LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

1º Desarrollo de Aplicaciones Multiplataforma.

# UND 6: XSL

# Contenido

# INTRODUCCIÓN.

Los documentos XML son documentos de texto con etiquetas que contienen exclusivamente información sin entrar en detalles de formato. Esto implica la necesidad de algún medio para expresar la transformación de un documento XML, para que una persona pueda utilizar directamente los datos para leer, imprimir, etc.

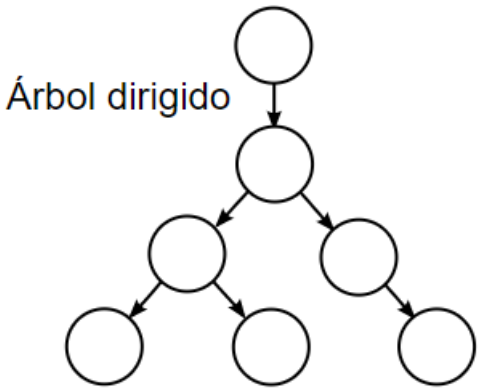
Las tecnologías que entran en juego en la transformación de documentos son:

- XSLT: (eXtensible Stylesheet Language Transformation) permite definir el modo de transformar un documento XML en otro.
- XSL-FO: (eXtensible Stylesheet Language - Formatting Object) 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

La parte de transformaciones ganó en importancia, y se llega a la terminología actual que es XSL y comprende las anteriores.

# XPATH

XPath opera sobre la estructura lógica abstracta de un documento XML y lo modela como un árbol de nodos.



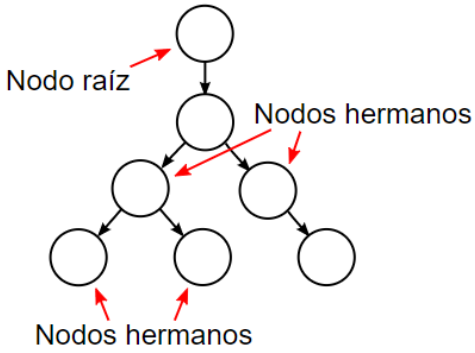

Un documento XML puede representarse como un árbol dirigido, considerando por ejemplo los elementos como nodos y que un elemento es padre de los elementos que contiene. En XPath no sólo los elementos son nodos, en realidad hay siete tipos de nodos:

- Raíz
- Elemento
- Atributo
- Texto
- Comentario
- Instrucción de procesamiento
- Espacio de nombres

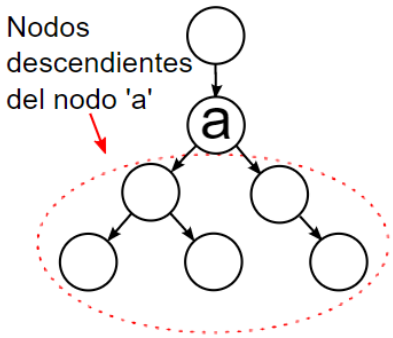
**Nota: La declaración DOCTYPE no se considera como nodo.**

## Nomenclatura de nodos

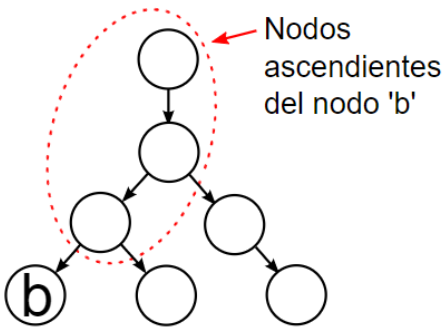
El **nodo raíz** de un árbol dirigido es el único nodo sin padre. Los **nodos hermanos** son los nodos que tienen el mismo padre.




Los **nodos descendientes** de un nodo son todos los nodos a los que se llega desde el nodo: **los hijos**, los hijos de los hijos, etc.



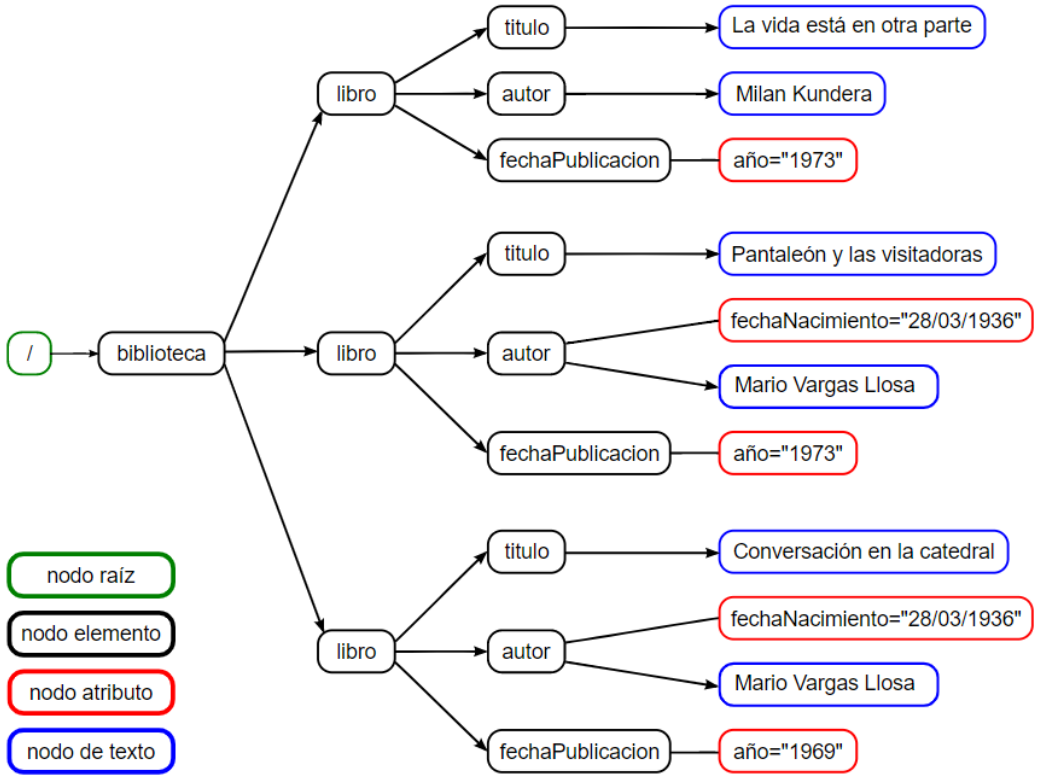

Los **nodos ascendientes** de un nodo son todos los nodos de los que un nodo es descendiente: el padre, el padre del padre, etc.



Por ejemplo, el documento 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>
```

se puede representar mediante el siguiente grafo:



Los nodos **atributos** y de **texto** no son como los nodos **elemento**. Por ejemplo, los nodos atributo y de texto no pueden tener descendientes. En realidad, el nodo atributo ni siquiera se considera como hijo, sino como una etiqueta adosada al elemento. El texto contenido por una etiqueta sí que se considera hijo del elemento, aunque **las expresiones XPath suelen trabajar con nodos elemento y para referirse a los atributos o al texto se utilizan notaciones especiales.**

## Sintaxis de las expresiones XPath

Una expresión XPath es una cadena de texto que representa un recorrido en el árbol del documento. Las expresiones más simples se parecen a las rutas de los archivos en el explorador de Windows o en la shell de GNU/Linux.

**Evaluar una expresión** XPath es buscar si hay nodos en el documento que se ajustan al recorrido definido en la expresión. El resultado de la evaluación son todos los nodos que se ajustan a la expresión. **Para poder evaluar una expresión XPath, el documento debe estar bien formado.**

Las expresiones XPath se pueden escribir de dos formas distintas:

- Sintaxis abreviada: más compacta y fácil de leer, que se explica en esta lección
- Sintaxis completa: más larga, pero con más opciones disponibles

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: restringe la selección del eje a que los nodos cumplan ciertas condiciones.
- Selección de nodos: de los nodos seleccionados por el eje y predicado, selecciona los elementos, el texto que contienen o ambos.

## Sintaxis abreviada

Veamos unos ejemplos de expresiones XPath de sintaxis abreviada y el resultado de su evaluación en el documento de ejemplo anterior:

### Ejes

El eje nos permite seleccionar un subconjunto de nodos del documento y corresponde a recorridos en el árbol del documento. Los **nodos elemento** se indican mediante el nombre del elemento. Los **nodos atributo** se indican mediante **@** y el nombre del atributo.

| ELEMENTOS DE SINTAXIS PARA EJES | |
|---|---|
| **Sintaxis** | **Descripción** |
| **/** | Selecciona desde el nodo raíz. |
| **//** | Selecciona nodos desde el nodo contextual (sin importar donde se encuentren). |
| **.** | Selecciona el nodo contextual. |
| **..** | Selecciona el padre del nodo contextual. |
| **nombre** | Selecciona los nodos hijos de nodo nombrado. |
| **@prueba** | Selecciona todos los atributos "prueba". |
| **\|** | Unión entre 2 node-sets |

(1) **Conjunto de nodos (Node-set):** es un conjunto de nodos (sin duplicados y sin orden) que se genera como resultado de evaluar una expresión XPath. Estos nodos pueden ser de cualquiera de las tipologías anteriormente descritas, pero suelen corresponderse con los tipos: elemento, atributo y texto.

- **“/”:** si está al principio de la expresión, indica el nodo raíz, si no, indica "hijo". Debe ir seguida del nombre de un elemento.

| Sintaxis XPath abreviada | Resultado |
|---|---|
| /biblioteca/libro/autor | <autor>Milan Kundera</autor><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></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"<br>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.

- “**//**”: indica "descendiente" (hijos, hijos de hijos, etc.).

| Sintaxis XPath abreviada | Resultado |
|---|---|
| /biblioteca//autor | <autor>Milan Kundera</autor><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor> |
| //autor | <autor>Milan Kundera</autor><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor> |
| //autor//libro | No devuelve nada porque <libro> no es descendiente de <autor>. |
| /@año | año="1973"<br>año="1973"<br>año="1969" |

- "**/..**": indica el elemento padre.

**Nota**: En el resultado de los ejemplos siguientes se obtienen únicamente los nodos que tienen el atributo fechaNacimiento.

| Sintaxis XPath abreviada | Resultado |
|---|---|
| /biblioteca/libro/autor/@fechaNacimiento/.. | <autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor> |
| //@fechaNacimiento/../.. | <libro><br><titulo>Pantaleón y las visitadoras<br></titulo><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor><br><fechaPublicacion año="1973"/><br></libro><br><libro><br><titulo>Conversación en la catedral<br></titulo><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor><br><fechaPublicacion año="1969"/><br></libro> |

- **"|"**: permite indicar varios recorridos.

| Sintaxis XPath abreviada | Resultado |
|---|---|
| //autor\|//titulo | <titulo>La vida está en otra parte</titulo><br><autor>Milan Kundera</autor><br><titulo>Pantaleón y las visitadoras</titulo><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor><br><titulo>Conversación en la catedral</titulo><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor> |
| //autor\|//titulo\|//@año | <titulo>La vida está en otra parte</titulo><br><autor>Milan Kundera</autor><br>año="1973"<br><titulo>Pantaleón y las visitadoras</titulo><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor><br>año="1973"<br><titulo>Conversación en la catedral</titulo><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor><br>año="1969" |

## Predicados

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.

| OPERADORES PARA LOS PREDICADOS | |
|---|---|
| **Sintaxis** | **Descripción** |
| **+** | Suma |
| **-** | Resta |
| ***** | Multiplicación |
| **div** | División |
| **=** | Igualdad |
| **!=** | Diferencia |
| **<** | Menor que |
| **>** | Mayor que |
| **<=** | Menor o igual que |
| **>=** | Mayor o igual que |
| **or** | Disyunción |
| **and** | Conjunción |
| **mod** | Módulo (resto) |

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

| Sintaxis XPath abreviada | Resultado |
|---|---|
| //autor**[@fechaNacimiento]** | `<autor fechaNacimiento="28/03/1936">`<br>`Mario Vargas Llosa`<br>`</autor>`<br>`<autor fechaNacimiento="28/03/1936">`<br>`Mario Vargas Llosa`<br>`</autor>` |

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

| Sintaxis XPath abreviada | Resultado |
|---|---|
| //libro**[1]** | `<libro>`<br>`<titulo>La vida está en otra parte</titulo>`<br>`<autor>Milan Kundera</autor>`<br>`<fechaPublicacion año="1973"/>`<br>`</libro>` |
| //libro**[last()]** | `<libro>`<br>`<titulo>`<br>Conversación en la catedral<br>`</titulo>`<br>`<autor fechaNacimiento="28/03/1936">`<br>Mario Vargas Llosa<br>`</autor>`<br>`<fechaPublicacion año="1969"/>`<br>`</libro>` |
| //libro**[last()-1]** | `<libro>`<br>`<titulo>`<br>Pantaleón y las visitadoras<br>`</titulo>`<br>`<autor fechaNacimiento="28/03/1936">`<br>Mario Vargas Llosa<br>`</autor>`<br>`<fechaPublicacion año="1973"/>`<br>`</libro>` |

- **[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: =, !=, <, >, <=, >=

  Las comparaciones se pueden hacer entre valores de nodos y atributos o con cadenas de texto o numéricas. **Las cadenas de texto deben escribirse entre comillas simples o dobles.** En el caso de las cadenas numéricas, las comillas son optativas.

  La condición puede utilizar el valor de un atributo (utilizando @) o el texto que contiene el elemento.

En los ejemplos siguientes se obtienen respectivamente los elementos <fechaPublicacion> cuyo atributo año es posterior/mayor a 1970 y los elementos <libro> cuya subelemento <autor> tiene como contenido "Mario Vargas Llosa":

| Sintaxis XPath abreviada | Resultado |
|---|---|
| //fechaPublicacion**[@año>1970]** | `<fechaPublicacion año="1973"/>`<br>`<fechaPublicacion año="1973"/>` |
| //libro**[autor="Mario Vargas Llosa"]** | `<libro>`<br>`<titulo>`<br>`Pantaleón y las visitadoras`<br>`</titulo>`<br>`<autor fechaNacimiento="28/03/1936">`<br>`Mario Vargas Llosa`<br>`</autor>`<br>`<fechaPublicacion año="1973"/>`<br>`</libro>`<br>`<libro>`<br>`<titulo>`<br>`Conversación en la catedral`<br>`</titulo>`<br>`<autor fechaNacimiento="28/03/1936">`<br>`Mario Vargas Llosa`<br>`</autor>`<br>`<fechaPublicacion año="1969"/>`<br>`</libro>` |

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

| Sintaxis XPath abreviada | Resultado |
|---|---|
| //@año**[.>1970]** | `año="1973"`<br>`año="1973"` |
| //autor**[.="Mario Vargas Llosa"]** | `<autor fechaNacimiento="28/03/1936">`<br>`Mario Vargas Llosa`<br>`</autor>`<br>`<autor fechaNacimiento="28/03/1936">`<br>`Mario Vargas Llosa`<br>`</autor>` |

**Un predicado puede contener condiciones compuestas.**

En los ejemplos siguientes se seleccionan, respectivamente, los libros escritos por Mario Vargas Llosa y publicados en 1973 (primer ejemplo) y los libros escritos por Mario Vargas Llosa o publicados en 1973 (segundo ejemplo):

| Sintaxis XPath abreviada | Resultado |
|---|---|
| //libro**[autor="Mario Vargas Llosa" and fechaPublicacion/@año="1973"]** | `<libro>`<br>`<titulo>`<br>Pantaleón y las visitadoras<br>`</titulo>`<br>`<autor fechaNacimiento="28/03/1936">`<br>Mario Vargas Llosa<br>`</autor>`<br>`<fechaPublicacion año="1973"/>`<br>`</libro>` |
| //libro**[autor="Mario Vargas Llosa" or fechaPublicacion/@año="1973"]** | `<libro>`<br>`<titulo>`La vida está en otra parte`</titulo>`<br>`<autor>`Milan Kundera`</autor>`<br>`<fechaPublicacion año="1973"/>`<br>`</libro>`<br>`<libro>`<br>`<titulo>`<br>Pantaleón y las visitadoras<br>`</titulo>`<br>`<autor fechaNacimiento="28/03/1936">`<br>Mario Vargas Llosa<br>`</autor>`<br>`<fechaPublicacion año="1973"/>`<br>`</libro>`<br>`<libro>`<br>`<titulo>`<br>Conversación en la catedral<br>`</titulo>`<br>`<autor fechaNacimiento="28/03/1936">`<br>Mario Vargas Llosa<br>`</autor>`<br>`<fechaPublicacion año="1969"/>`<br>`</libro>` |

**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:

| Sintaxis XPath abreviada | Resultado |
|---|---|
| //libro**[autor="Mario Vargas Llosa"] [fechaPublicacion/@año="1973"]**<br><br>Equivale a: *//libro**[autor="Mario Vargas Llosa" and fechaPublicacion/@año="1973"]*** | `<libro>`<br>`<titulo>`<br>Pantaleón y las visitadoras<br>`</titulo>`<br>`<autor fechaNacimiento="28/03/1936">`<br>Mario Vargas Llosa<br>`</autor>`<br>`<fechaPublicacion año="1973"/>`<br>`</libro>` |

## Expresiones anidadas

Las expresiones XPath pueden anidarse, lo que permite definir expresiones más complicadas. 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 mediante la expresión anidada:

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

Como se puede observar, la forma de hacerlo es usar una expresión Xpath como valor de comparación en la condición. **Un detalle importante es que no hay que escribir esta expresión entre comillas.**

## 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.

| ELEMENTOS DE SINTAXIS PARA EJES | |
|---|---|
| **Sintaxis** | **Descripción** |
| **node()** | Selecciona todos los nodos del documento. |
| **text()** | Selecciona el texto contenido en el nodo. |
| **//*** | Selecciona todos los nodos del documento. |
| ***** | Selecciona todos los nodos elemento. |
| **@*** | Selecciona todos los nodos atributos. |
| **/<nodo_ejemplo>/*** | Selecciona todos los los nodos hijos del nodo ejemplo. |

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

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

| Sintaxis XPath abreviada | Resultado |
|---|---|
| //libro**/node()** | <titulo>La vida está en otra parte</titulo><br><autor>Milan Kundera</autor><br><fechaPublicacion año="1973"/><br><titulo>Pantaleón y las visitadoras</titulo><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa |

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

- **"/text()"**: selecciona únicamente el texto contenido en el nodo.

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

| Sintaxis XPath abreviada | Resultado |
|---|---|
| //autor**/text()** | Milan Kundera<br>Mario Vargas Llosa<br>Mario Vargas Llosa |
| //libro**/text()** | No devuelve nada porque <libro> no contiene texto. |
| //libro**//text()** | La vida está en otra parte<br>Milan Kundera<br>Pantaleón y las visitadoras<br>Mario Vargas Llosa<br>Conversación en la catedral<br>Mario Vargas Llosa |

- **“/*”**: selecciona todos los hijos (sólo elementos) del nodo.

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

| Sintaxis XPath abreviada | Resultado |
|---|---|
| /biblioteca/* | <libro><br>  <titulo>La vida está en otra parte</titulo><br>  <autor>Milan Kundera</autor><br>  <fechaPublicacion año="1973"/><br></libro><br><libro><br>  <titulo>Pantaleón y las visitadoras</titulo><br>  <autor fechaNacimiento="28/03/1936"><br>  Mario Vargas Llosa<br>  </autor><br>  <fechaPublicacion año="1973"/><br></libro><br><libro><br>  <titulo>Conversación en la catedral</titulo><br>  <autor fechaNacimiento="28/03/1936"><br>  Mario Vargas Llosa<br>  </autor><br>  <fechaPublicacion año="1969"/><br></libro> |
| //autor/* | No devuelve nada porque <autor> sólo contiene texto. |
| /biblioteca//* | <libro><br>  <titulo>La vida está en otra parte</titulo><br>  <autor>Milan Kundera</autor><br>  <fechaPublicacion año="1973"/><br></libro><br><titulo>La vida está en otra parte</titulo><br><autor>Milan Kundera</autor><br><fechaPublicacion año="1973"/><br><libro><br>  <titulo>Pantaleón y las visitadoras</titulo><br>  <autor fechaNacimiento="28/03/1936"><br>  Mario Vargas Llosa<br>  </autor><br>  <fechaPublicacion año="1973"/><br></libro><br><titulo>Pantaleón y las visitadoras</titulo><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor><br><fechaPublicacion año="1973"/><br><libro><br>  <titulo>Conversación en la catedral</titulo><br>  <autor fechaNacimiento="28/03/1936"><br>  Mario Vargas Llosa<br>  </autor><br>  <fechaPublicacion año="1969"/><br></libro><br><titulo>Conversación en la catedral</titulo><br><autor fechaNacimiento="28/03/1936"><br>Mario Vargas Llosa<br></autor><br><fechaPublicacion año="1969"/> |

**"/@*"**: selecciona todos los atributos del nodo.

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

| Sintaxis XPath abreviada | Resultado |
|---|---|
| //@* | año="1973"<br>fechaNacimiento="28/03/1936"<br>año="1973"<br>fechaNacimiento="28/03/1936"<br>año="1969" |
| //libro/@* | No devuelve nada porque <libro> sólo contiene texto. |
| //autor/@* | fechaNacimiento="28/03/1936"<br>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

## Sintaxis completa

Veamos unos ejemplos de expresiones XPath de sintaxis completa.

### Ejes

| ELEMENTOS DE SINTAXIS PARA EJES | |
|---|---|
| **Sintaxis** | **Descripción** |
| ancestor | Selecciona todos los ancestros del nodo contextual. |
| ancestor-or-self | Selecciona todos los ancestros del nodo contextual y el mismo. |
| attribute | Selecciona todos los atributos del nodo contextual. |
| child | Selecciona todos los hijos del nodo contextual. (Suele estar implícito) |
| descendant | Selecciona todos los descendientes del nodo contextual. |
| descendant-orself | Selecciona todos los descendientes del nodo contextual y el mismo. |
| following | Selecciona todo el documento que se encuentra a partir del nodo contextual. |
| following-sibling | Selecciona a todos los hermanos posteriores al nodo contextual. |
| namespace | Selecciona todos los nodos con el namespace del nodo contextual. |
| parent | Selecciona el nodo padre del nodo contextual. |
| preceding | Selecciona todo en el documento que está antes del nodo actual. |
| preceding-sibling | Selecciona todos los hermanos anteriores al nodo actual. |
| self | Selecciona el nodo actual. |
| ancestor | Selecciona todos los ancestros del nodo contextual. |
| ancestor-or-self | Selecciona todos los ancestros del nodo contextual y el mismo. |
| attribute | Selecciona todos los atributos del nodo contextual. |
| child | Selecciona todos los hijos del nodo contextual. (Suele estar implícito) |
| descendant | Selecciona todos los descendientes del nodo contextual. |
| descendant-orself | Selecciona todos los descendientes del nodo contextual y el mismo. |

| | |
|---|---|
| following | Selecciona todo el documento que se encuentra a partir del nodo contextual. |
| following-sibling | Selecciona a todos los hermanos posteriores al nodo contextual. |
| namespace | Selecciona todos los nodos con el namespace del nodo contextual. |
| parent | Selecciona el nodo padre del nodo contextual. |
| preceding | Selecciona todo en el documento que está antes del nodo actual. |
| preceding-sibling | Selecciona todos los hermanos anteriores al nodo actual. |
| self | Selecciona el nodo actual. |

Ejemplos de sintaxis para ejes.

| Ruta no abreviada | Ruta abreviada | Significado |
|---|---|---|
| child::para | para | Selecciona los elementos *para* hijos del nodo contextual |
| child::* | * | Selecciona todos los elementos hijos del nodo contextual |
| child::text() | text() | Selecciona todos los nodos texto hijos del nodo contextual |
| attribute::name | name | Selecciona el atributo name del nodo contextual |
| attribute::* | @* | Selecciona todos los atributos del nodo contextual |
| descendant::para | .//para | Selecciona los elementos *para* descendientes del nodo contextual |
| child::chapter/ descendant::para | chapter//para | Selecciona los elementos *para* descendientes de los elementos *chapter* hijos del nodo contextual |
| child::*/child::para | */para | Selecciona todos los nietos *para* del nodo contextual |
| / | / | Selecciona la raíz del documento. |
| /descendant::para | //para | Selecciona todos los elementos *para* en el mismo documento que el nodo contextual |
| /descendant::olist/child::item | //olist/item | Selecciona todos los elementos *item* que tienen un padre *olist* y que estén en el mismo documento que el nodo contextual |
| child::para[position()=1] | para[1] | Selecciona el primer hijo *para* del nodo contextual |
| child::para[position()=last()] | para[last()] | Selecciona el último hijo *para* del nodo contextual |
| child::para[position()=last( )-1] | para[last()-1] | Selecciona el penúltimo hijo *para* del nodo contextual |
| /child::doc/child::chapter [position()=5]/child:: section[position()=2] | /doc/chapter[5]/ section[2] | Selecciona la segunda *section* del quinto *chapter* del elemento de documento *doc* |
| child::para[attribute:: type="warning"] | para[@type= "warning"] | Selecciona todos los hijos *para* del nodo contextual que tengan un atributo *type* con valor *warning* |

| | | |
|---|---|---|
| child::para[attribute:: type='warning'] [position()=5] | para[@type= "warning"][5] | Selecciona el quinto hijo *para* del nodo contextual que tenga un atributo *type* con valor *warning* |
| child::para[position()=5] [attribute::type="warning"] | para[5][@type= "warning"] | Selecciona el quinto hijo *para* del nodo contextual si ese hijo tiene un atributo *type* con valor *warning* |
| child::chapter[child:: title='Introduction'] | chapter[title= "Introduction"] | Selecciona los hijos *chapter* del nodo contextual que tengan uno o más hijos *title* cuyo valor de la cadena sea igual a *Introduction* |
| child::chapter[child::title] | chapter[title] | Selecciona los hijos *chapter* del nodo contextual que tengan uno o más hijos *title* |

## Funciones.

En XPath está permitido el uso de funciones que facilitan realizar algunas operaciones necesarias para evaluar expresiones.

Cada función se especifica utilizando un prototipo de función, que da el tipo devuelto, el nombre de la función y el tipo de los argumentos. Si un tipo de argumento es seguido por un signo de interrogación, entonces el argumento es opcional; en otro caso, el argumento es obligatorio. Estas funciones se resumen en las siguientes tablas según su tipo:

| Función | Definición |
|---|---|
| **FUNCIONES DE CONJUNTOS DE NODOS** | |
| *number***last()** | La función *last* devuelve un número igual al *tamaño contextual* del contexto de evaluación de la expresión. |
| *number***position()** | La función *position* devuelve un número igual a la *posición contextual* del contexto de evaluación de la expresión. |
| *number***count(*node-set* )** | La función *count* devuelve el número de nodos que contiene el node-set del argumento. |
| *node-set***id(*object* )** | La función *id* selecciona elementos mediante su identificador único. |
| *string***local-name ( *node-set*?)** | La función *local-name* devuelve la parte local del nombre *expandido* del nodo en el node-set del argumento. |
| *string***namespace-uri (*node-set*?)** | La función namespace-*uri* devuelve la URI del espacio de nombres del nombre *expandido* del nodo en el node-set del argumento. |
| *string***name(*node-set* ?)** | La función *name* devuelve una cadena con un *Qname* que representa el *nombre expandido* del nodo en el node-set del argumento. |
| **FUNCIONES DE CADENAS** | |
| *string***string (***object***?)** | La función string convierte un objeto en cadena. |
| *string***concat (***string, string, string****)** | La función concat devuelve la concatenación de sus argumentos. |

| | |
|---|---|
| *boolean***starts-with (***string, string* **)** | La función starts-with devuelve verdadero si el primer string del argumento empieza con el segundo string del argumento, y devuelve falso en otro caso. |
| *boolean***contains (***string, string***)** | La función contains devuelve verdadero si el primer string del argumento contiene al segundo string del argumento, y devuelve falso en otro caso. |
| *string***substring-before (***string, string* **)** | La función substring-before devuelve la subcadena del primer string del argumento que precede a la primera aparición del segundo string del argumento en el primer string del argumento, o la cadena vacía si el primer string del argumento no contiene al segundo string del argumento. Por ejemplo, substring-before ("1999/04/01","/") devuelve 1999. |
| *string***substring-after (***string, string* **)** | La función substring-after devuelve la subcadena del primer string del argumento que sigue a la primera aparición del segundo string del argumento en el primer string del argumento, o la cadena vacía si el primer string del argumento no contiene al segundo string del argumento. Por ejemplo, substring-after ("1999/04/01","/") devuelve 04/01, y substring-after ("1999/04/01","19") devuelve 99/04/01. |
| *string***substring (***string, number, number***?)** | La función substring devuelve la subcadena del primer argumento que comienza en la posición especificada por el segundo argumento y con longitud especificada por el tercer argumento. Por ejemplo, substring ("12345",2,3) devuelve "234". Si no se especifica el tercer argumento, devuelve la subcadena que comienza en la posición especificada en el segundo argumento y continúa hasta el final de la cadena. Por ejemplo, substring ("12345",2) devuelve "2345". |
| *number***string-length (***string***?)** | La función string-length devuelve el número de caracteres en la cadena. |
| *string***normalize-space (***string***?)** | La función normalize-space devuelve el string del argumento con el espacio en blanco normalizado mediante la eliminación del que se encuentra al principio y al final y la substitución de secuencias de caracteres de espacio en blanco por un solo espacio. |
| *string***translate (**string, string, string**)** | La función translate devuelve el string del primer argumento con las apariciones de caracteres del segundo argumento sustituidas por los caracteres en las posiciones correspondientes del tercer argumento.<br>Por ejemplo, translate ("bar","abc","ABC") devuelve la cadena BAr. |
| **FUNCIONES BOOLENAS** | |
| *boolean***boolean(***object***)** | La función boolean convierte su argumento en booleano de forma que:<br>• Un número es verdadero si no es ni cero positivo o negativo ni NaN.<br>• Un conjunto de nodos es verdadero si está no vacío.<br>• Una cadena es verdadera si su longitud no es cero. |

| | |
|---|---|
| *boolean***not(***boolean***)** | La función not devuelve verdadero si su argumento es falso, y falso en otro caso. |
| *boolean***true()** | La función true devuelve verdadero. |
| *boolean***false()** | La función false devuelve falso. |
| *boolean***lang(***string***)** | La función lang devuelve verdadero o falso dependiendo si el lenguaje del nodo contextual es el mismo que el que se especifica por los atributos de xml:lang, o es un sublenguaje del lenguaje especificado por el string del argumento. |
| **FUNCIONES NUMÉRICAS** | |
| *number***number(***object***?)** | La función number convierte su argumento en un número. |
| *number***sum(***node-set***)** | La función sum devuelve la suma, por cada nodo del node-set del argumento, del resultado de convertir los valores de cadena en números. |
| *number***floor(***number***)** | La función floor devuelve el mayor (más cercano al infinito positivo) número que no sea mayor que el argumento y que sea entero. |
| *number***ceiling(***number***)** | La función ceiling devuelve el menor (más cercano al infinito negativo) número que no sea menor que el argumento y que sea entero. |
| *number***round(***number***)** | La función round devuelve el número que esté más próximo al argumento y que sea entero. |

Aquí se muestran algunos ejemplos del uso de las funciones del cuadro anterior:

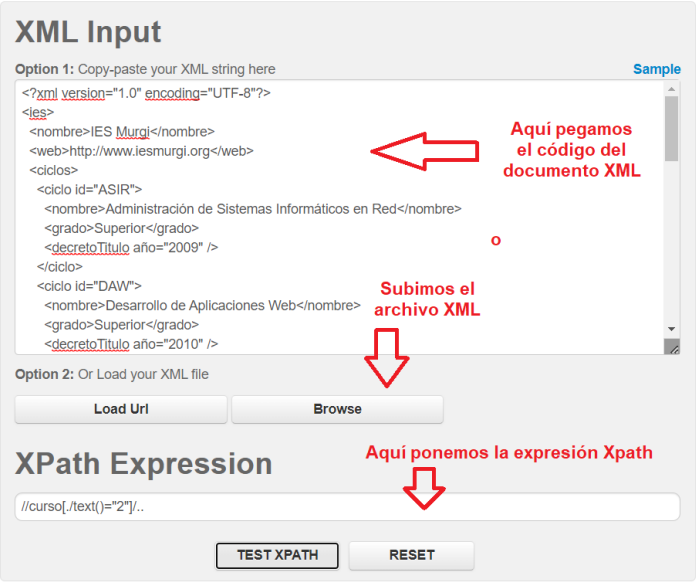
| Sintaxis funciones | Resultado |
|---|---|
| count(//book/title) | Devuelve el número de nodos seleccionados. |
| sum(//book/price) | Devuelve la suma de los valores de los nodos seleccionados. |
| //book/author[contains(text(),'De')] | Devuelve los autores cuya información contine la subcadena “De”. |

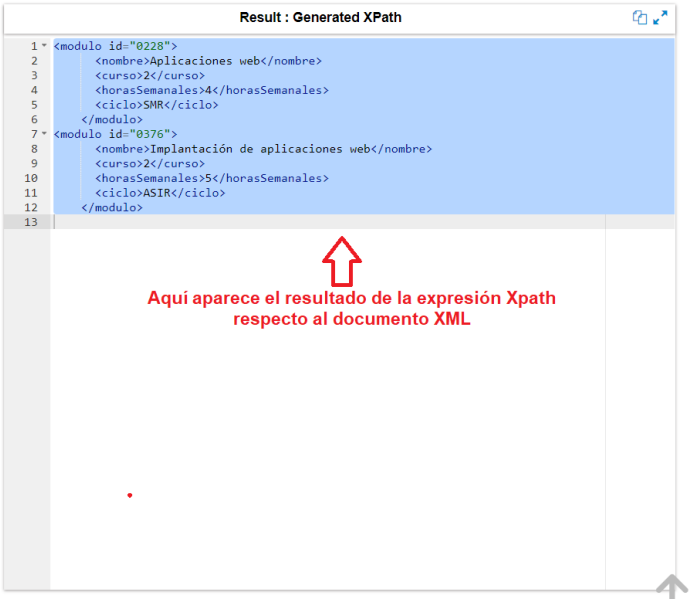
## Analizador de expresiones Xpath

Para probar expresiones XPath se puede usar el analizador de expresiones en línea Code Beautify que puedes encontrar en:

https://codebeautify.org/Xpath-Tester

Su uso es muy sencillo y se puede ver en la captura de imagen siguiente:



# XSLT

Ante la posibilidad de que distintos aplicaciones utilicen esquemas diferentes, es necesario un sistema que permita "transformar" los datos de un documento XML

**XSLT (eXtensible Stylesheet Language – Transformations),** describe un lenguaje basado en XML para transformar documentos XML a cualquier otro formato. Normalmente, utilizaremos XSLT para transformar documentos XML en HTML, WML, o cualquier otro formato que facilite su presentación en la pantalla de un ordenador o en impresora. La transformación de XML a HTML es el principal uso que se hace de XSLT.

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

## Estructura de una hoja de estilo XSLT

Una hoja de estilo XSLT **es un documento XML bien formado**. Las hojas de estilo se guardarán siempre en archivos independientes con extensión .xsl.

**El elemento raíz** de la hoja de estilo XSLT es **stylesheet**. Este elemento contendrá a todos los demás, y **debe ir precedido por el alias xsl** correspondiente al espacio de nombres para hojas de estilo XSLT. En las hojas de estilo XSLT, los nombres de los elementos "reservados" por la especificación, proceden de un mismo espacio de nombres, y por lo tanto deben escribirse precedidos por el correspondiente alias que debe "apuntar" a la URL: http://www.w3.org/1999/XSL/Transform De esta forma, el elemento raíz quedará así:

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

## Asociar una hoja de estilo a un documento

Debemos incluir, tras la declaración XML, la siguiente instrucción de procesamiento:

```
<?xml-stylesheet type=”text/xsl” href=”hojaEstilo.xsl”?>
```

Por ejemplo:

```
<?xml version="1.0"?>
<?xml-stylesheet type=”text/xsl” href=”hojaEstilo.xsl”?>
<documento>
      <titulo>Programar ASP</titulo>
      <paginas>456</paginas>
      <anno-pub>2001</anno-pub>
</documento>
```

## Creación de hojas de estilo XSL modulares: xsl:import y xsl:include

Es posible crear hojas de estilo XSLT modulares, es decir, divididas en distintos archivos físicos. En la hoja de estilo se incluirán referencias a otras hojas de estilo XSLT en las que se incluyen el resto de reglas.

Para incluir las referencias, se pueden utilizar los elementos **xsl:import** y **xsl:include**. Estos dos elementos deben ir acompañados por un elemento **href** que tomará como valor el URL absoluto o relativo de la hoja de estilo que se quiere utilizar.

Los elementos **xsl:import** se debe incluir justo a continuación de la etiqueta de inicio del elemento **xsl:stylesheet**, y antes de cualquier otro elemento.

El elemento **xsl:include** se puede incluir en cualquier lugar del documento, siempre que se escriba fuera de una regla **xsl:template**.

En cualquier hoja se podría incluir una referencia a otra hoja de estilo, utilizando la siguiente sintaxis:

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

## Abrir documentos XML con hojas de estilo XSLT en el navegador

Al abrir en un navegador una página XML enlazada con una hoja de estilo XSLT, el navegador muestra el resultado de la transformación. Pero no muestra el código fuente obtenido como resultado, sino interpretando ese código fuente, como cuando se enlaza una hoja de estilo CSS vacía.

## Plantillas

Dentro de la etiqueta del elemento raíz xsl:stylesheet, se escribirán las reglas de transformación propiamente dichas, llamadas también plantillas, definidas mediante un elemento **xsl:template**. La regla indica qué instancias de los elementos del documento XML se van a transformar, así como también indicará cómo se deben transformar cada una de ellas.
EJEMPLO:

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

La regla se aplicará a todas las instancias del elemento nombre. Esto se indica mediante el atributo match que acompaña al elemento **xsl:template**. Entre las etiquetas de inicio y de fin del elemento **xsl:template** se escribe la transformación que se debe realizar, es decir, qué texto y qué marcas se escribirán en el documento resultado de la transformación, cada vez que se encuentre una instancia del elemento nombre en el documento origen.

Con **<xsl:value-of...>**, se recupera y escribe en el documento resultado el valor del elemento que está siendo procesado.

## Ejemplo de transformación XSLT

Para el siguiente documento.xml:

```
<?xml version=“1.0”?>
<ciudades>
	<ciudad>
		<nombre>Madrid</nombre>
		<habitantes>3500000</habitantes>
	</ciudad>
	<ciudad>
		<nombre>Málaga</nombre>
		<habitantes>800000</habitantes>
	</ciudad>
	<ciudad>
		<nombre>Toledo</nombre>
		<habitantes>50000</habitantes>
	</ciudad>
</ciudades>
```

Un documento.xsl podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:template match="/">
		<html>
			<head>
				<title>Ejemplo XSLT</title>
			</head>
			<body>
				<xsl:apply-templates select=”//nombre” />
			</body>
		</html>
	 </xsl:template>
	 <xsl:template match=”//nombre”>
		<h2>
			<xsl:value-of select=”.” />
		</h2>
	 </xsl:template>
</xsl:stylesheet>
```

La regla **<xsl:template match=“/”>** se ejecuta cuando se encuentra el elemento raíz del documento XML.

Dentro de esta regla, podemos incluir llamadas a otras reglas definidas en la hoja de estilo, mediante el elemento: **<xsl:apply-templates select=”…” />**

El atributo **select** tomará como valor el nombre del elemento asociado a la regla que queremos usar.

Esto nos ofrece un control real sobre el “orden” de ejecución de las reglas.

El resultado de la transformación sería:

```
<html>
	<head>
		<title>Ejemplo XSLT</title>
	</head>
	<body>
		<h2>Madrid</h2>
		<h2>Málaga</h2>
		<h2>Toledo</h2>
	</body>
</html>
```

## Orden de procesamiento

Las reglas se van activando y ejecutando a medida que se recorre el documento origen que se quiere transformar, de esta forma**, las reglas se ejecutan en el orden en el que se van encontrando los elementos en el documento**.

**Este comportamiento por defecto puede cambiarse en las hojas de estilo XSLT**, a diferencia de lo que sucedía en las hojas de estilo CSS. Esto permite "reordenar" los contenidos del documento XML, de una forma distinta a como están ordenadas en el documento XML inicial.

## Selección de valores de los elementos <xsl:value-of...>

En el elemento **<xsl:value-of…>** se puede indicar que se quiere mostrar el valor del elemento que estamos procesando. También podemos indicar que queremos mostrar el valor de un elemento hijo, o descendiente, del elemento que se está procesando

En el ejemplo anterior, podríamos utilizar xsl:value-of para mostrar en el documento resultado de la transformación el título, código de registro o fecha de préstamo de cada libro…

Esto es posible porque en **el atributo select podemos utilizar una "expresión XPATH"**. Por ejemplo, para mostrar el valor del elemento "titulo", que es un hijo del elemento ejemplar, podríamos utilizar la siguiente regla:

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

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

En resumen, en las reglas XSLT, entre sus marcas de inicio y de fin, se puede incluir:

- Texto que se escribirá “tal cual” en el documento resultado de la transformación.
- Marcas HTML o XML que se añadirán al documento resultado de la transformación.
- Elementos reservados de la especificación XSLT que realizarán una acción como recuperar el valor de un elemento, ordenar los resultados, llamar a otras reglas de la hoja de estilo, etc.

## Selección de los valores de los atributos <xsl:value-of...>

En XSLT podemos “filtrar” o indicar qué instancias de un elemento queremos procesar, tomando como criterio de selección el valor de los atributos que acompañan a los elementos.

Para hacer esto, en un elemento **xsl:value-of**, podemos recuperar el valor de un atributo mediante la expresión **@nombreAtributo**, por ejemplo:

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

## Clasificación de valores.

Para ordenar los contenidos, se utiliza el elemento **xsl:sort** que acepta dos atributos:

**select**. Que toma como valor el nombre del elemento que se va a utilizar como criterio de ordenación.

**order**. Que indica si se debe utilizar un orden ascendente o descendente.

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

## Ejecución condicional de reglas

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

```
<xsl:if test="@destino='JFK'">
      <tr>
            <td><xsl:value-of select="@numero" /></td>
            <td><xsl:value-of select="@origen" /></td>
            <td><xsl:value-of select="@destino" /></td>
            <td><xsl:value-of select="@hora" /></td>
      </tr>
</xsl:if>
```

## Selección entre varias alternativas: xsl:choose, xsl:when y xsl:otherwise

Estos elementos "amplían" las posibilidades del elemento xsl:if. Permiten indicar qué transformación se debe realizar en el caso de que se cumpla una condición, y en el resto de casos.

**Se utilizan de forma conjunta**. El elemento **xsl:choose** contendrá a uno o más elementos **xsl:when** y a un elemento **xsl:otherwise**.

El elemento xsl:when incluye un atributo test que tomará como valor la expresión que se evaluará. Si se cumple, se ejecutará el código escrito entre las etiquetas de inicio y de fin del elemento xsl:when.

El elemento xsl:otherwise contendrá el código que se ejecutará si no se cumplen las expresiones indicadas en los atributos test de los elementos xsl:when.

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

## Repetición: xsl:for-each

Con xsl-for each se puede repetir un procesamiento para un conjunto de nodos, aunque ha que tener en cuenta que no es un bucle “for” convencional.

Aquí podemos ver un ejemplo:

```
<xsl:template match=”club”>
      <ul>
            <xsl:for-each select=”plantilla”>
                  <li><xsl:value-of select=”jugador”/></li>
            </xsl:for-each>
      </ul>
<xsl:template>
```

## Uso de variables: xsl:variable.

En XSLT se pueden declarar variables, a éstas, **cuando se le asigna un valor, éste ya no se puede cambiar**.

El elemento **xsl:variable** se utiliza para declarar una variable. **Las variables nos permiten realizar operaciones con los datos del documento XML para luego mostrar el resultado en el documento “resultado”**

Para declarar una variable, se utilizará la sintaxis:

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

A continuación, tenemos un ejemplo:

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

## Copia de contenidos

### xsl:copy-of

Se utiliza para copiar un conjunto de nodos del documento origen, al documento resultado de la transformación. Se copiarán todos los nodos hijos y los atributos (en el caso de los elementos que los tengan).

Este elemento es especialmente útil cuando se quiere convertir un documento XML a otro documento XML con una estructura diferente. El elemento **xsl:copy-of** irá acompañado por un atributo **select** que toma como valor una expresión que determinará los nodos que se van a copiar.

Este elemento también se puede utilizar para copiar en el documento resultado el valor de una variable. En este caso, se escribirá como valor del atributo **select** el nombre de la variable precedido por el carácter **$**.

Ejemplo, para el fichero xml:

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

fichero xsl:

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

fichero resultado:

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

## xsl:copy

Similar al elemento anterior, se utiliza para copiar elementos, pero no se copiarán sus atributos ni sus elementos hijos.

Cuando se aplica sobre elementos, se copia el elemento, pero no su valor.

Ejemplo:

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

En este ejemplo, se crea un elemento autor vacío en el documento destino, para cada elemento autor existente en el documento original

Para copiar el valor de los elementos autor, habría que modificar la XSLT

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

## Creación de elementos: xsl:element

Se utiliza para crear elementos en el documento resultado de la transformación. Es especialmente útil cuando se utiliza XSLT para transformar un documento XML en otro con una estructura diferente.

**xsl:element** irá acompañado por un atributo "name" que tomará como valor el nombre del elemento que se va a crear. Si el elemento procede de un espacio de nombres, el URI que corresponde a este espacio de nombres se puede indicar en otro atributo: **namespace.**

```
<xsl:template match="libro">
      <xsl:element name="h1">
            <xsl:value-of select="titulo" />
      </xsl:element>
</xsl:template>
```

## Creación de atributos: xsl:attribute

**xsl:attribute** permite crear un atributo en el documento resultado de la transformación. Irá acompañado por un atributo "**name"**, que recogerá el nombre del atributo, y opcionalmente por un atributo "**namespace"** que recogerá el alias del espacio de nombres del cual procede el atributo.

```
<xsl:template match="enlace">
      <xsl:element name="a">
            <xsl:attribute name="href">
                  <xsl:value-of select="@url" />
            </xsl:attribute>
            <xsl:value-of select="." />
      </xsl:element>
</xsl:template>
```

## Conjuntos de atributos: xsl:attribute-set

Mediante **xsl:attribute-set** pueden definirse conjuntos de atributos para su reutilización, por ejemplo:

```
<xsl:attribute-set name=”img-grande”>
      <xsl:attribute name=”width”>250px</xsl:attribute>
      <xsl:attribute name=”height”>250px</xsl:attribute>
</xsl:attribute-set>


<xsl:attribute-set name=”img-peque”>
      <xsl:attribute name=”width”>50px</xsl:attribute>
      <xsl:attribute name=”height”>50px</xsl:attribute>
</xsl:attribute-set>

<xsl:template match=”imagen”>
      <img xsl:use-attribute-sets=”img-peque” src=”{@src}” />
</xsl:template>
```

Luego se pueden usar con **xsl:use-attribute-sets** como se ve en el ejemplo.

## Inclusión de texto: xsl:text

Con xsl:text se puede incluir texto en el archivo resultante:

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

## Inclusión de comentarios: xsl:comment

Este elemento se utilizará para crear un comentario en el documento resultado de la transformación.

El elemento **xsl:comment** contendrá el texto del comentario, **sin las marcas <!--** y **-->**

```
<xsl:template match="vuelo">
      <tr>
            <td><xsl:value-of select="@numero" /></td>
            <xsl:comment>Aquí se va a añadir texto</xsl:comment>
            <td>
                  <xsl:text>Aeropuerto:</xsl:text>
                  <xsl:value-of select="@origen" />
            </td>
            <td>
                  <xsl:text>Aeropuerto:</xsl:text>
                  <xsl:value-of select="@destino" />
            </td>
            <td><xsl:value-of select="@hora" /></td>
      </tr>
</xsl:template>
```

## Añadir instrucciones de procesamiento: xsl:processing-instruction

Se utiliza para crear una instrucción de procesamiento en el documento resultado de la transformación. Debe ir acompañado por un atributo **"name"**, que es obligatorio, y que toma como valor el nombre de la instrucción de procesamiento.

Entre sus etiquetas de inicio y de fin se escribirán los calificadores de la instrucción de procesamiento, entre las marcas **<xsl:text>** y **</xsl:text>.**

Por ejemplo, el siguiente código crearía una instrucción de procesamiento en el documento destino:

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

## Procesador de hojas de estilo XSLT

Para probar hojas de estilo XSLT se puede usar el procesador en línea freeformatter que puedes encontrar en:

https://www.freeformatter.com/xsl-transformer.html

Su uso es muy sencillo y se puede ver en la captura de imagen siguiente: