U3- GESTIÓN Y ALMACENAMIENTO DE INFORMACIÓN EN FORMATO XML

U3- Gestión y almacenamiento de la información en formato XML

1. Sistemas de almacenamiento de información.
2. Inserción y extracción de información en XML.
3. Búsqueda de información en documentos XML: Lenguajes de consulta y manipulación.
    - 3.1. Xpath
        - A. Los nodos en XPath
        - B. Cómo seleccionar los nodos de un documento XML
        - C. Utilizando predicados para mejorar las búsquedas
        - D. Combinar varias rutas
        - E. Comodines
        - F. Los axes en Xpath
        - G. Operadores

1. Sistemas de almacenamiento de información

Existen tres formas de almacenar información XML:

1. En ficheros.
2. En bases de datos XML no nativas llamadas también *XML enabled* (habilitadas)
3. En bases de datos nativas XML.

1. Ficheros

No es la mejor opción, puesto que no puede garantizarse la concurrencia, integridad de atomicidad, el nivel de seguridad que ofrecen otros sistemas como las bases de datos, etc.

1. Sistemas de almacenamiento de información

2. Bases de datos XML no nativas o XML enabled

Almacenamiento en bases de datos relacionales. Son las llamadas bases de datos *XML enabled o habilitadas.* En realidad, es una base de datos relacional que convierte los documentos XML en un esquema relacional y, una vez hecho esto, vuelca los datos en dicho esquema.

Estas bases de datos suelen ser bases de datos tradicionales como las bases de datos relacionales y permiten XML como entrada y pueden generar XML como salida.

La base de datos se encarga de procesar el XML sin necesidad de utilizar ningún *middleware*.

Estas bases de datos XML *enabled* presentan los siguientes problemas:

- No puede restaurarse el documento original.
- Diferencia de filosofía. XML es jerárquico, mientras que las bases de datos XML *enabled* son relacionales.

1. Sistemas de almacenamiento de información

3. Bases de datos XML

Su modelo interno se basa en XML y su unidad de almacenamiento es el documento XML (para las bases de datos relacionales es la fila).

Son las que mejores prestaciones ofrecen con este tipo de documentos. A diferencia de las bases de datos XML *enabled*, puede recuperarse el documento original, puesto que lo almacenan sin alterarlo.

Además, estas bases de datos utilizan el modelo jerárquico del documento XML mediante nodos y cada nodo tendrá sus comentarios, atributos, instrucciones, etc.

En estos entornos nativos XML, ya no se utilizan lenguajes como SQL, sino otros adaptados a este modelo como pueden ser XQuery y XPath.

Algunas bases de datos XML son:

- *MarkLogic.* Con licencia propietaria.
- *BaseX.* Con licencia BSD.
- *eXist-db.* Con licencia LGPL.
- *sedna.* Con licencia Apache.

Figura 6.1
Algunas bases de datos XML.

2. Inserción y extracción de información en XML

Muchas herramientas son capaces de guardar y leer documentos en XML. Esto es así porque XML es un formato universal y sencillo de generar y *parsear* (leer). No obstante, en este apartado, va a presentarse una herramienta muy interesante para cualquier programador web que se llama *ImportXML* y está en la *suite* de Google Docs, específicamente en la hoja de cálculo.

2. Inserción y extracción de información en XML

Quieren importarse todos los enlaces que tenga la página web de la Wikipedia que hablan de HTML: https://es.wikipedia.org/wiki/HTML
Utiliza la función ImportXML de las hojas de cálculo de Google.

```
IMPORTXML(url; consulta_xpath)
```

- *url:* URL de la página que desea examinarse (hay que incluir el protocolo como, por ejemplo, http://). La URL tiene que ir entre comillas (también puede ser una referencia a una celda).
- *consulta_xpath:* es la consulta en lenguaje XPath que va a ejecutarse sobre la URL anterior. XPath se estudiará en un apartado posterior de este capítulo.

¿Para qué puede servir esta herramienta? Para extraer cualquier información de una página web o cualquier otro formato estructurado de datos. Imagine que quieren extraerse los correos electrónicos de los integrantes de un foro, los *nicks,* los hiperenlaces, etc. Todas esas tareas que pueden llevar mucho tiempo pueden hacerse de forma rápida con esta herramienta.

3. Búsqueda de información en documentos XML: Lenguajes de consulta y manipulación

1. **Xpath**: Lenguaje para seleccionar nodos dentro de un DOM.
2. **Xquery**:
   - Es un superconjunto de Xpath.
   - Ofrece sintaxis FLWOR (Incluye cláusulas For, Let, Where, Order by, y Return).
   - Parecido a SQL.
3. **Xpointer**
   - Incluye Xpath.
   - Más sofisticado.
   - Permite seleccionar nodos dentro de fragmentos XML, o incluso partes de nodos.

3.1. XPath

XPath es un lenguaje de rutas para XML *(path language)* y utiliza una sintaxis *path like* para identificar y navegar entre los nodos de un documento XML.

Una de las razones que hacen a XPath tan potente es por el número de funciones predefinidas de las que dispone (más de 200). Existen funciones para tratar *strings*, booleanos, valores numéricos, comparar fechas y horas, manipulación de secuencias y nodos, etc.

Las expresiones XPath pueden utilizarse en Java, JavaScript, PHP, Python, C, C++, etc.

XPath utiliza un analizador cuya función es crear un árbol de nodos a partir de un documento XML. Los nodos creados en estos árboles son de siete tipos: nodo raíz, elemento, texto, atributo, *namespace* (espacio de nombres), instrucción procesable y comentario.

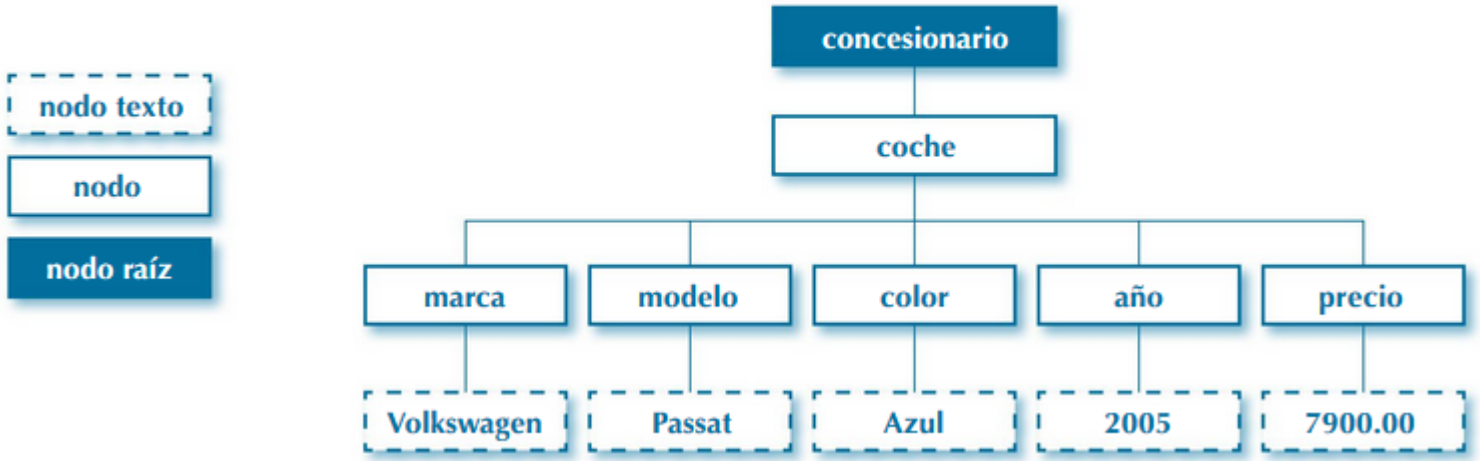
SABÍAS QUE...

XPath es una recomendación del W3C desde el 16 de noviembre de 1999 en el que apareció la versión 1.0 de XPath. Actualmente, la versión es la 3.0 y data del 2014.

3.1. XPath

```xml
<?xml version="1.0" encoding="UTF-8"?>
<concesionario>
    <coche>
        <marca>Volkswagen</marca>
        <modelo>Passat</modelo>
        <color>Azul</color>
        <año>2005</año>
        <precio>7900.00</precio>
    </coche>
</concesionario>
```

A partir de este documento XML, XPath creará un árbol de nodos



3.1. XPath

A. Los nodos en XPath

```xml
<?xml version="1.0" encoding="UTF-8"?>
<concesionario>
    <coche>
        <marca>Volkswagen</marca>
        <modelo>Passat</modelo>
        <color>Azul</color>
        <año>2005</año>
        <precio>7900.00</precio>
    </coche>
</concesionario>
```

Los nodos en las estructuras arborescentes de XML están relacionados unos con otros. Pueden encontrarse nodos:

- *Padre (parent)*. Un nodo es padre de otro cuando este último desciende de él. Por ejemplo, coche desciende de concesionario.
- *Hijo (children)*. El hijo es el nodo que desciende del nodo padre. Por ejemplo, coche es nodo hijo de concesionario.
- *Hermanos (siblings)*. Son aquellos nodos hijo del mismo padre (marca, modelo, color, etc.).
- *Antecesores (ancestors)*. Los nodos coche y concesionario serán antecesores de color, puesto que jerárquicamente están en un orden superior directo.
- *Descendientes (descendant)*. Son los hijos, hijos de los hijos, etc., de un nodo.

3.1. XPath

B. Cómo seleccionar los nodos de un documento XML

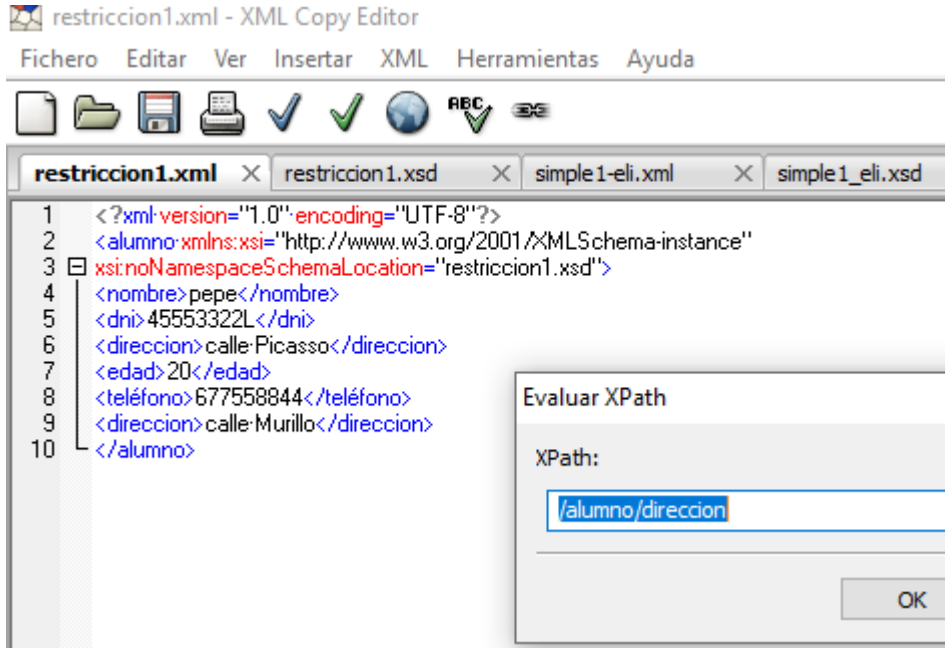
Para seleccionar los nodos de un documento XML, XPath utiliza expresiones y, de esa manera, un nodo puede ser seleccionado siguiendo una serie de pasos o ruta.

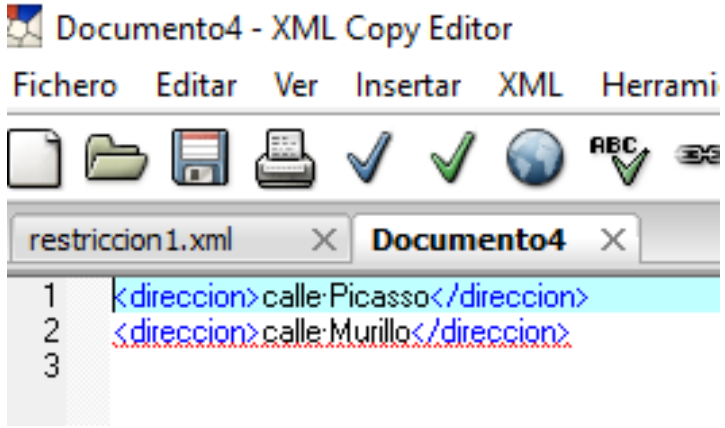
Algunas de las expresiones más utilizadas:

Expresión	Descripción
nombre_nodo	Selecciona todos los nodos con el nombre *nombre_nodo*.
/	Selecciona desde el nodo raíz.
//	Selecciona los nodos del documento que cumplan los criterios de selección desde el nodo actual sin importar dónde se encuentren.
.	Selecciona el nodo actual.
..	Selecciona el nodo padre del nodo actual.
@	Selecciona atributos.

3.1. XPath

- Podemos probarlo en XML Copy Editor -> XML -> Evaluar XPath (F9)
- Cuando introduzcamos el xpath concreto, se mostrarán en otro documento los nodos resultantes.





3.1. XPath

- Por ejemplo, para seleccionar los nodos *ingrediente*:

  /pizzas/pizza/ingrediente

- Si no supiéramos que ingrediente es hijo de *pizza*, sino simplemente que está en el documento como descendiente de *pizzas*, podríamos indicarlo así:

  /pizzas//ingrediente

- Para situarnos en el nodo padre de los nodos *ingrediente*:

  /pizzas//ingrediente/..

- Para seleccionar todos los atributos con nombre *nombre:*

  //@nombre

```
▼<pizzas>
  ▼<pizza nombre="Barbacoa" precio="8">
      <ingrediente nombre="Salsa Barbacoa"/>
      <ingrediente nombre="Mozzarella"/>
      <ingrediente nombre="Pollo"/>
      <ingrediente nombre="Bacon"/>
      <ingrediente nombre="Ternera"/>
    </pizza>
  ▼<pizza nombre="Margarita" precio="6">
      <ingrediente nombre="Tomate"/>
      <ingrediente nombre="Jamón"/>
      <ingrediente nombre="Queso"/>
    </pizza>
  ▼<pizza nombre="Tres Quesos" precio="10">
      <ingrediente nombre="Cabrales"/>
      <ingrediente nombre="Mozzarella"/>
      <ingrediente nombre="Manchego"/>
      <ingrediente nombre="Jamón"/>
      <ingrediente nombre="Bacon"/>
    </pizza>
</pizzas>
```

3.1. XPath

C. Utilizando predicados para mejorar las búsquedas

Los predicados son modificadores que se colocan entre corchetes y sirven para extraer un nodo concreto o un nodo con un determinado valor.

Teniendo en cuenta el ejemplo anterior, se ofrece una serie de predicados con el resultado correspondiente en el cuadro

Expresión	Resultado
/concesionario/coche[1]/marca	Seleccionaría la marca del primer coche del concesionario. En algunos navegadores, los nodos empiezan a contar desde el 0.
/concesionario/coche[last()]/marca	Seleccionaría la marca del último coche del concesionario.
/concesionario/coche[last() -1]/marca	Seleccionaría la marca del penúltimo coche del concesionario.
/concesionario/coche[position()<4]/marca	Seleccionaría la marca de los tres primeros coches del concesionario.
/concesionario/coche[precio>8000.00]/modelo	Selecciona el modelo de aquellos coches cuyo precio sea superior a 8000 euros. El resultado es el siguiente: modelo: Touran modelo: Astra

```xml
<?xml version="1.0" encoding="UTF-8"?>
<concesionario>
    <coche combustible="D">
        <marca>Volkswagen</marca>
        <modelo>Passat</modelo>
        <color>Azul</color>
        <año>2005</año>
        <precio>7900.00</precio>
    </coche>
    <coche combustible="G">
        <marca>Volkswagen</marca>
        <modelo>Touran</modelo>
        <color>Azul</color>
        <año>2007</año>
        <precio>8200.00</precio>
    </coche>
    <coche combustible="D">
        <marca>Opel</marca>
        <modelo>Astra</modelo>
        <color>Negro</color>
        <año>2013</año>
        <precio>8490.00</precio>
    </coche>
</concesionario>
```

3.1. XPath

D. Combinar varias rutas

En ocasiones, no solamente quiere seleccionarse un nodo, sino varios nodos con distinto contenido. Imagínese que quiere conocerse el modelo y el precio de los coches del concesionario. En este caso, se utilizará el operador | que funciona como un AND lógico.

La ruta utilizada es la siguiente:

```
"concesionario/coche/modelo | concesionario/coche/precio"
```

3.1. XPath

E. Comodines

Comodines cuyo uso permite XPath

Comodín	Descripción
*	Selecciona cualquier nodo elemento.
@*	Selecciona cualquier nodo atributo.
node()	Selecciona cualquier nodo de cualquier clase.

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

<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>

<book>
  <title lang="en">Learning XML</title>
  <price>39.95</price>
</book>

</bookstore>
```

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

3.1. XPath

F. Los axes de XPath

Un eje representa una relación con el nodo de contexto (actual), y se utiliza para localizar nodos relativos a ese nodo en el árbol.

AxisName	Result
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
attribute	Selects all attributes of the current node
child	Selects all children of the current node
descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
namespace	Selects all namespace nodes of the current node
parent	Selects the parent of the current node
preceding	Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes
preceding-sibling	Selects all siblings before the current node
self	Selects the current node

3.1. XPath

F. **Los axes de XPath**

- Location Paths:

```
An absolute location path:

/step/step/...

A relative location path:

step/step/...
```

Cada paso se evalúa contra los nodos en el actual conjunto de nodos.

Un paso consiste en:

- un eje (define la relación de árbol entre los nodos seleccionados y el nodo actual)
- un nodo-test (identifica un nodo dentro de un eje)
- cero o más predicados (para perfeccionar el conjunto de nodos seleccionado)

La sintaxis para un paso de localización es:

```
axisname::nodetest[predicate]
```

3.1. Xpath

F. Los axes de Xpath (cont)

Examples

Example	Result
child::book	Selects all book nodes that are children of the current node
attribute::lang	Selects the lang attribute of the current node
child::*	Selects all element children of the current node
attribute::*	Selects all attributes of the current node
child::text()	Selects all text node children of the current node
child::node()	Selects all children of the current node
descendant::book	Selects all book descendants of the current node
ancestor::book	Selects all book ancestors of the current node
ancestor-or-self::book	Selects all book ancestors of the current node - and the current as well if it is a book node
child::*/child::price	Selects all price grandchildren of the current node

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

<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>

<book>
  <title lang="en">Learning XML</title>
  <price>39.95</price>
</book>

</bookstore>
```

3.1.XPath

- G. **Operadores**

Operator	Description	Example
|	Computes two node-sets	//book | //cd
+	Addition	6 + 4
-	Subtraction	6 - 4
*	Multiplication	6 * 4
div	Division	8 div 4
=	Equal	price=9.80
!=	Not equal	price!=9.80
<	Less than	price<9.80
<=	Less than or equal to	price<=9.80
>	Greater than	price>9.80
>=	Greater than or equal to	price>=9.80
or	or	price=9.80 or price=9.70
and	and	price>9.00 and price<9.90
mod	Modulus (division remainder)	5 mod 2

Ejemplo w3schools

- https://www.w3schools.com/xml/xpath_examples.asp