Apuntes básicos del Tema 8

Introducción a XPATH

Contenidos:

8.0- Introducción y definición

Lo que hemos aprendido hasta ahora con XML y sus tecnologías asociadas nos dejan claro que se trata de un sistema de almacenamiento de información fiable y ordenada, de forma que los distintos documentos que siguen un patrón determinado tendrán un contenido homogéneo que garantice la compatibilidad de la información. Para ello hemos dispuesto de herramientas y técnicas que validaban los documentos creados, mediante DTD o esquemas, que filtraban cualquier error de contenido que no se adaptase a la estructura establecida.

Pero la clave que justifica utilizar XML no se queda solo en almacenar, sino también en *extraer* esa información que, si el documento ha sido comprobado con la correspondiente validación, ya

sabemos que existirá y será posible localizar sus componentes de forma sencilla y adecuada.

El consorcio W3C fue consciente de ello y poco después de publicar las especificaciones XML comenzó a emitir las primeras recomendaciones XPath.

Se usa el término **Path** ya que la estructura está inspirada en el conocido ***File Path*** o ruta de acceso usada en el S.O. para carpetas y archivos (veremos posteriormente que la barra "/" inicia el camino absoluto que luego le seguirá los hijos y descendientes). Sin embargo se ha de tener clara desde el principio la diferencia consistente en que en una ruta de ficheros se devolvería el propio (y único) fichero final, mientras que en la estructura de Xpath se devuelve una referencia a **TODOS** los elementos que cumplan la expresión .

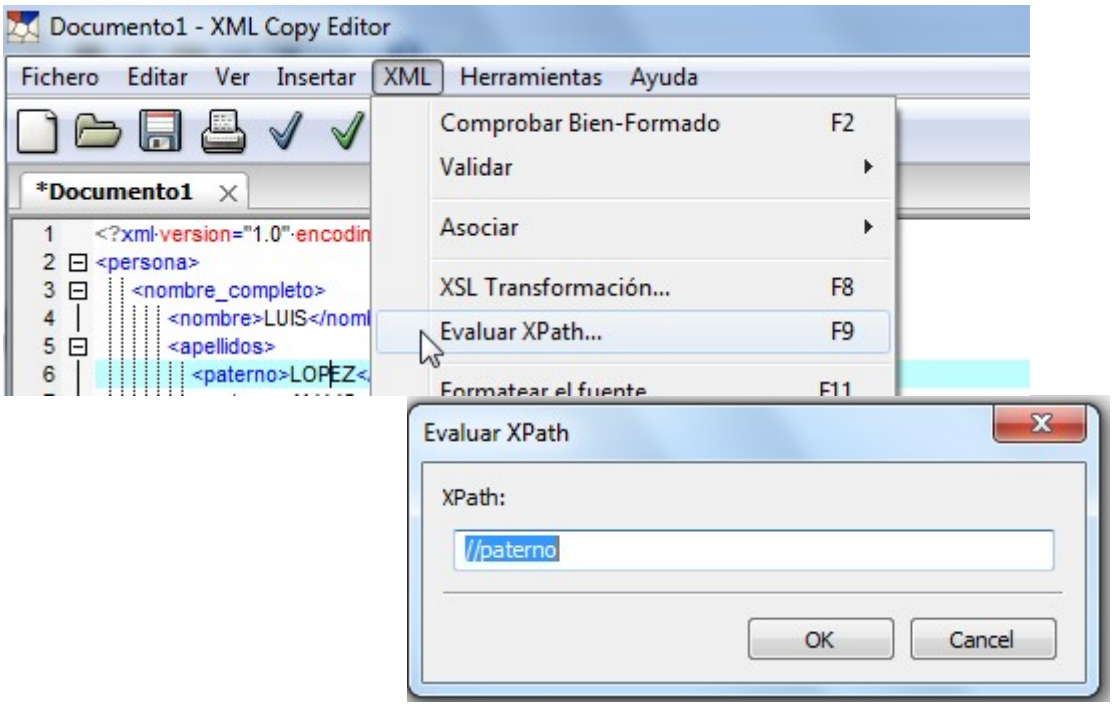
XML Path, que utilizamos por su abreviado **Xpath**, es un lenguaje **declarativo** que se basa en la utilización de unas determinadas expresiones que forman un conjunto de reglas sintácticas y que permiten hacer búsquedas y seleccionar partes del documento XML, apoyándose para ello en una estructura jerárquica que interpreta el contenido como un *árbol de nodos*.

Al decir que XPath es un lenguaje **declarativo** nos referimos a que no se trata de los conocidos lenguajes procedurales (C, Java, Pascal, etc.,) que usan la programación **imperativa**, es decir, necesitan un algoritmo para describir los pasos que debe seguir el programa para alcanzar su resolución. Por el contrario, los lenguajes declarativos como XPath solo describen el problema, (por ejemplo, decir lo que quieren encontrar) y si la sintaxis es la adecuada, el sistema ya tiene los mecanismos internos para dar la respuesta correspondiente.

Hemos de admitir que XPath es un lenguaje sofisticado y complejo, agravado por la circunstancia de que al igual que el resto de tecnologías XML esta en un estado de relativo desarrollo, que dificulta su uso en todas las funcionalidades teóricamente previstas. Por esa razón **en este tema** y teniendo en cuenta las limitaciones temporales del curso se expondrá una visión general teórica del contenido, aunque las prácticas y ejemplos tendrán **una aplicación simplificada** de sus inmensas posibilidades.

Además, hemos de entender que el verdadero significado de la aplicación de XPath está en su **utilización conjunta con otras tecnologías, como es XSLT que veremos en los próximos temas**, y que se ocuparán de hacer transformaciones sobre los documentos XML para convertirlos en otros documentos (bien sean otros documentos XML, o documentos XHTML preparados para usar plenamente las hojas de estilo). Para usarlo en esas trasformaciones XPath no necesita herramientas auxiliares, pero para practicar las búsquedas de forma aislada (que es lo que haremos en este tema como paso previo para entender su base practicando y a modo de ensayo) es

necesario un programa que nos muestre el resultado de la evaluación de las expresiones que aplicamos, y para ello seguiremos disponiendo del XML CopyEditor, que aunque **es muy básico** nos mostrará los nodos que hayan sido seleccionados.



8.1 XML en forma de árbol.

Para comprender el funcionamiento de XPath debemos ver siempre el documento XML como un árbol de nodos, que se conoce como *estructura jerárquica en árbol*.

Entendemos como **nodo** cualquier parte del documento XML, pudiendo ser uno de los siguientes:

- **Raíz**
- **Elementos**
- **Texto**
- **Atributos**
- **Instrucciones de proceso**
- **Espacios de nombres**
- **Comentarios**

Aunque las expresiones XPath procesan principalmente nodos de elementos y atributos.

El elemento raíz (representado como **/**) no es lo mismo que la raíz del documento, ya que el primero puede contener además del elemento principal el prólogo o instrucción de proceso. Siempre tendremos un elemento raíz que está por encima y contiene todo el documento. De esta

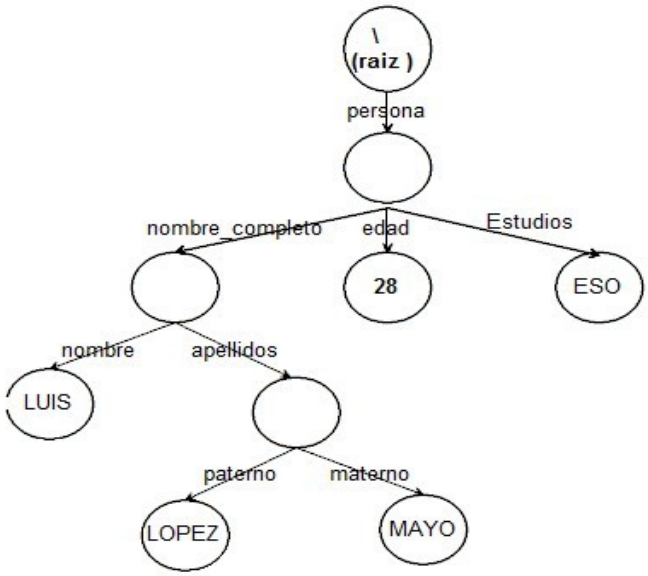
forma, hemos de entender que es simplemente la raíz conceptual del documento.

Podemos imaginarnos una representación gráfica de cualquier documento mediante un gráfico que asocie su estructura con esa estructura de árbol, como en el siguiente ejemplo sencillo sobre la identificación de una persona:

Si tenemos el siguiente documento básico XML:

```
<persona>
    <nombre_completo>
        <nombre>LUIS</nombre>
        <apellidos>
            <paterno>LOPEZ</paterno>
            <materno>MAYO</materno>
        </apellidos>
    </nombre_completo>
    <edad>28</edad>
    <estudios>ESO</estudios>
</persona>
```

Podemos visualizarlo con una estructura de árbol tal y como lo entendería XPath similar a:



No existe una normalización sobre la forma de presentar el contenido del árbol. En este caso se ha reflejado el nombre de los elementos en la línea que conecta a los nodos, y su valor dentro de él. Podemos ver así claramente la estructura, que los valores no tienen descendientes (se para allí el arbol), y cómo están anidados.

Podría haber sido alternativamente representado por cajas o rectángulos, o poner el nombre del elemento en los círculos y su valor mediante otro sistema. Lo importante es que relacionemos gráficamente la estructura del documento con esa representación arbórea.

Así podremos también tener en cuenta el orden del documento (además de su estructura) para ver cuando las expresiones nos devuelven los elementos en el mismo orden en que se encuentran en el documento. Aunque habrá que tener presente que sin embargo los atributos no tienen orden preestablecido.

8.2 Trayecto de búsqueda

La relación que existe entre los distintos niveles del documento puede ser muy simple, como la que se desprende de padres a hijos, pero la evaluación de XPath tendrá en cuenta el ***eje*** sobre el que se mueve, es decir la dirección en la que se debe evaluar la expresión, moviéndose en el árbol hacía arriba , buscando padres(parent) y otros antecesores (ancestor) o hacia abajo, buscando hijos (child) o descendientes(descendant).

Además las rutas pueden ser **absolutas** o **relativas.** Una ruta **absoluta** comienza siempre con el símbolo **"/"** ya que hace referencia al nodo raíz, y va seguida de la lista de los elementos **hijo,** separados en cada nivel por la **"/"**, formando el camino de la propia expresión de búsqueda, mientras que una ruta **relativa** tiene en cuenta la posición en la que se encuentra en ese momento, es decir, el **nodo de contexto**.

Las limitaciones impuestas por la sencillez de la herramienta que utilizamos para las pruebas en este tema nos llevarán a probar solamente expresiones que partan **de rutas absolutas**. Recordemos que el uso completo y adecuado de XPath lo pondremos en práctica embebiéndolo en las transformaciones que realicemos en los temas siguientes dentro de XSALT y que no necesitarán herramientas auxiliares

Los elementos que forman parte de un trayecto de búsqueda se suelen clasificar en tres tipos

-Ejes
-Nodos
-Predicados

y aunque la sintaxis formal sigue el modelo:

```
eje :: nodo [predicado]
```

nosotros simplificaremos siempre que sea posible las expresiones por lo que el primer paso de la sintaxis (*eje::*) será sustituido por la abreviatura que en cada caso corresponda.

No vamos a profundizar en todos los detalles y particularidades de la sintaxis de XPath, algunas de cuyas características se comprobarán en el tema siguiente, pero vamos a mostrar especialmente algunos pasos y abreviaciones que son de uso más común. Resaltamos ya algunos conceptos básicos que en algunos casos se volverán a comentar en su apartado correspondiente, y para ello recurriremos a nuestro conocido ejemplo de peliculas (de momento con una sola), utilizado en los temas anteriores:

```
<?xml version="1.0" encoding="UTF-8"?>
<filmoteca>
   <pelicula
              estreno="1942"  minutos="102 ">
        <titulo>Casablanca </titulo>
        <director>Michael Curtiz</director>
        <guion>
            <guionista>Julius J. Epstein </guionista>
            <guionista>Philip G. Epstein</guionista>
            <guionista>Howard Koch </guionista>
        </guion>
        <reparto>
           <interprete papel="protagonista">Humphrey Bogart</interprete>
           <interprete papel="protagonista"> Ingrid Bergman</interprete>
           <interprete papel="secundario">Paul Henreid</interprete>
           <interprete papel="secundario">Claude Rains</interprete>
           <interprete papel="secundario">Conrad Veidt</interprete>
           <interprete papel="secundario">Curt Bois</interprete>
        </reparto>
        <esloganes>
            <eslogan> El mito hecho celuloide</eslogan>
            <eslogan>La obra maestra absoluta</eslogan>
            <eslogan>Puede que nunca se haga una película mejor</eslogan>
        </esloganes>
   </pelicula>
   <!-- otras peliculas  -->
</filmoteca>
```

(peliculas1.xml)

- Una ruta estará formada por distintos pasos separados con **"/".**

  ejemplo 1: /filmoteca/pelicula/titulo

  nos devuelve como resultado :<titulo>Casablanca </titulo>

  ejemplo 2:/filmoteca/pelicula/guion/guionista

  nos devuelve como resultado:

  ```
  <guionista>Julius J. Epstein </guionista>
  <guionista>Philip G. Epstein</guionista>
  <guionista>Howard Koch </guionista>
  ```

- El nodo contexto (**Self)** es en el que nos encontramos en un momento determinado. Se representa abreviadamente con un punto (**.**) pero su utilización tiene sentido en rutas relativas

El ejemplo 1 anterior puede expresarse como /filmoteca/pelicula/titulo/**.** aportando el mismo resultado, por lo que no suele utilizarse en las rutas absolutas.

- Se puede seleccionar cualquier nodo que sea *descendiente* sin necesidad de describir todo el camino, usando "//" (que como repetiremos luego, es la abreviatura de eje "/descendent::") De esta forma se pueden seleccionar todos los elementos del documento que cumplan las condiciones pero sin tener en cuenta la ruta anterior.

  Ejemplo://guionista es equivalente y da el mismo resultado que el ejemplo anterior filmoteca/pelicula/guion/guionista

- El operador " | " puede utilizarse para seleccionar varios recorridos alternativos.

  Ejemplo:/filmoteca/pelicula/titulo |/filmoteca/pelicula/director

  Tendría como resultado:

  ```
  <titulo>Casablanca </titulo>
  <director>Michael Curtiz</director>
  ```

  y sería equivalente a la expresión //titulo |//director

- Podemos seleccionar atributos mediante @

  Ejemplo:/filmoteca/pelicula/@estreno

  da como resultado:

  estreno="1942"

8.3 EJES

Los ejes se refieren a dirección en la que nos podemos mover dentro del documento, especialmente cuando estamos en una ruta relativa (Aunque recordemos que esas rutas relativas NO los utilizaremos inicialmente en las pruebas de los ejemplos de este tema, pero pueden servirnos para temas posteriores dónde utilizaremos XPath en el contexto de otras tecnologías):. Ya nos hemos referido a ellos en algún caso al tratar la sintaxis simplificada, y a modo de ampliación (aunque la mayoría se representarán mediante sus alternativas de abreviatura más simplificadas) los siguientes son algunos seleccionados:

child::	Es el eje por defecto, ya que se refiere al contenido del nodo actual, por lo que **NO SE USA**
descendant::	Cualquier nodo descendiente del nodo contexto. Lo utilizamos **abreviadamente como //**
self::	Nodo actual Se representa **abreviadamente** como un punto (**.**)
attribute::	Sirve para localizar los atributos de un nodo, y se representa **abreviadamente** como **@**
parent::	Es el padre del nodo actual Se identifica **abreviadamente** mediante los dos puntos (**..**)
ancestor::	Localiza todos los antecesores del del nodo actual, es decir, el padre, el padre del padre, etc., hasta el nodo raíz. No tiene forma abreviada.

8.4 Nodos de comprobación

Se les llama también *filtros* o *nodo test* y se encargan de hacer la identificación de la selección en un eje.

Los más útiles pueden ser:

*	Conocido como *wilcard* según la jerga del tenis. Sirve para seleccionar el nodo teniendo en cuenta su nivel en el arbol. Por ejemplo: /*/*/*/eslogan Obtendría los elementos eslogan que tienen tres antecesores (están en cuarto nivel) , mientras que /*/eslogan no devolvería ningún elemento ya que no se encuentra en el segundo nivel. También sirve para seleccionar todos los elementos hijos de un trayecto: /*/*/*/* Obtiene todos los nodos que están en cuarto nivel //*/reparto/* Obtiene todos los interpretes, ya que son hijos de reparto.	
text()	Devuelve los nodos de tipo texto, es decir los valores. Por ejemplo //guionista/text() en nuestro ejemplo inicial devolvería: Julius J. Epstein Philip G. Epstein Howard Koch	
node( )	Localiza todos los nodos de cualquier tipo (no se usa habitualmente)	

8.5 Predicados

Las expresiones que permiten restringir o filtrar un conjunto de nodos dentro del trayecto especificado siguiendo un criterio determinado se denominan predicados, y se escriben entre corchetes **[…]**.

Para continuar haciendo más ejemplos ampliaremos nuestros XML de películas, añadiendo otra obra maestra:

```
<?xml version="1.0" encoding="UTF-8"?>
<filmoteca>
<pelicula
            estreno="1942"  minutos="102 ">
    <titulo>Casablanca</titulo>
    <director>Michael Curtiz</director>
    <guion>
        <guionista>Julius J. Epstein</guionista>
        <guionista>Philip G. Epstein</guionista>
        <guionista>Howard Koch</guionista>
    </guion>
    <reparto>
       <interprete papel="protagonista">Humphrey Bogart</interprete>
       <interprete papel="protagonista">Ingrid Bergman</interprete>
       <interprete papel="secundario">Paul Henreid</interprete>
       <interprete papel="secundario">Claude Rains</interprete>
       <interprete papel="secundario">Conrad Veidt</interprete>
       <interprete papel="secundario">Curt Bois</interprete>
    </reparto>
    <esloganes>
        <eslogan> El mito hecho celuloide</eslogan>
        <eslogan>La obra maestra absoluta</eslogan>
        <eslogan>Puede que nunca se haga una película mejor</eslogan>
    </esloganes>
  </pelicula>

  <pelicula estreno="1960"  minutos="125 ">
    <titulo>El apartamento</titulo>
    <director>Billy Wilder</director>
       <guion>
          <guionista>Billy Wilder</guionista>
          <guionista>I.A.L. Diamond</guionista>
       </guion>
       <reparto>
          <interprete papel="protagonista">Jack Lemmon</interprete>
          <interprete papel="protagonista">Shirley MacLaine</interprete>
          <interprete papel="secundario">Ray Walston</interprete>
          <interprete papel="secundario">Edie Adams</interprete>
       </reparto>

      <esloganes>
          <eslogan>La obra maestra del genio Wilder</eslogan>
          <eslogan>Para los que no creemos en Dios pero creemos en Billy Wilder</eslogan>
          <eslogan>Perfecta para hacerme reír y llorar</eslogan>
      </esloganes>
  </pelicula>
  <!-- otras peliculas  -->
<filmoteca>
```

(peliculas2.xml)

Ejemplos

/filmoteca/pelicula[titulo="Casablanca"]
Nos extraerá los nodos de la pelicula *Casablanca*

/filmoteca/pelicula[director="Billy Wilder"]
No extraerá los nodos de la pelicula *El apartamento*

/filmoteca/pelicula[director="Billy Wilder"]/reparto
Nos devuelve

```
<reparto>
  <interprete papel="protagonista">Jack Lemmon</interprete>
  <interprete papel="protagonista"> Shirley MacLaine</interprete>
  <interprete papel="secundario">Ray Walston</interprete>
  <interprete papel="secundario">Edie Adams</interprete>
</reparto>
```

Usando el carácter @ para hacer restricciones respecto a los atributos podemos hacer://interprete[@papel]

y nos devolvería todos los elementos que contienen el atributo *papel*. (Útil cuando el atributo es optativo).

Podemos incorporarlo en la ruta:

//*[@*]/text()

y nos devuelve todos los valores de los elementos que tendrían algún atributo, en nuestro caso:

```
Humphrey Bogart
Ingrid Bergman
Paul Henreid
Claude Rains
Conrad Veidt
Curt Bois
Jack Lemmon
Shirley MacLaine
Ray Walston
Edie Adams
```

Se pueden suceder varios predicados, y su resultado dependerá de que se cumplan todos y cada uno de ellos al mismo tiempo (un *AND* lógico):

//pelicula[titulo="Casablanca"] [director="Michael Curtiz"] Nos selecciona la película Casablanca

pero

//pelicula[titulo="Casablanca"] [director="Billy Wilder"] No encuentra ningún nodo

8.6 Funciones

En XPath se dispone de una importante biblioteca de funciones predefinidas (más de 100) que nos permiten encontrar nodos ampliando las herramientas y terminología que hemos visto mediante la sintaxis vista hasta ahora.

El nombre de una función siempre termina con paréntesis **( )** y pueden contener información que se pasa como ***parámetro*** dentro de ellos.

Comentamos aquí una pequeña selección, agrupándolas según criterios de su aplicación:

Funciones sobre los nodos

name()	Devuelve el nombre del nodo ejemplo://*[name()='reparto']
text()	Obtiene el contenido de un nodo que tenga un valor Ejemplo: /pelicula/guion/guionista/text() nos devuelve el nombre de los guionistas

Funciones de posición

position()	Obtiene la posición de un nodo en un conjunto de nodos. Ejemplo: //interprete[position()=2] Nos devuelve: <interprete papel="protagonista"> Ingrid Bergman</interprete> NOTA: se puede abreviar como //interprete[2]
last()	Obtiene el último de un conjunto de nodos Ejemplo: //interprete[last()] Nos devuelve: <interprete papel="secundario">Curt Bois</interprete>
count()	Devuelve la cantidad de nodos localizado en el argumento Ejemplo: count(/*/*/eslogan) devolvería 3 Pero en el XML CopyEditor no podemos comprobarlo, ya que solo devuelve nodos. Aunque podemos hacer una prueba como //*[count(*)=3] que selecciona los elementos con tres hijos, por lo que nos devolvería los tres guionistas y los tres eslogan

Funciones de cadena

string()	Convierte números a cadenas, aunque no es necesario ya que en principio los números están en formato texto,pero puede ser útil para convertir resultados de un cálculo.
string-length()	Devuelve el número de caracteres de la cadena que contiene como parámetro. Podemos probar con //*[string-length(name()) =6] y nos devolvería el elemento titulo por ser el único que tiene 6 caracteres.

stars-with()	Comprueba si el primer parámetro comienza por la letra (o letras) del segundo parámetro ejemplo: //*[starts-with(name(),'t')] Devolvería los elementos ***t****itulo*
contains()	Indica si el primer parámetro contiene el segundo. Por ejemplo : //*[contains(name(),'tul')] tambien devuelve los elementos *ti****tul****o*

Nota: como se ha dicho al principio, existen más de 100 funciones predefinidas, aquí solo se ha hecho una referencia a alguna de ellas, las más importantes y/o que podían comprobarse con la herramienta básica utilizada en este tema. En el próximo tema se ampliará alguna función también usual y que ya tienen sentido dentro del contexto de XSALT.

8.7 Expresiones y operadores

Solo a título informativo y de ampliación, estos son los operadores que se permiten utilizar en las expresiones XPath, con lo que se puede realizar expresiones más complejas:

Operadores para expresiones numéricas	+ (suma) - (resta) * (multiplicación) Div (división) Mod (resto)
Operadores de relación para comparaciones	= (igualdad) != (desigualdad) < , <= , >, >= (para las distintas comparaciones relacionales)
Operadores Booleanos	or, and y not