UNIDAD 5 CONVERSIÓN Y ADAPTACIÓN DE DOCUMENTOS XML
PARTE I XPATH

1 INTRODUCCIÓN

A pesar de que el XML es un formato relativamente legible al ser visualizado, este no es uno de sus objetivos principales, de hecho en la especificación XML no se incluye información sobre la visualización en los documentos.

En determinados casos puede ser necesario transformar los documentos XML para que sean más fáciles de visualizar, o también adaptarlos para que puedan ser leídos por programas específicos.

XML está pensado sobre todo para **almacenar e intercambiar información**, de manera que si hay que representar los datos de una manera diferente para optimizar un proceso o para mejorar la visualización habrá varias posibilidades:

1. **Desarrollar un programa**: al ser relativamente sencillo trabajar con XML, se podría desarrollar un programa que tome los datos XML y genere la salida tal como la queremos. Esto tiene el inconveniente de que habrá que tener conocimientos de programación y que puede representar mucho trabajo aunque lo que haya que hacer sea trivial.
2. **Utilizar CSS**: en muchos casos una solución sencilla sería utilizar CSS para representar la información de manera más amigable usando un navegador. Pero sólo sirve para cambiar la visualización, no para cambiar el documento.
3. **Transformar el documento** : una solución alternativa consiste en transformar el documento en otro que esté pensado para ser visualizado.

2 CSS

En muchos casos si el objetivo es que el documento sea visualizado de una manera más agradable la solución más sencilla puede ser visualizarlo mediante CSS.

CSS resulta útil si se quieren hacer adaptaciones sencillas. Por ejemplo:

```
<? xml version="1.0" ?>
<empleado>
      <nombre>Marcelino</nombre>
      <apellido>Garcia</apellido>
      <departamento>Departamento de Informática </departamento>
      <tareas>
            <tarea>Mantener la base de datos de la empresa</tarea>
            <tarea>Administrador del servidor web</tarea>
      </tareas>
</empleado>
```

y se quiere representar como un informe de sus tareas se puede usar un documento CSS que contenga estas líneas:

```
empleado{
      padding:30px;
      margin:30px;
      border:4px black solid;
      width:40%;
}
```

```
nombre,apellido{
      font-size:30px;
}
departamento{
      padding-top:20px;
      display:block;
      font-weight:bold;
}

tarea{
      font-style:italic;
      padding-left:10px;
}
```

Para enlazar un documento XML con un fichero CSS habría que añadir en el fichero XML la siguiente línea:

```
<?xml-stylesheet href="empresa.css"?>
```

A pesar de ello, CSS tiene muchas limitaciones a la hora de presentar la información:

La información no puede ser reordenada con queramos. La única manera de simular el cambio de orden es usar posicionamientos absolutos.

1. Los atributos se pueden mostrar pero hay muchas limitaciones para hacerlo.
2. No se pueden añadir estructuras nuevas producto de cálculos de los datos del documento.
3. No tiene maneras sencillas de formatear los datos en páginas de texto para ser impresas.

Si el objetivo final no es simplemente decorar el archivo sino transformarlo en otro documento totalmente diferente, CSS no sirve. CSS no transforma el documento sino que simplemente cambia la forma en que se visualiza.

3 TRANSFORMACIÓN DE DOCUMENTOS

Para intentar conseguir hacer todo lo que CSS no podía hacer se creó un nuevo lenguaje de plantillas: XSL (extensible stylesheet language).

Inicialmente se concentraron poder representar la información de manera que pudiera ser mostrada en documentos impresos y en pantalla pero luego se terminó definiendo un sistema para hacer transformaciones genéricas de documentos XML en otras cosas: documentos de texto, documentos XML, páginas web, etc.

La familia XSL está formada por tres lenguajes:

- **XSL-FO** (XSL formatting objects): un lenguaje para definir el formato que se aplicará a un documento.
- **XSLT** (XSL transformations): un lenguaje para transformar documentos XML.
- **XPath**: un lenguaje para acceder a partes de los documentos XML.

4 XPATH

4.1 Introducción

XPath se usa para navegar por los elementos y atributos en un documento XML.

¿Qué es XPath?

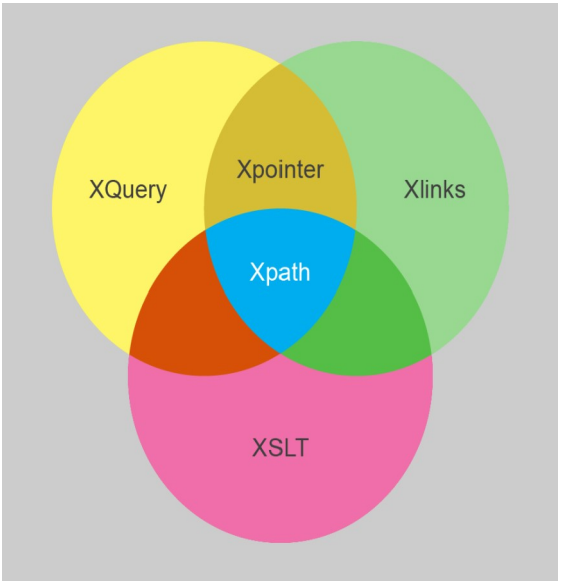
- XPath es una sintaxis para definir partes de un documento XML.
- XPath usa expresiones de ruta para navegar en documento XML.
- XPath contiene una librería de funciones estándar.
- XPath es un componente de XSLT
- XPath es una recomendación de W3C

XPath no es un lenguaje XML, de manera que se podría incluir en otros lenguajes XML sin tener que preocuparse si el resultado está bien formado o no.

La base del funcionamiento de XPath es la evaluación de expresiones. Una expresión que se evaluará contra un documento XML y nos dará un resultado que puede ser de diferentes tipos:

1. Un booleano: cierto o falso
2. Un número
3. Una cadena de caracteres
4. Un grupo de elementos

XPath está desarrollado por los comités de creación de XSL y XQuery y se ha convertido en un componente esencial para diferentes lenguajes XML como XLink, XSLT y XQuery.



La versión 2.0 de XPath está tan integrada dentro de Query que cualquier expresión XPath es también automáticamente una expresión XQuery correcta. Actualmente, la versión es la 3.0 y data de 2014.

4.2 Ejemplo para trabajar

El ejemplo con el que vamos a trabajar durante esta unidad es el archivo pelicula.xml, que podéis encontrar en el moodle.

Con XPath vamos a conseguir contestar a preguntas como éstas:

- ¿Cuál es el título de esta película?
- ¿Cuántos productores conocemos?
- ¿Cuáles son sus nombres completos?
- ¿Cuántos actores tiene?

4.3 Nodos

XPath trata a los documentos XML desde el punto de vista de un árbol de nodos en los que hay una raíz que no se corresponde con la raíz del documento, sino que es el documento y se representa con el símbolo "/".

Para XPath, todos los elementos que componen un documento XML son nodos, así existen diferentes clases de nodos:

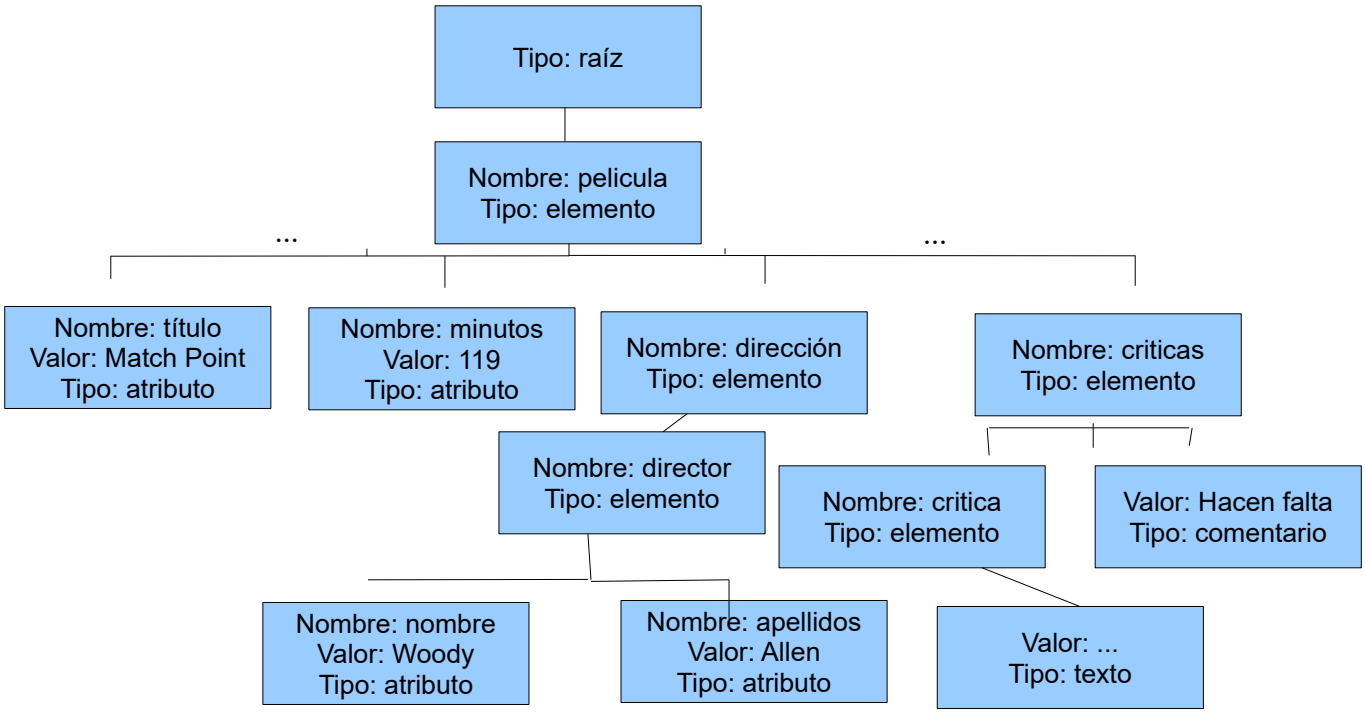
- **Raíz**: El elemento principal (o raíz) del documento XML
- **Instrucciones de proceso**: directivas que permiten definir aspectos del documento, así como declararlo como documento XML.

  Elementos: los bloques sobre los que se apoya XML.
- **Atributos**: propiedades de los elementos.
- **Texto**: los valores incluidos en el interior de los elementos, entre etiquetas de apertura y cierre.
- **Comentarios**: también podemos acceder al contenido de todos los comentarios.

Si tenemos el siguiente documento XML:

```
<pelicula título="Match Point" minutos="119">
      <dirección>
            <director nombre="Woody" apellidos="Allen" />
      </dirección>
      <criticas>
            <critica>Buena</critica>
            <!-- Hacen falta -->
      </criticas>
</pelicula>
```

De esta manera, para XPath, el documento XML tendría un aspecto similar al que puede verse en la siguiente figura:



4.4 Consultas

Las consultas realizadas con XPath pueden devolver valores de diferentes tipos:

- **Listas de nodos**: si intentamos obtener por ejemplo, la lista de intérpretes de la película.
- **Valores booleanos**: si intentamos saber el documento XML contiene alguna de las críticas que han puesto.
- **Cadenas**: si intentamos obtener el nombre completo (nombre y apellidos) del director de la película.
- **Números**: si intentamos obtener el número de intérpretes de los que se tiene noticia.

Expresar caminos en XPath se parece tanto a como lo hacen los sistemas operativos que incluso se utilizan símbolos similares.

Lo más importarte para tener en cuenta a la hora de crear una expresión XPath es saber el nodo en el que se está situado, ya que es desde este que se evaluar´la expresión. Así, podemos expresar las consultas XPath de dos maneras.

- **Ruta relativa**, que parten del nodo en el que estamos situados.

  `criticas/critica` ,a partir de de `/pelicula`

- **Ruta absoluta**, son caminos que empiezan siempre desde la raíz del árbol.

  `/pelicula/criticas/critica`

4.5 Relaciones entre los nodos

Los nodos en XML están relacionados unos con otros. Así podemos encontrar los siguientes tipos de nodos:

Padre (Parent)

Un nodo es padre de otro cuando éste desciende de él. Cada elemento y atributo tiene un único padre.

En el siguiente ejemplo, el elemento libro es el padre de titulo, autor, año y precio.

```
<libro>
  <titulo>Riña de gatos</titulo>
  <autor>Eduardo Mendoza</autor>
  <año>2010</año>
  <precio>20.50</precio>
</libro>
```

Hijo (Children)

El hijo es el nodo que desciende del nodo padre. Los elementos nodos pueden tener cero, uno o más hijos.

En el ejemplo anterior, los elementos título, autor, año y precio son todos hijos del elemento libro.

Hermanos (Siblings)

Nodos que tienen el mismo padre.

En el ejemplo anterior los elementos titulo, autor, año y precio son todos hermanos.

Antepasados (Ancestors)

El padre del padre del padre … de un nodo.

En ele ejemplo siguiente los antepasados del elemento titulo son los elementos libro y el elemento librería.

```
<libreria>
<libro>
  <titulo>Riña de gatos</titulo>
  <autor>Eduardo Mendoza</autor>
  <año>2010</año>
  <precio>20.50</precio>
</libro>
</libreria>
```

Descendientes (Descendants)

El hijo de un hijo de un hijo … de un nodo.

En el ejemplo anterior descendientes del elemento librería son los elementos libro, titulo, autor, año y precio.

4.6 Nodo raíz

Obtener el nodo raíz es el proceso más sencillo, solo hemos de preguntar por la barra inclinada, operación que nos devolverá todo el documento XML. La / selecciona todo el documento.

4.7 Elementos

Para seleccionar un determinado elemento sólo es necesario indicar su ruta. El elemento más sencillo es el nodo raíz, sólo necesitamos escribir / para acceder a él.

En ele ejemplo, un elemento puede ser el nodo director. así su ruta es:

```
/pelicula/direccion/director
```

Una consulta también puede devolver un conjunto de nodos. Por ejemplo:

```
/pelicula/reparto/interprete
```

devuelve todos los nodos interprete que aparezcan en el documento.

4.8 Atributos

Para mostrar el valor de los atributos de un nodo hemos de acceder al elemento que nos interesa. Así, para acceder al nombre del director, la consulta es:

```
/pelicula/direccion/director/@nombre
```

La @ diferencia entre elementos y atributos.

4.9 Texto

Para obtener el texto de un determinado elemento podemos utilizar text(). Así, para obtener el texto de una crítica la consulta es:

```
/pelicula/criticas/critica/text()
```

4.10 Comentarios

Para obtener el valor de un determinado comentario hay que utilizar comment(). Para obtener el comentario del documento es:

```
/pelicula/criticas/comment()
```

4.11 Funciones

Además de acceder a los valores incluidos en un determinado documento XML, también podemos realizar operaciones a partir de elementos, como por ejemplo, concatenar varias cadenas o contar elementos.

Ejemplo: ¿Cuántos intérpretes tenemos en la película?

```
count (/pelicula/reparto/interprete)
```

Ejemplo: Concatenamos el nombre y apellidos de todos los actores. Podríamos mostrar primero el

apellido y luego una coma, y después el nombre.

```
concat (/pelicula/reparto/interprete/@apellidos, ', ', /pelicula/reparto/interprete/@nombre)
```

Xpath tiene una lista de unas 200 funciones, para ver una lista completa podéis consultar el siguiente enlace: https://www.w3.org/TR/xpath-functions-31

Veamos las funciones más importantes que podemos usar:

Funciones Booleanas

`boolean()`	Toma un objeto como argumento y devuelve un valor booleano. • Si el argumento es un número, devuelve true si el número no es cero. • Si el argumento es un conjunto de nodos, devuelve true si el conjunto de nodos no está vacío. • Si el argumento es un string, devuelve true si la cadena es no vacía.
`false()`	No lleva argumentos y devuelve false
`not()`	Lleva como argumento una expresión booleana y devuelve true si la expresión evaluada es false, y false si la expresión es true.
`true()`	No lleva argumentos y devuelve false

Funciones de conjuntos de nodos

`count()`	Toma como argumento un conjunto de nodos y devuelve el valor correspondiente a l número de nodos del conjunto de nodos.
`id()`	Toma como argumento un string y devuelve un conjunto de nodos conteniendo cualquier nodo que tenga un atributo del tipo ID igual al valor del argumento.
`last()`	Devuelve un valor igual al último nodo del conjunto.
`position()`	Devuelve un número correspondiente con la posición actual.

Funciones numéricas

`ceiling()`	Toma un número como argumento y devuelve el número entero más pequeño que es mayor que el argumento.
`floor()`	Toma un número como argumento y devuelve el número entero más alto que es menor que el argumento.
`number()`	Toma como argumento un string, booleano o un conjunto de nodos y: • Si es un string y contiene caracteres que son números, devuelve esos números. • Si es el booleando true devuelve 1, y si es el booleano false devuelve 0.
`round()`	Redondea un número al entero más próximo.
`sum()`	Toma como argumentos un conjunto de nodos y devuelve la suma del valor de

	cada nodo individual después de convertir el valor a numérico si es posible.

Funciones de cadena

`concat()`	Toma como argumentos dos o más cadenas y devuelve la concatenación de estas cadenas.
`contains()`	Toma como argumentos dos strings y devuelve true si la primera cadena contiene a la segunda.
`normalize-space()`	Toma una cadena como argumento. Los espacios son convertidos a espacios simples y después eliminados.
`starts-with()`	Toma como argumentos dos strings y devuelve true si el primer argumento empieza con el segundo argumento.
`string()`	Toma un booleano, un conjunto de nodos o un número y devuelve un valor string.
`string-length()`	Toma como argumento un string y devuelve la longitud del mismo.

4.12 Sintaxis de XPath

XPath usa expresiones de ruta para acceder a un nodo o a un conjunto de nodos. XPath usa expresiones de ruta para seleccionar nodos. El nodo es seleccionado mediante una serie de rutas o trayectorias.

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 es la que vamos a ver
- 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.

Hay que tener cuidado en escribir las expresiones de XPath, ya que si el camino especificado no se corresponde con un camino real dentro del árbol no se devolverá ningún resultado.

Las expresiones de ruta más usadas son las siguientes:

Operador path	Descripción
nombre_nodo	Selecciona todos los nodos hijos del nodo nombrado

/	Selecciona desde el nodo raíz
//	Selecciona nodos en el documento desde el nodo actual, no importando donde están. “recursive descendant”
.	Selecciona el nodo actual
..	Selecciona el padre del nodo actual
@	Selecciona atributos
*	“wildcard” Comodín que indica todo

Veamos con el siguiente ejemplo cómo se seleccionan los nodos con los operadores anteriores.

```
<empleados>
    <empleado ID=”1234”>
     <nombre>Antonio</nombre>
     <apellido>Pérez</apellido>
    </empleado>
</empleados>
```

Ejemplo de operador PATH	Explicación
empleados	Selecciona todos los nodos hijos del elemento empleados
/empleados	Selecciona el elemento raíz empleado
empleados/empleado	Selecciona todos los elementos empleado que son hijos de empleados
//empleado	Selecciona todos los elementos empleados no importando donde se encuentren en el documento.
empleados//nombre	Selecciona el elemento nombre de cualquier nivel por debajo del elemento empleados
//@ID	Selecciona todos los atributos nombrados como ID
empleado/*	Selecciona todo debajo de empleado
empleado/@*	Selecciona todos los atributos que hay por debajo de empleado

Usando el operador | en una expresión XPath podemos seleccionar varias rutas. La tabla siguiente muestra algunos ejemplos de esto:

Expresión Path	Resultado
//empleado/nombre | //empleado/apellido	Selecciona todos los elementos nombre y los elementos precio de todos los elementos empleado.
//nombre | //apellido	Selecciona todos los elementos nombre y todos los elementos precio en el documento.

4.13 XPath Axes

Un axis o eje define un conjunto de nodos en relación con el nodo actual.

Nombre Axis	Resultado
ancestor	Selecciona todos los antepasados (ancestors) del nodo actual.
ancestor-or-self	Selecciona todos los antepasados (ancestors) del nodo actual. Y el nodo actual.
attribute	Selecciona todos los atributos del nodo actual.
child	Selecciona todos los hijos del nodo actual.
descendant	Selecciona todos los descendientes del nodo actual.
descendant-or-self	Selecciona todos los descendientes del nodo actual y el nodo en si mismo.
following	Selecciona todo en el documento después de la etiqueta de cierre del nodo actual.
following-sibling	Selecciona todos los hermanos después del nodo actual.
parent	Selecciona el padre del nodo actual.
preceding	Selecciona todo en el documento que esté antes de la etiqueta de apertura del nodo actual.
preceding-sibling	Selecciona todos los hermanos antes del nodo actual.
self	Selecciona el nodo actual.

Una trayectoria de localización (location path) consiste en:

- Un axis
- un node-test
- un predicado (opcional)

axis::node-test[predicate]

Veamos algunos ejemplos:

Ejemplo	Resultado
child::empleado	Seleccionamos todos los nodos empleados que son hijos del nodo actual.
attribute::ID	Selecciona el atributo ID del nodo actual.
child::*	Selecciona todos los hijos del nodo actual.
atribute::*	Selecciona todos los atributos del nodo actual.
child::text()	Selecciona todos los nodos de tipo texto del nodo actual.
child::node()	Selecciona todos los nodos hijos del nodo actual.
descendant:: empleado	Selecciona todos los descendientes de empleados del nodo actual.
ancestor::empleado	Selecciona todos loa antepasados de empleado del nodo actual.
ancestor-or-self::empleado	Selecciona todos los antepasados de empleado el nodo actual y el nodo actual si es un nodo empleado.
child::*/child::apellido	Selecciona todos los nietos apellidos del nodo actual.

Algunos de estos ejes ya no se usan, porque es más cómodo y corto definir las expresiones a partir del símbolo.

4.14 Filtros o predicados

Podemos aplicar también filtros o predicados en la consulta de XPath.

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.

Los predicados pueden contener diferentes operadores

- operadores lógicos: and, or, not()
- operadores relacionales: = != > >= < <=
- operadores aritméticos: +, -, *, div, mod

Veamos varios ejemplos:

```
/pelicula/produccion[productor/@nombre='Stephen']
/pelicula[(@minutos <(100*0.05))]/produccion
```

Veamos más ejemplos:

Presencia de elementos hijos:

```
empleado[salario]/nombre
```

Nos situamos en el elemento nombre, por debajo de empleado que tengan elemento salario. Si tenemos un elemento empleado que no tiene salario no lo selecciona.

Valor de los elementos hijos:

`empleado[salario>2500]`

Nos situamos en el elemento empleado cuyo salario sea >2500.

Presencia de atributos:

`empleado[@ID]`

Selecciona los elementos empleado que tengan un atributo ID.

Valor de atributos

`empleado[@ID=”1234”]`

Buscamos el elemento por el valor del atributo.

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

```
/empleados/empleado[1]
```

4.15 Pasos de búsquedas consecutivos

En el siguiente ejemplo:

```
<?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>
```

En el ejemplo siguiente se obtienen los títulos de los libros publicados después de 1970, mediante dos pasos de búsqueda: `//fechaPublicacion[@año>1970]/../titulo`

- 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>.

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"]
```

- mediante dos pasos de búsqueda.

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

  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>).

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

- `//@año[.=1969]/../../autor`
- `//libro[fechaPublicacion/@año=1969]/autor`
- `//fechaPublicacion[@año=1969]/../autor`
- `//autor[../fechaPublicacion/@año=1969]`

4.16 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:

- 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
```

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

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

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

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

Otro ejemplo de expresión anidada sería obtener los títulos de los libros del mismo autor que la novela "Pantaleón y las visitadoras". Como en el ejemplo anterior, la respuesta puede obtenerse en

dos pasos:

- obtener primero el autor de la novela "Pantaleón y las visitadoras":
  `//libro[titulo="Pantaleón y las visitadoras"]/autor/text()`
- y obtener después los títulos de los libros escritos por Mario Vargas LLosa:
  `//libro[autor="Mario Vargas Llosa"]/titulo`

Estas dos expresiones se pueden unir en una única expresión, sustituyendo en la segunda expresión el valor "Mario Vargas Llosa" por la primera expresión:

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

Un detalle importante es que no hay que escribir la primera expresión entre comillas.
Incluso se puede omitir la selección de nodos /text() de la segunda expresión y escribir la expresión XPath así:

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

5 BIBLIOGRAFÍA

- W3School
- Documentación en Internet