3. XSLT

3.11. 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

3. XSLT

3.11. 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>
```

3. XSLT

3.12. 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>
```

3. XSLT

3.13. 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. Introducción
2. XPath: Conceptos esenciales
 1. Antes de nada
 2. Sintaxis XPath
 3. Operadores XPath
 4. Funciones XPath
 5. Más ejemplos
3. XSLT
 1. Referenciar un fichero XSLT desde un fichero XML
 2. XSLT: <xsl:stylesheet>
 3. XSLT <xsl:template>
 4. XSLT <xsl:value-of>
 5. XSLT <xsl:for-each>
 6. XSLT <xsl:sort>
 7. XSLT <xsl:if>
 8. XSLT <xsl:choose>
 9. XSLT <xsl:apply-templates>
 10. XSLT: <xsl:element> y <xsl:attribute>
 11. XSLT: <xsl:output>
 12. XSLT: <xsl:comment>
 13. XSLT: Referencia

4. Bibliografía

4. 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>