LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE LA INFORMACIÓN

U.T.4 . XML. CONVERSION Y ADAPTACION DE DOCUMENTOS XML.

TRANSFORMACION DE DOCUMENTOS XML

- LAS TECNOLOGÍAS PRINCIPALES EN LA TRANSFORMACIÓN DE DOCUMENTOS SON:
- **XSLT**: (E***X***TENSIBLE ***S***TYLESHEET ***L***ANGUAGE ***T***RANSFORMATION)PERMITE DEFINIR EL MODO DE TRANSFORMAR UN DOCUMENTO XML EN OTRO.
- **XSL-FO**: (E***X***TENSIBLE ***S***TYLESHEET ***L***ANGUAGE - ***F***ORMATTING ***O***BJECT) SE UTILIZA PARA TRANSFORMAR XML EN UN FORMATO LEGIBLE E IMPRIMIBLE POR UNA PERSONA, POR EJEMPLO EN UN DOCUMENTO PDF.
- **XPATH**: PERMITE EL ACCESO A LOS DIVERSOS COMPONENTES DE UN DOCUMENTO XML

XPATH 1.0

- EL COMPONENTE FUNDAMENTAL ES LA EXPRESION XPATH. UNA EXPRESION SE EVALUA Y GENERA UN OBJETO, QUE PUEDE SER DE LOS SIGUIENTES TIPOS:
- NODE-SET
- BOOLEAN
- NUMERO (UN NUMERO REAL)
- STRING
- UNA EXPRESIÓN XPATH ES UN PREDICADO O CONDICION QUE SE APLICA SOBRE UNA ESTRUCTURA DE ÁRBOL CORRESPONDIENTE A LA JERARQUÍA DE LOS DATOS DE UN DOCUMENTO XML, Y DEVUELVE TODO LO QUE ENCAJA CON ESE PREDICADO.
- NOS PERMITE ACCEDER A PARTES DE UN DOCUMENTO XML BASÁNDONOS EN LAS RELACIONES DE PARENTESCO ENTRE LOS NODOS DEL DOCUMENTO.
- SU NOTACIÓN ES SIMILAR A LAS DE LAS RUTAS DE LOS FICHEROS.

XPATH: TERMINOS BASICOS

- **Nodo raíz**, es el nodo que contiene al ejemplar del fichero XML. No debe confundirse con el elemento raíz del documento, ya que éste último está por debajo de él. Se identifica por “/”.
- **Nodos elemento**, son cada uno de los elementos del documento XML. Todos ellos tienen un elemento padre, el padre del elemento raíz, es decir del ejemplar, es el nodo raíz del documento. Pueden tener identificadores únicos, para ello es necesario que un atributo esté definido de ese modo en un DTD o un fichero XSD asociado, esto nos permite referenciar dicho elemento de forma mucho más directa.
- **Nodos texto**, son aquellos caracteres del documento que no están marcados con ninguna etiqueta. Este tipo de nodos no tienen hijos, son las hojas.
- **Nodos atributo**, no se consideran hijos del elemento al que están asociados sino etiquetas añadidas al nodo elemento. Aquellos atributos que tengan un valor asignado en el esquema asociado, se tratarán como si ese valor se le hubiese dado al escribir el documento XML. (No se incluyen los #IMPLIED)
- **Nodo actual**, es aquél al que nos referimos cuando se evalúa una expresión en Xpath.

XPATH: TERMINOS BASICOS

- **Nodo contexto**, Un nodo contexto es el nodo que el procesador XPath está evaluando actualmente. El nodo e contexto cambia cuando el procesador XPath evalúa una consulta. Si se le pasa un documento al procesador XPath, el nodo raíz es el nodo contexto inicial. Si se pasa un nodo al procesador XPath, el nodo que pasa es el nodo contexto inicial.
- **Tamaño del contexto**, es el número de nodos que se están evaluando en un momento dado en una expresión Xpath.

XPATH: EJEMPLOS

- LAS RUTAS EN XPATH SON SIMILARES AUNQUE NO IGUALES A LAS DE UN SISTEMA DE FICHEROS, P.E.:
- /libro/capitulo/parrafo
- HACE REFERENCIA A TODOS LOS ELEMENTOS PARRAFO QUE CUELGUEN DIRECTAMENTE DE CUALQUIER ELEMENTO CAPITULO QUE CUELGUE DE CUALQUIER ELEMENTO LIBRO QUE CUELGUEN DEL NODO RAÍZ, /.

XPATH: RESULTADOS EVALUACIÓN DE UNA EXPRESIÓN XPATH

- UN CONJUNTO DE NODOS QUE NO ESTÁ ORDENADO Y SIN DUPLICADOS: **NODE-SET**
- BOOLEAN, ES DECIR EL RESULTADO DE EVALUAR UNA EXPRESION LOGICA.
- UN NUMERO.
- UNA CADENA
- EL NODE-SET PUEDE SER DE VARIOS TIPOS DIFERENTES:
  - ELEMENTO.
  - ATRIBUTO.
  - TEXTO.
  - INSTRUCCIÓN DE PROCESAMIENTO.
  - RAÍZ.

XPATH:PASOS DE BUSQUEDA

- LAS EXPRESIONES XPATH SE PUEDEN DIVIDIR EN PASOS DE BÚSQUEDA. CADA PASO DE BÚSQUEDA SE PUEDE A SU VEZ DIVIDIR EN TRES PARTES:

➢EJE: SELECCIONA NODOS ELEMENTO O ATRIBUTO BASÁNDOSE EN SUS NOMBRES.

➢PREDICADO: SI EL EJE HA SELECCIONADO UNOS NODOS, EL PREDICADO PERMITE RESTRINGIR ESA SELECCIÓN A LOS QUE CUMPLAN DETERMINADAS CONDICIONES.

➢SELECCIÓN DE NODOS: DE LOS NODOS SELECCIONADOS POR EL EJE Y PREDICADO, SELECCIONA LOS ELEMENTOS, EL TEXTO QUE CONTIENEN O AMBOS.

XPATH:EJES

- EJEMPLOS: DADO EL XML SIGUIENTE:


```xml
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Pantaleón y las visitadoras</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Conversación en la catedral</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1969"/>
  </libro>
</biblioteca>
```

XPATH:EJES

- SE VAN A EVALUAR UNA SERIE DE EXPRESIONES XPATH. PUEDEN PROBARSE EN:

- https://www.freeformatter.com/xpath-tester.html

XPATH:EJES

- EJEMPLOS QUE COMIENZAN CON “/”:

“/" si está al principio de la expresión, indica el nodo raíz, si no, indica "hijo".

/biblioteca/libro/autor	<autor>Milan Kundera</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
/autor	No devuelve nada porque <autor> no es hijo del nodo raíz.
/biblioteca/autor	No devuelve nada porque <autor> no es hijo de <biblioteca>.
/biblioteca/libro/autor/@fechaNacimiento	fechaNacimiento="28/03/1936" fechaNacimiento="28/03/1936"
/biblioteca/libro/@fechaNacimiento	No devuelve nada porque <libro> no tiene el atributo fechaNacimiento.

Nota: En XPath 1.0 no se puede seleccionar únicamente el valor del atributo, sino que se obtienen respuestas del tipo nombreDelAtributo=ValorDelAtributo

XPATH:EJES

- EJEMPLOS QUE INCLUYEN “/..”: (indica elemento padre)

```
/biblioteca/libro/autor/@fechaNacimiento/..
```

```
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

```
//@fechaNacimiento/../..
```

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

En el primer caso se selecciona el padre de @fechaNacimiento y en el segundo el “abuelo”.

XPATH:EJES

- EJEMPLOS QUE COMIENZAN O INCLUYEN “//”:

//: indica "descendiente" (hijos, hijos de hijos, etc.).

`/biblioteca//autor`	`<autor>Milan Kundera</autor>`
	`<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>`
	`<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>`

`//autor`	`<autor>Milan Kundera</autor>`
	`<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>`
	`<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>`

`//autor//libro`	No devuelve nada porque <libro> no es descendiente de <autor>.

`//@año`	`año="1973"`
	`año="1973"`
	`año="1969"`

Nota: En XPath 1.0 no se puede seleccionar únicamente el valor del atributo, sino que se obtienen respuestas del tipo nombreDelAtributo=ValorDelAtributo

XPATH:EJES

- EJEMPLOS QUE INCLUYEN “|”: (permite elegir varios caminos)

```
//autor|//titulo
```

```
<titulo>La vida está en otra parte</titulo>
<autor>Milan Kundera</autor>
<titulo>Pantaleón y las visitadoras</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<titulo>Conversación en la catedral</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

```
//autor|//titulo|//@año
```

```
<titulo>La vida está en otra parte</titulo>
<autor>Milan Kundera</autor>
 año="1973"
<titulo>Pantaleón y las visitadoras</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
 año="1973"
<titulo>Conversación en la catedral</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
 año="1969"
```

XPATH:PREDICADOS: INDICES

- El predicado se escribe entre corchetes, a continuación del eje. Si el eje ha seleccionado unos nodos, el predicado permite restringir esa selección a los que cumplan determinadas condiciones.

- **[@atributo]**: selecciona los elementos que tienen el atributo.

```
//autor[@fechaNacimiento]
```

```
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

- **[número]**: si hay varios resultados selecciona uno de ellos por número de orden; **last()** selecciona el último de ellos

```
//libro[1]
```

```
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
```

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

```
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

```
//libro[last()-1]
```

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
```

XPATH:PREDICADOS: CONDICIONES

- **[condicion]**: selecciona los nodos que cumplen la condición. Los predicados permiten definir condiciones sobre los valores de los atributos. En las condiciones se pueden utilizar los operadores siguientes:
- operadores lógicos: and, or, not()
- operadores aritméticos: +, -, *, div, mod
- operadores de comparación: =, !=, <, >, <=, >=

XPATH:PREDICADOS: CONDICIONES

```
//fechaPublicacion[@año>1970]
```

```
<fechaPublicacion año="1973"/>
<fechaPublicacion año="1973"/>
```

```
//libro[autor="Mario Vargas Llosa"]
```

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

Para hacer referencia al propio valor del elemento seleccionado se utiliza el punto (.).

```
//@año[.>1970]
```

```
año="1973"
año="1973"
```

```
//autor[.="Mario Vargas Llosa"]
```

```
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

XPATH:PREDICADOS: CONDICIONES

```
//libro[autor="Mario Vargas Llosa" and fechaPublicacion/@año="1973"]
```

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
```

```
//libro[autor="Mario Vargas Llosa" or fechaPublicacion/@año="1973"]
```

```
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

XPATH:PREDICADOS: CONDICIONES

Se pueden escribir varios predicados seguidos, cada uno de los cuales restringe los resultados del anterior, como si estuvieran encadenados por la operación lógica and.
En el ejemplo siguiente se seleccionan los libros escritos por Mario Vargas Llosa y publicados en 1973:

```
//libro[autor="Mario Vargas Llosa"][fechaPublicacion/@año="1973"]
```

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
```

XPATH:SELECCIÓN DE NODOS

La selección de nodos se escribe a continuación del eje y el predicado. Si el eje y el predicado han seleccionado unos nodos, la selección de nodos indica con qué parte de esos nodos nos quedamos.

- **/node()**: selecciona todos los hijos (elementos o texto) del nodo.

//node(): selecciona todos los descendientes (elementos o texto) del nodo.

```
//libro/node()
```

```
<titulo>La vida está en otra parte</titulo>
<autor>Milan Kundera</autor>
<fechaPublicacion año="1973"/>
<titulo>Pantaleón y las visitadoras</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1973"/>
<titulo>Conversación en la catedral</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1969"/>
```

```
//autor/node()
```

```
Milan Kundera
Mario Vargas Llosa
Mario Vargas Llosa
```

XPATH:SELECCIÓN DE NODOS(I)

La selección de nodos se escribe a continuación del eje y el predicado. Si el eje y el predicado han seleccionado unos nodos, la selección de nodos indica con qué parte de esos nodos nos quedamos.

- **/node()**: selecciona todos los hijos (elementos o texto) del nodo.

//node(): selecciona todos los descendientes (elementos o texto) del nodo.

```
//libro//node()
```

```
<titulo>La vida está en otra parte</titulo>
La vida está en otra parte
<autor>Milan Kundera</autor>
Milan Kundera
<fechaPublicacion año="1973"/>
<titulo>Pantaleón y las visitadoras</titulo>
Pantaleón y las visitadoras
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
Mario Vargas Llosa
<fechaPublicacion año="1973"/>
<titulo>Conversación en la catedral</titulo>
Conversación en la catedral
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
Mario Vargas Llosa
<fechaPublicacion año="1969"/>
```

En este caso difiere la respuesta respecto a /libro/node() en que devuelve el elemento y el nodo texto también para cada uno de los hijos

XPATH:SELECCIÓN DE NODOS (III)

/text(): selecciona únicamente el texto contenido en el nodo.
//text(): selecciona únicamente el texto contenido en el nodo y todos sus descendientes.

`//autor/text()`	`Milan Kundera`
	`Mario Vargas Llosa`
	`Mario Vargas Llosa`

`//libro/text()`	No devuelve nada porque <libro> no contiene texto.

`//libro//text()`	`La vida está en otra parte`
	`Milan Kundera`
	`Pantaleón y las visitadoras`
	`Mario Vargas Llosa`
	`Conversación en la catedral`
	`Mario Vargas Llosa`

XPATH:SELECCIÓN DE NODOS (IV)

/text(): selecciona únicamente el texto contenido en el nodo.
//text(): selecciona únicamente el texto contenido en el nodo y todos sus descendientes.

`//autor/text()`	`Milan Kundera`
	`Mario Vargas Llosa`
	`Mario Vargas Llosa`

`//libro/text()`	No devuelve nada porque <libro> no contiene texto.

`//libro//text()`	`La vida está en otra parte`
	`Milan Kundera`
	`Pantaleón y las visitadoras`
	`Mario Vargas Llosa`
	`Conversación en la catedral`
	`Mario Vargas Llosa`

XPATH:SELECCIÓN DE NODOS (V)

/*: selecciona todos los hijos (sólo elementos) del nodo.
//*: selecciona todos los descendientes (sólo elementos) del nodo.

```
/biblioteca/*
```

```
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

```
//autor/*
```

No devuelve nada porque <autor> sólo contiene texto.

La diferencia con node() es que node() devuelve también el texto y aquí solo se devuelven los elementos

XPATH:SELECCIÓN DE NODOS (VI)

```
//autor/*
```

No devuelve nada porque <autor> sólo contiene texto.

```
/biblioteca//*
```

```
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
<titulo>La vida está en otra parte</titulo>
<autor>Milan Kundera</autor>
<fechaPublicacion año="1973"/>
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<titulo>Pantaleón y las visitadoras</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1973"/>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
<titulo>Conversación en la catedral</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1969"/>
```

La segunda expresión xpath devuelve todos los hijos de biblioteca

XPATH:SELECCIÓN DE NODOS (VII)

- */@**: selecciona todos los atributos del nodo.
  *//@**: selecciona todos los atributos de los descendientes del nodo.

`//@*`	`año="1973"`
	`fechaNacimiento="28/03/1936"`
	`año="1973"`
	`fechaNacimiento="28/03/1936"`
	`año="1969"`

`//libro/@*`	No devuelve nada porque <libro> no tiene atributos.
`//autor/@*`	`fechaNacimiento="28/03/1936"` `fechaNacimiento="28/03/1936"`

Nota: En XPath 1.0 no se puede seleccionar únicamente el valor del atributo, sino que se obtienen respuestas del tipo nombreDelAtributo=ValorDelAtributo

XPATH: PASOS CONSECUTIVOS

Una expresión XPath puede contener varios pasos de búsqueda consecutivos. Cada uno incluirá su eje (y en su caso, su predicado) y el último paso de búsqueda incluirá en su caso una selección de nodos. Cada paso de búsqueda trabaja a partir de los nodos seleccionados por el paso de búsqueda anterior.

En el ejemplo siguiente se obtienen los títulos de los libros publicados después de 1970, mediante dos pasos de búsqueda:

- en el primer paso (//fechaPublicacion[@año>1970]) se seleccionan los elementos <fechaPublicacion> cuyo atributo año es superior a 1970.
- en el segundo paso (/../titulo), se seleccionan primero los elementos padre (/..) de los <fechaPublicacion> seleccionados en el primer paso de búsqueda (es decir, elementos <libro>) y a continuación sus subelementos <titulo>.

```
//fechaPublicacion[@año>1970]/../titulo
```

```
<titulo>La vida está en otra parte</titulo>
<titulo>Pantaleón y las visitadoras</titulo>
```

XPATH: PASOS CONSECUTIVOS

Un determinado resultado se puede obtener mediante un sólo paso de búsqueda o mediante varios pasos.

•En los ejemplos siguientes se obtienen los libros escritos por Mario Vargas Llosa de dos formas distintas:

- mediante un sólo paso de búsqueda. Se seleccionan los elementos <libro> cuyo subelemento <autor> tiene como contenido la cadena "Mario Vargas Llosa".

```
//libro[autor="Mario Vargas Llosa"]
```

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

mediante dos pasos de búsqueda. En el primer paso se seleccionan los elementos <autor> cuyo contenido es la cadena "Mario Vargas Llosa". En el segundo paso de búsqueda se seleccionan los elementos padre (es decir, los elementos <libro>).

```
//autor[.="Mario Vargas Llosa"]/..
```

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

XPATH: PASOS CONSECUTIVOS

En los ejemplos siguientes se obtiene el autor que ha publicado libros en 1969 de varias formas distintas:

`//@año[.=1969]/../../autor`	`<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>`
`//libro[fechaPublicacion/@año=1969]/autor`	`<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>`
`//fechaPublicacion[@año=1969]/../autor`	`<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>`
`//autor[../fechaPublicacion/@año=1969]`	`<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>`

XPATH: EXPRESIONES ANIDADAS

Las expresiones XPath pueden anidarse, lo que permite definir expresiones más complicadas. Por ejemplo, en el documento utilizado anteriormente:

Un ejemplo de expresión anidada sería, por ejemplo, obtener los títulos de los libros publicados el mismo año que la novela "La vida está en otra parte". Esta información no está directamente almacenada en el documento, pero se puede obtener la respuesta en dos pasos:

P.E.: obtener primero el año en que se publicó la novela "La vida está en otra parte":

```
//libro[titulo="La vida está en otra parte"]/fechaPublicacion/@año
```

```
año="1973"
```

y obtener después los títulos de los libros publicados en 1973:

```
//libro[fechaPublicacion/@año=1973]/titulo
```

```
<titulo>La vida está en otra parte</titulo>
<titulo>Pantaleón y las visitadoras</titulo>
```

Estas dos expresiones se pueden unir en una única expresión, sustituyendo en la segunda expresión el valor 1973 por la primera expresión:

```
//libro[fechaPublicacion/@año=//libro[titulo="La vida está en otra parte"]/fechaPublicacion/@año]
/titulo
```

```
<titulo>La vida está en otra parte</titulo>
<titulo>Pantaleón y las visitadoras</titulo>
```

Como cada una de las expresiones puede escribirse de varias maneras, en realidad hay muchas formas de encontrar la respuesta. Por ejemplo, en la solución siguiente los predicados se encuentran al final del eje en cada subexpresión:

```
//titulo[../fechaPublicacion/@año=//@año[../../titulo="La vida está en otra parte"]]
```

```
<titulo>La vida está en otra parte</titulo>
<titulo>Pantaleón y las visitadoras</titulo>
```

XPATH: EXPRESIONES ANIDADAS

títulos de los libros del mismo autor que la novela "Pantaleón y las visitadoras".

```
//libro[autor=//libro[titulo="Pantaleón y las visitadoras"]/autor/text()]/titulo
```

Resultado:

```
<titulo>Pantaleón y las visitadoras</titulo>
<titulo>Conversación en la catedral</titulo>
```

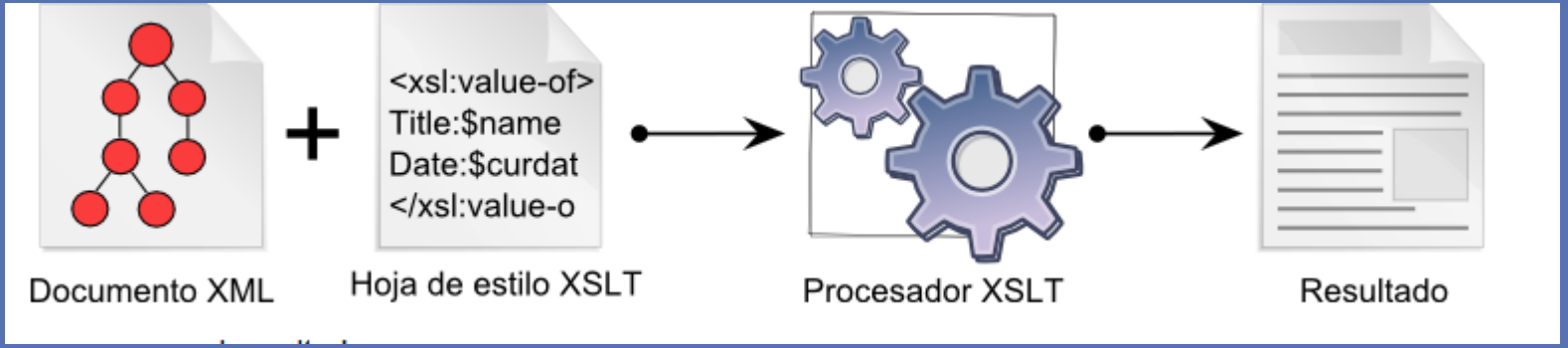
También podría haberse obtenido mediante:

```
//libro[autor=//libro[titulo="Pantaleón y las visitadoras"]/autor]/titulo
```

TRANSFORMACION DE DOCUMENTOS XML

- **XSLT**: (E***X***TENSIBLE ***S***TYLESHEET ***L***ANGUAGE ***T***RANSFORMATION)PERMITE DEFINIR EL MODO DE TRANSFORMAR UN DOCUMENTO XML EN OTRO.
- Es un lenguaje declarativo, como SQL, le decimos lo que queremos sin tener que especificar como hacerlo.
- Veremos características comunes a todas las versiones:
- Noviembre de 1999: XSLT 1.0
- Enero de 2007: XSLT 2.0
- Junio de 2017: XSLT 3.0
- Marzo de 2021: XSLT 2.0 (2ª ed)

TRANSFORMACION DE DOCUMENTOS XML



- El documento XML → es el documento inicial a partir del cual se va a generar el resultado.
- La hoja de estilo XSLT → las reglas de transformación que se van a aplicar al documento inicial.
- Procesador XSLT → programa de ordenador que aplica al documento inicial las reglas de transformación incluidas en la hoja de estilo XSLT y genera el documento final.
- Resultado → El resultado de la ejecución del programa es un nuevo documento (xml u otro formato).

TRANSFORMACION DE DOCUMENTOS XML

XSLT es un estándar aprobado por el W3C, ¿pero una hoja XSLT es también un documento XML?

Sí, XSLT es uno de los lenguajes derivados de XML, por tanto las hojas XSLT también son documentos XML (al igual que sucede con los canales RSS, atom o los documentos XSD).

¿Qué transformaciones podemos realizar sobre un documento XML usando XSLT?

- A otro documento XML.
- A un documento HTML.
- A un documento de texto.
- Etc…….

TRANSFORMACION DE DOCUMENTOS XML

PASOS PARA TRANSFORMAR EL DOCUMENTO

•El procesador analiza el documento y construye el árbol del documento.
•El procesador recorre el árbol del documento desde el nodo raíz.
•En cada nodo recorrido, el procesador aplica o no alguna plantilla:

- Si a un nodo no se le puede aplicar ninguna plantilla, su contenido se incluye en el documento final (el texto del nodo, no el de los nodos descendientes).
- A continuación, el procesador recorre sus nodos hijos.
- Si a un nodo se le puede aplicar una plantilla, se aplica la plantilla.
- La plantilla puede generar texto que se incluye en el documento final.
- En principio, el procesador no recorre sus nodos hijos, salvo que la plantilla indique al procesador que sí que deben recorrerse los nodos hijos.
- Cuando el procesador ha recorrido el árbol, se ha terminado la transformación.

TRANSFORMACION DE DOCUMENTOS XML

EJEMPLO DE HOJA DE ESTILO:

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<XSL:STYLESHEET XMLNS:XSL="HTTP://WWW.W3.ORG/1999/XSL/TRANSFORM" VERSION="1.0">
</XSL:STYLESHEET>
```

ES LA HOJA DE ESTILO MAS SIMPLE. CONSTA DE:

- LA DECLARACIÓN XML <?XML>, PROPIA DE CUALQUIER DOCUMENTO XML.
- LA INSTRUCCIÓN <XSL:STYLESHEET> ES LA ETIQUETA RAÍZ DE LA HOJA DE ESTILO,

SUS ATRIBUTOS INDICAN LA VERSIÓN Y EL ESPACIO DE NOMBRES CORRESPONDIENTE.

TRANSFORMACION DE DOCUMENTOS XML

Elementos XSLT.

El elemento raíz de una hoja XSLT es xsl:stylesheet o xsl:transform, que son prácticamente equivalentes. Sus atributos principales son:
version, cuyo valor puede ser 1.0, 2.0, 3.0.
xmlns:xsl, se utiliza para declarar el espacio de nombres xsl. Para XSLT suele ser la dirección:
http://www.w3.org/1999/XSL/Transform

TRANSFORMACION DE DOCUMENTOS XML

Elementos XSLT.

Los más destacados son:

xsl:attribute, añade un atributo a un elemento en el árbol de resultados.

xsl:choose, permite decidir que parte de la hoja XSL se va a procesar en función de varios resultados.

xsl:decimal-format, define un patrón que permite convertir en cadenas de texto números en coma flotante.

xsl:for-each, aplican sentencias a cada uno de los nodos del árbol que recibe como argumento.

xsl:if, permite decidir si se va a procesar o no una parte del documento XSL en función de una condición

xsl:key, define una o varias claves que pueden ser referenciadas desde cualquier lugar del documento.

xsl:output, define el tipo de salida que se generará como resultado.

xsl:sort permite aplicar un template a una serie de nodos ordenándolos alfabético numéricamente.

xsl:template, es el bloque fundamental de una hoja XSLT, por lo que veremos su descripción en el apartado siguiente.

xsl:value-of, calcula el valor de una expresión XPath dada y lo inserta en el árbol de resultados del documento de salida.

xsl:variable, asigna un valor a una etiqueta para usarlo cómodamente.

ELEMENTOS XSLT: TEMPLATE

Un template contiene reglas a aplicar cuando se hace “match” con un determinado nodo.
Para realizar un “match” se utiliza una expresión XPATH:
Dado el siguiente XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
 <libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
 </libro>
 <libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
 </libro>
 <libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
 </libro>
</biblioteca>
```

ELEMENTOS XSLT: TEMPLATE

El siguiente XSL:

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

</xsl:stylesheet>
```

- En el ejemplo anterior, el resultado incluye el contenido de los nodos <título> y <autor> puesto que no hay ninguna plantilla, ya que si a un nodo no se le puede aplicar ninguna plantilla, su contenido se incluye en el documento final (el texto del nodo, no las etiquetas de los nodos descendientes).

ELEMENTOS XSLT: TEMPLATE

Y dada la siguiente plantilla XSLT:

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

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

</xsl:stylesheet>
```

Este template selecciona los nodos autor pero no les aplica una regla concreta, por lo que al aplicarse nodo por nodo, solo se generan los títulos.

Hay que tener en cuenta que dentro de <xsl:template> </xsl:template> tiene que haber cualquier código xhtml bien formado. XHTML (eXtensible HyperText Markup Language) no es nada mas que HTML expresado como XML válido. XHTML buscaba ser una alternativa extensible, interoperable y estandarizada al HTML, aunque es una especificación fallida.

ELEMENTOS XSLT: VALUE-OF

EXTRAE EL CONTENIDO DEL NODO SELECCIONADO:

EN EL EJEMPLO SIGUIENTE, EL DOCUMENTO FINAL CONTIENE LOS AUTORES DE LOS LIBROS PORQUE LA PLANTILLA LOS GENERA CON LA INSTRUCCIÓN <XSL:VALUE-OF>. COMO SE HA APLICADO UNA PLANTILLA AL NODO <LIBRO>, SUS HIJOS (<TITULO>, <AUTOR> Y <FECHAPUBLICACION>) NO SE RECORREN. POR ESO, LOS TÍTULOS DE LOS LIBROS NO APARECEN EN EL DOCUMENTO FINAL.

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

  <xsl:template match="libro">
     <xsl:value-of select="autor"/>
  </xsl:template>

</xsl:stylesheet>
```

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

  Milan Kundera
  Mario Vargas Llosa
  Mario Vargas Llosa
```

ELEMENTOS XSLT: VALUE-OF

EXTRAE EL CONTENIDO DEL NODO SELECCIONADO:

EN EL EJEMPLO SIGUIENTE, EL DOCUMENTO FINAL CONTIENE LOS AUTORES DE LOS LIBROS PORQUE LA PLANTILLA LOS GENERA CON LA INSTRUCCIÓN <XSL:VALUE-OF>.

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

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

</xsl:stylesheet>
```

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

  La vida está en otra parteMilan Kundera
  Pantaleón y las visitadorasMario Vargas Llosa
  Conversación en la catedralMario Vargas Llosa
```

ELEMENTOS XSLT: VALUE-OF

EN EL EJEMPLO SIGUIENTE, LAS FECHAS DE PUBLICACIÓN SE OBTIENEN GRACIAS A LA REGLA QUE EXTRAEN EL VALOR DEL ATRIBUTO Y LOS TÍTULOS Y AUTORES SE OBTIENEN PORQUE AL NO HABER REGLAS PARA ESE NODO SE EXTRAE EL CONTENIDO. ESTO SE VE CLARAMENTE SI SE AÑADE UNA CADENA AL PRINCIPIO ANTES DEL VALUE-OF (PUEDE PONERSE “AÑO”).

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

  <xsl:template match="fechaPublicacion">
     <xsl:value-of select="@año"/>
  </xsl:template>

</xsl:stylesheet>
```

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

    La vida está en otra parte
    Milan Kundera
    1973

    Pantaleón y las visitadoras
    Mario Vargas Llosa
    1973

    Conversación en la catedral
    Mario Vargas Llosa
    1969
```

ELEMENTOS XSLT: VALUE-OF

EN EL EJEMPLO SIGUIENTE, LAS FECHAS DE PUBLICACIÓN SE OBTIENEN GRACIAS A LA REGLA QUE EXTRAEN EL VALOR DEL ATRIBUTO Y LOS TÍTULOS Y AUTORES SE OBTIENEN PORQUE AL NO HABER REGLAS PARA ESE NODO SE EXTRAE EL CONTENIDO:

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

  <xsl:template match="fechaPublicacion">
     <xsl:value-of select="@año"/>
  </xsl:template>

</xsl:stylesheet>
```

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

    La vida está en otra parte
    Milan Kundera
    1973

    Pantaleón y las visitadoras
    Mario Vargas Llosa
    1973

    Conversación en la catedral
    Mario Vargas Llosa
    1969
```

ELEMENTOS XSLT: GENERAR TEXTO

EN EL EJEMPLO SIGUIENTE SE OBTIENEN LOS NOMBRES DE LOS AUTORES PORQUE LA REGLA SELECCIONA EL NODO <LIBRO>, PERO ADEMÁS GENERAMOS LAS ETIQUETAS <P>.

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

  <xsl:template match="libro">
     <p><xsl:value-of select="autor"/></p>
 </xsl:template>

  <xsl:template match="libro">
     <p><xsl:value-of select="titulo"/></p>
  </xsl:template>

</xsl:stylesheet>
```

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

<p>La vida está en otra parte</p>

<p>Pantaleón y las visitadoras</p>

<p>Conversación en la catedral</p>
```

EN ESTE CASO SI TENEMOS DOS REGLAS PARA EL MISMO NODO, EL PROCESADOR SÓLO APLICA UNA DE ELLAS (LA ÚLTIMA, EN ESTE CASO). LOS EJEMPLOS DE ARRIBA MUESTRAN QUE PARA OBTENER LOS RESULTADOS EXACTOS QUE BUSCAMOS ES MEJOR USAR APPLY-TEMPLATES PARA EVITAR EFECTOS NO BUSCADOS DE XSLT.

EJEMPLOS

- DADO EL XML SIGUIENTE:

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
 <libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
 </libro>
 <libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
 </libro>
 <libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
 </libro>
</biblioteca>
```

EJEMPLOS

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/biblioteca">
<xsl:value-of select="."/> ||
</xsl:template>
</xsl:stylesheet>
```

La vida está en otra parte Milan Kundera Pantaleón y las visitadoras Mario Vargas Llosa Conversación en la catedral Mario Vargas Llosa ||

El template selecciona solo los Valores de /biblioteca e inserta “||”. Se observa como ha procesado todo el xml en una sola pasada (pues hay un solo “||” en la salida).

EJEMPLOS

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/biblioteca/libro">
<xsl:value-of select="."/> ||
</xsl:template>
</xsl:stylesheet>
```

La vida está en otra parte Milan Kundera || Pantaleón y las visitadoras Mario Vargas Llosa || Conversación en la catedral Mario Vargas Llosa ||

El template selecciona solo los Valores de /biblioteca/libro e inserta “||”. Se observa ha procesado el xslt en tres bloques (pues hay tres bloques “||”).

EJEMPLOS

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/biblioteca/libro">
<xsl:value-of select="fechaPublicacion/@año/../.."/> ||
</xsl:template>
</xsl:stylesheet>
```

La vida está en otra parte Milan Kundera || Pantaleón y las visitadoras Mario Vargas Llosa || Conversación en la catedral Mario Vargas Llosa ||

Se solicita el “abuelo” del @año, xslt (xpath) trata los atributos como si fueran un hijo mas del nodo.

EJEMPLOS

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

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

<xsl:template match="libro">

    <p><xsl:value-of select="fechaPublicacion/@año"/></p>

 </xsl:template>

</xsl:stylesheet>
```

```
<p>1973</p>
<p>1973</p>
<p>1969</p>
```

Se extrae el atributo año de cada libro

EJEMPLOS

Se modifica el XML para que tenga dos títulos y se pueda seleccionar uno de ellos

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

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

<xsl:template match="libro">

    <p><xsl:value-of select="titulo[2]"/></p>

 </xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
 <libro>
  <titulo>La vida está en otra parte</titulo>
  <titulo>Life is in another place</titulo>

</libro>
 <libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <titulo>Pantaleón and the visitors</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas
Llosa</autor>
</libro>
 <libro>
  <titulo>Conversación en la catedral</titulo>
  <titulo>Conversation in the cathedral </titulo>
</libro>
</biblioteca>
```

```
 <p>Life is in another place</p>
 <p>Pantaleon and the visitors</p>
 <p>Conversation in the cathedral</p>
```

ELEMENTOS XSLT: GENERAR TEXTO

ADEMÁS DE GENERAR ETIQUETAS, SE PUEDE GENERAR TEXTO:

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

  <xsl:template match="libro">
    <p><xsl:value-of select="autor"/> escribió "<xsl:value-of select="titulo"/>"</p>
  </xsl:template>

</xsl:stylesheet>
```

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

<p>Milan Kundera escribió "La vida está en otra parte"</p>

<p>Mario Vargas Llosa escribió "Pantaleón y las visitadoras"</p>

<p>Mario Vargas Llosa escribió "Conversación en la catedral"</p>
```

ELEMENTOS XSLT: APLICAR REGLAS DE TRANSFORMACION

LA INSTRUCCIÓN <XSL:APPLY-TEMPLATES> HACE QUE SE APLIQUEN A LOS SUBELEMENTOS LAS REGLAS QUE LES SEAN APLICABLES.

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

  <xsl:template match="/">
    <html>
      <h1>Autores</h1>
      <xsl:apply-templates />
    </html>
  </xsl:template>

  <xsl:template match="libro">
     <p><xsl:value-of select="autor"/></p>
  </xsl:template>

</xsl:stylesheet>
```

APLICA UN REGLA DE TEMPLATE AL ELEMENTO EN CURSO O A SUS HIJOS.

ELEMENTOS XSLT: APLICAR REGLAS DE TRANSFORMACION

LA INSTRUCCIÓN <XSL:APPLY-TEMPLATES> HACE QUE SE APLIQUEN A LOS SUBELEMENTOS LAS REGLAS QUE LES SEAN APLICABLES. RESULTADO DE LA APLICACIÓN DE LA DIAPOSITIVA ANTERIOR:

```
<?xml version="1.0" encoding="UTF-8"?>
<html><h1>Autores</h1>
  <p>Milan Kundera</p>
  <p>Mario Vargas Llosa</p>
  <p>Mario Vargas Llosa</p>
</html>
```

APLICA UN REGLA DE TEMPLATE AL ELEMENTO EN CURSO O A SUS HIJOS.

ELEMENTOS XSLT: APLICAR REGLAS DE TRANSFORMACION

LA INSTRUCCIÓN <XSL:APPLY-TEMPLATES> HACE QUE SE APLIQUEN A LOS SUBELEMENTOS LAS REGLAS QUE LES SEAN APLICABLES. RESULTADO DE LA APLICACIÓN DE LA DIAPOSITIVA ANTERIOR:

```
<?xml version="1.0" encoding="UTF-8"?>
<html><h1>Autores</h1>
  <p>Milan Kundera</p>
  <p>Mario Vargas Llosa</p>
  <p>Mario Vargas Llosa</p>
</html>
```

LE DICE AL PROCESADOR XSLT QUE APLIQUE EL TEMPLATE APROPIADO BASANDOSE EN EL TIPO Y CONTEXTO DEL NODO SELECCIONADO. APPLY-TEMPLATES PUEDE IR ACOMPAÑADO DE UN “SELECT”

ELEMENTOS XSLT: APLICAR REGLAS DE TRANSFORMACION

<XSL:APPLY-TEMPLATES> CON SELECT:

```
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>

<xsl:template match="cd">
  <p>
  <xsl:apply-templates select="title"/>
  <xsl:apply-templates select="artist"/>
  </p>
</xsl:template>

<xsl:template match="title">
  Title: <span style="color:#ff0000">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>
```

LA PRIMERA REGLA GENERA UN CODIGO HTML Y APLICA TEMPLATES A TODOS LOS ELEMENTOS DE LA ENTRADA,
SI ALGUNO DE ELLOS COINCIDE CON LA EXPRESION XPATH, APLICA LOS TEMPLATES NECESARIOS, EN ESTE CASO, VA PROCESANDO NODOS HASTA ENCONTRAR COINCIDENCIA CON TITULO Y ARTISTA

ELEMENTOS XSLT: FOR-EACH

<XSL:FOR-EACH SELECT ….> SELECCIONA UNO A UNO TODOS LOS ELEMENTOS DE UNA LISTA DETERMINADA.LOS ELEMENTOS QUE VAN A USARSE ESTAN FILTRADOS POR EL ATRIBUTO SELECT, EL CUAL UTILIZARA UNA EXPRESION XPATH.

EJ: DADO EL SIGUIENTE XML:

```
<TIENDA>
        <RAQUETA>
          <MARCA> BABOLAT</MARCA>
          <MODELO>PURE DRIVE</MODELO>
          <ANIO>2012</ANIO>
          <PRECIO>170</PRECIO>
        </RAQUETA>
        <RAQUETA>º
          <MARCA>BABOLAT</MARCA>
          <MODELO>PURE STORM</MODELO>
          <ANIO>2013</ANIO>
          <PRECIO>190</ANIO>
        </RAQUETA>
        <RAQUETA>
          <MARCA>YONEX</MARCA>
          <MODELO>MPTOUR R</MODELO>
          <ANIO>1997</ANIO>
          <PRECIO>150</PRECIO>
        </RAQUETA>
        <RAQUETA>
          <MARCA>WILSON</MARCA>
          <MODELO>HAMMER</MODELO>
          <ANIO>2000</ANIO>
          <PRECIO>95</PRECIO>
        </RAQUETA>
        <RAQUETA>
          <MARCA>WILSON</MARCA>
          <MODELO>PROSTAFF</MODELO>
          <ANIO>1997</ANIO>
          <PRECIO>215</PRECIO>
        </RAQUETA>
</TIENDA>
```

ELEMENTOS XSLT: FOR-EACH

OBTENER LA SIGUIENTE TABLA USANDO FOR-EACH:

MARCA	MODELO
BABOLAT	PURE DRIVE
BABOLAT	PURE STORM
YONEX	MPTOUR R
WILSON	HAMMER
WILSON	PROSTAFF

ELEMENTOS XSLT: FOR-EACH

CODIGO HTML A OBTENER:

```
<html>
  <body>
    <h2/>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>MARCA</th>
        <th>MODELO</th>
      </tr>
      <tr>
        <td> BABOLAT</td>
        <td>PURE DRIVE</td>
      </tr>
      <tr>
        <td>BABOLAT</td>
        <td>PURE STORM</td>
      </tr>
      <tr>
        <td>YONEX</td>
        <td>MPTOUR R</td>
      </tr>
      <tr>
        <td>WILSON</td>
        <td>HAMMER</td>
      </tr>
      <tr>
        <td>WILSON</td>
        <td>PROSTAFF</td>
      </tr>
    </table>
  </body>
</html>
```

ELEMENTOS XSLT: FOR-EACH

SOLUCION:

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

                    <xsl:template match="/">
                        <html>
                        <body>
                        <h2></h2>
                        <table border="1">
                         <tr bgcolor="#9acd32">
                          <th>MARCA</th>
                          <th>MODELO</th>
                         </tr>
                         <xsl:for-each select="TIENDA/RAQUETA">
                                <tr>
                                  <td><xsl:value-of select="MARCA"/></td>
                                  <td><xsl:value-of select="MODELO"/></td>
                                </tr>
                         </xsl:for-each>
                        </table>
                        </body>
                        </html>
                    </xsl:template>
</xsl:stylesheet>
```

ELEMENTOS XSLT: FOR-EACH

OBTENER SOLO LAS DE LA MARCA BABOLAT

MARCA	MODELO
BABOLAT	PURE DRIVE
BABOLAT	PURE STORM

ELEMENTOS XSLT: FOR-EACH

SOLUCION: USAR UNA CONDICION EN LA EXPRESION XPATH (VISTO ANTERIORMENTE EN LA TEORIA)

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

    <xsl:template match="/">
        <html>
        <body>
        <h2></h2>
        <table border="1">
         <tr bgcolor="#9acd32">
          <th>MARCA</th>
          <th>MODELO</th>
         </tr>
         <xsl:for-each select="TIENDA/RAQUETA[MARCA='BABOLAT']">
         <tr>
          <td><xsl:value-of select="MARCA"/></td>
          <td><xsl:value-of select="MODELO"/></td>
         </tr>
         </xsl:for-each>
        </table>
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

ELEMENTOS XSLT: FOR-EACH

OBTENER SOLO LAS DE LA MARCA BABOLAT Y PRECIO MAYOR QUE 180

MARCA	MODELO
BABOLAT	PURE DRIVE

ELEMENTOS XSLT: FOR-EACH

SOLUCION: USAR UNA CONDICION ADICIONAL EN LA EXPRESION XPATH

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

    <xsl:template match="/">
        <html>
        <body>
        <h2></h2>
        <table border="1">
         <tr bgcolor="#9acd32">
          <th>MARCA</th>
          <th>MODELO</th>
         </tr>
            <xsl:for-each select="TIENDA/RAQUETA[MARCA='BABOLAT AND PRECIO>180']">
         <tr>
          <td><xsl:value-of select="MARCA"/></td>
          <td><xsl:value-of select="MODELO"/></td>
         </tr>
         </xsl:for-each>
        </table>
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

ELEMENTOS XSLT: <XSL:IF..>

SINTAXIS:

<xsl:if test="*expression*">
...salida si la expression se evalua a true...
</xsl:if>

```
<?xml version="1.0"?>

<CentroRecreativo>

        <Miembro nivel="basica">
                <Nombre>Jose </Nombre>
                <Telefono tipo="casa">5555-1234</Telefono>
                <Telefono tipo="trabajo">5555-4321</Telefono>
                <Casilla>8700</Casilla>
        </Miembro>

        <Miembro nivel="premier">
                <Nombre>David</Nombre>
                <Telefono tipo="casa">3838-1234</Telefono>
                <Telefono tipo="trabajo">3838-4321</Telefono>
                <Casilla>5600</Casilla>
        </Miembro>

        <Miembro nivel="basica">
                <Nombre>Rogelio</Nombre>
                <Telefono tipo="casa">8888-1234</Telefono>
                <Telefono tipo="trabajo">8888-4321</Telefono>
                <Casilla>4000</Casilla>
        </Miembro>

</CentroRecreativo>
```

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

   Bienvenido <xsl:value-of select="Miembro/Nombre"/>

                <xsl:if test="Miembro/@nivel='premier'">
                  Por ser miembro especial le ofrecemos lo siguiente.....
                </xsl:if>

                <xsl:if test="Miembro/@nivel='basico'">
                  Le recordamos que si asciende su membresía a "premier"
                     obtiene.....
                </xsl:if>

</xsl:template>
```

ELEMENTOS XSLT: FOR-EACH

SOLUCION ALTERNATIVA AL EJERCICIO ANTERIOR USANDO UN IF:

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

    <xsl:template match="/">
        <html>
        <body>
        <h2></h2>
        <table border="1">
         <tr bgcolor="#9acd32">
          <th>MARCA</th>
          <th>MODELO</th>
         </tr>
            <xsl:for-each select="TIENDA/RAQUETA[MARCA='BABOLAT']">
                <xsl:if test="PRECIO>180">
                    <tr>
                    <td><xsl:value-of select="MARCA"/></td>
                    <td><xsl:value-of select="MODELO"/></td>
                    </tr>
                </xsl:if>
            </xsl:for-each>
        </table>
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

ELEMENTOS XSLT: CHOOSE

<XSL:CHOOSE..> SE USA EN CONJUNCION CON <XSL:WHEN> Y <XSL:OTHERWISE> PARA EXPRESAR VARIAS CONDICIONES POSIBLES. EJEMPLO: SI PRECIO ES MAYOR QUE 10 SE EXTRAE EL ARTISTA Y SE PONE COLOR DE FONDO A COLOR DE FONDO FF00FF (FUCSIA), SINO SE PONE EL ARTISTA NADA MAS.

```
<xsl:for-each select="catalog/cd">
<tr>
  <td><xsl:value-of select="title"/></td>
  <xsl:choose>
    <xsl:when test="price &gt; 10">
      <td bgcolor="#ff00ff">
      <xsl:value-of select="artist"/></td>
    </xsl:when>
    <xsl:otherwise>
      <td><xsl:value-of select="artist"/></td>
    </xsl:otherwise>
  </xsl:choose>
```

ELEMENTOS XSLT/XPATH:FUNCIONES

EXISTEN UNA SERIE DE FUNCIONES XPATH QUE PUEDEN USARSE EN XSLT Y XPATH. CITARLAS TODAS EXCEDE EL AMBITO DE ESTE CURSO PERO VAMOS A VER ALGUNAS COMO MUESTRA:

-SUM
-NUMBER
-MAX
-MIN
-COUNT
-POSITION

LA REFERENCIA COMPLETA PUEDE VERSE EN:

https://www.w3.org/2005/04/xpath-functions/

O EN ESPAÑOL EN LA PAGINA:

https://www.data2type.de/es/xml-xslt-xslfo/xslt/referencia-xslt-y-xpath

SUM: EJEMPLO

DADO EL XML SIGUIENTE:

```
<?xml version="1.0" encoding="utf-8"?>
<datos>
<dato>
   <number>1</number>
   <number>3</number>
   <number>4</number>
</dato>
<dato>
   <number>17</number>
   <number>8</number>
   <number>11</number>
   <number>12</number>
</dato>
</datos>
```

Obtener una xslt que muestre:

Suma parcial: 8

Suma parcial: 48

SUM: EJEMPLO

SOLUCION:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:template match="datos/dato">
      Suma parcial: <xsl:value-of select="sum(number)" />
 </xsl:template>

</xsl:stylesheet>
```

Se procesa cada uno de los grupos datos/dato y para cada uno de ellos se obtiene la suma de todos los elementos number de los que consta y se muestra en la salida.

SUM: EJEMPLO

DADO EL XML SIGUIENTE, OBTENER UNA CADENA QUE SE MUESTRE:

1+3+4+17+8+11 = 44

```
<?xml version="1.0" encoding="utf-8"?>
<data>
   <number>1</number>
   <number>3</number>
   <number>4</number>
   <number>17</number>
   <number>8</number>
   <number>11</number>
</data>
```

SUM: EJEMPLO. SOLUCION:

```
<?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:for-each select="//number“ <!– ITERAMOS SOBRE LOS ELEMENTOS NUMBER DEL RAIZ -->
     <xsl:value-of select="."/> <!– COGEMOS EL VALOR ACTUAL -->
     <xsl:choose>
      <xsl:when test="position()=last()">   <! SI LLEGAMOS A LA ULTIMA POSICION DEL ELEMENTO DATA-- >
       <xsl:text> = </xsl:text> <!– INSERTAMOS UN SIGNO IGUAL EN LA SALIDA -- >
      </xsl:when>
      <xsl:otherwise> <! – EN CASO CONTRARIO ,INSERTAMOS UN MAS>
       <xsl:text> + </xsl:text>
      </xsl:otherwise>
     </xsl:choose>
    </xsl:for-each> <!– COGEMOS LA SUMA DE TODOS LOS VALORES NUMBER DEL DOCUMENTO -- >
    <xsl:value-of select="sum(//number)"/>
   </xsl:template>
</xsl:stylesheet>
```

SUM: EJEMPLO2:

DADO EL XML SIGUIENTE, OBTENER UNA CADENA QUE MUESTRE:

Subtotal

1 + 3 + 4 = 8

Subtotal

17 + 8 + 11 = 36

```
<?xml version="1.0" encoding="utf-8"?>
<datos>
<dato>
   <number>1</number>
   <number>3</number>
   <number>4</number>
</dato>
<dato>
   <number>17</number>
   <number>8</number>
   <number>11</number>
</dato>
</datos>
```

SUM: EJEMPLO2:

SOLUCION:

```
<?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:for-each select="/datos/data">
Subtotal
    <xsl:for-each select="number">
     <xsl:value-of select="."/>
     <xsl:choose>
      <xsl:when test="position()=last()">
       <xsl:text> = </xsl:text>
      </xsl:when>
      <xsl:otherwise>
       <xsl:text> + </xsl:text>
      </xsl:otherwise>
     </xsl:choose>
    </xsl:for-each>
    <xsl:value-of select="sum(number)"/>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

SUM: EJEMPLO2:

DADO EL XML SIGUIENTE, OBTENER UNA CADENA QUE MUESTRE:

Subtotal

1 + 3 + 4 = 8

Subtotal

17 + 8 + 11 = 36

Total:44

```
<?xml version="1.0" encoding="utf-8"?>
        <datos>
        <dato>
          <number>1</number>
          <number>3</number>
          <number>4</number>
        </dato>
        <dato>
          <number>17</number>
          <number>8</number>
          <number>11</number>
        </dato>
        </datos>
```

SUM: EJEMPLO2:

SOLUCION:

```
<?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:for-each select="/datos/data">
Subtotal
     <xsl:for-each select="number">
      <xsl:value-of select="."/>
      <xsl:choose>
        <xsl:when test="position()=last()">
          <xsl:text> = </xsl:text>
        </xsl:when>
        <xsl:otherwise>
          <xsl:text> + </xsl:text>
        </xsl:otherwise>
      </xsl:choose>
     </xsl:for-each>
     <xsl:value-of select="sum(number)"/>
     </xsl:for-each>
Total:<xsl:value-of select="sum(//number)"/> <!– se hace la suma de todos los elementos number -->
   </xsl:template>
</xsl:stylesheet>
```

NUMBER: EJEMPLO:

NUMBER SE PUEDE USAR EN CONJUNCION CON VALUE PARA EXTRAER LA POSICION: (FORMAT =1 permite que se imprima en el formato ordinal normal (p.e. 1, 2,3 etc, otros podrían ser a,b,c,d etc))

```
<xsl:template match="/">
 <html>
 <body>
 <p>
 <xsl:for-each select="catalog/cd">
  <xsl:number value="position()" format="1" /> --->
  <xsl:value-of select="title" /><br />
 </xsl:for-each>
 </p>
 </body>
 </html>
</xsl:template>

</xsl:stylesheet>
```

NUMBER: EJEMPLO:

SALIDA:

1 ---> Empire Burlesque
2 ---> Hide your heart
3 ---> Greatest Hits
4 ---> Still got the blues
ETC

NUMBER: EJEMPLO:

Alternativamente:

```
<xsl:template match="/">
 <html>
 <body>
 <p>
 <xsl:for-each select="catalog/cd">
  <xsl:number value="position()" format=“a. " /> --->
  <xsl:value-of select="title" /><br />
 </xsl:for-each>
 </p>
 </body>
 </html>
</xsl:template>

</xsl:stylesheet>
```

NUMBER: EJEMPLO:

SALIDA:

a. ---> Empire Burlesque
b. ---> Hide your heart
c. ---> Greatest Hits
d. ---> Still got the blues
ETC

EJERCICIO: COUNT Y POSITION

Dado el xml siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<datos>
<dato>
   <number>1</number>
   <number>3</number>
   <number>4</number>
</dato>
<dato>
   <number>17</number>
   <number>8</number>
   <number>11</number>
   <number>12</number>
</dato>
</datos>
```

Generar la salida siguiente:

Total de elementos number: 7

Total de elementos number dentro del grupo 1 : 3

Total de elementos number dentro del grupo 2 : 4

SOLUCION

```
<?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:apply-templates select="/datos"/>
   <xsl:apply-templates select="/datos/dato"/>
</xsl:template>
<xsl:template match="/datos">
          Total de elementos number: <xsl:value-of select="count(/datos/dato/number)"/>
</xsl:template>
<xsl:template match="/datos/dato">
Total de elementos number dentro del grupo
<xsl:number value="position()" format="1" /> : <xsl:value-of select="count(number)"/>
</xsl:template>
</xsl:stylesheet>
```