UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Interacțiunea om-calculator

28. Proiectare şi implementare de interfețe declarative la documente XML

XPATH
Prezentare

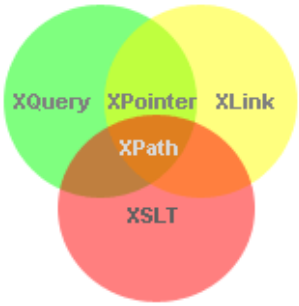
1. Despre XPath

XPath este un „limbaj" ce permite regasirea unor parti dintr-un fisier XML precum si „navigarea" prin fisiere XML. Documentele XML au o structura arborescenta, datorita relatiei „tata-fiu" a nodurilor din acel document, ceea ce permite limbajului XPath sa poata sa acceseze diferite elemente din cadrul documentelor XML. Acesta este si *scopul principal* al acestui limbaj. Datorita acestei posibilitati, limbajul permite selectarea nodurilor sau a unor seturi de noduri din documentul XML.

Pentru a permite accesarea diferitelor elemente ale documentului XML, XPath foloseste o sintaxa foarte asemanatoare cu cea utilizata pentru accesarea fisierelor intr-un sistem de fisiere UNIX (ex: „/grupa/studenti/student/nume" pentru fisierul XML de mai jos). XPath nu respecta structura documentelor XML (nu este un fisier XML).

Numele limbajului se trage de la faptul ca utilizeaza „cai" (PATHS) pentru identificarea elementelor XML.

XPath este un standard W3C de sine statator si o componenta de baza a XSLT, XLink, XQuery si XPointer, asa cum se poate observa din figura de mai jos:



Poza downloadata pe data de 18.10.2007 de la adresa http://www.w3schools.com/xpath/default.asp

2. Sintaxa

In cadrul XPath exista sapte tipuri de noduri: element (nodurile documentului), atribut, text, namespace, instructiuni de procesare, comentariu, si radacina documentului (root). Datorita structurii arborescente a documentelor XML, intre nodurile acestuia se stabilesc *relatii de rudenie* (ex: parinte, copil, urmas, strabun, etc.).

Selectarea unui nod se face prin urmarea unei **cai** XPath. Aceste cai pot fi **absolute** si atunci incep cu „/” sau pot fi **relative** si atunci NU incep cu „/”. In primul caz, calea catre destinatie porneste din radacina documentului, iar in cel de-al doilea, calea porneste din nodul curent. O cale XPath contine unul sau mai multi pasi separati de „/”:

- pentru expresii absolute: /PAS1/PAS2/...
- pentru expresii relative: PAS1/PAS2/...

Pasii unei expresii XPath sunt evaluati in ordine, de la stanga la dreapta. Daca s-a ajuns la evaluarea PAS2, aceasta se realizeaza relativ la fiecare nod rezultat in urma evaluarii PAS1.

Pentru o mai usoara intelegere a acestor notiuni, vom folosi urmatorul fisier XML pentru exemplificare.

```
<?xml version=„1.0” encoding=„ISO-8859-1”?>
<grupa id=„324CB”>
    <studenti>
        <student id=„1212”>
            <nume>Popescu</nume>
            <prenume>Ion</prenume>
        </student>
        <student id=„1213”>
            <nume>Ionescu</nume>
            <prenume>Dan</prenume>
        </student>
    </studenti>
    <advisor id=„323”>
        <nume>Georgescu</nume>
        <prenume>Alex</prenume>
    </advisor>
</grupa>
```

Un **pas** dintr-o cale XPath este un tuplu care are urmatoarele trei componente:

nume_axa + nod_test + lista_de_predicate (optional)

ex: „child::student[nume=Ionescu]”

Fiecare componenta a unui pas selecteaza anumite noduri, iar privite de la stanga la dreapta, fiecare adauga un set de restrictii suplimentar (ex: nume_axa=„child” se refera la toti copii nodului curent, nod_test=„student” se refera la copiii nodului curent care au numele „student”, iar lista_de_predicate=„nume=Ionescu” se refera la copiii nodului curent care au numele „student" si care au un nod „nume” cu valoarea Ionescu.

In continuare vor fi prezentate cele trei componente ale unui pas:

- **nume_axa** = specifica relatia intre nodurile care sunt selectate de catre acest pas si nodul curent (ex: „child” specifica faptul ca nodurile identificate in pasul curent sunt copii ai nodului curent; nod curent in acest caz e nodul identificat de pasul anterior). XPath se bazeaza pe navigarea de-a lungul acestor axe.
  OBSERVATII:
  - Daca nu se specifica „nume_axa” atunci se utilizeaza axa implicita, aceasta fiind axa „child::”
  - Atributele sunt specificate fie folosind numele de axa „attribute::” fie prescurtarea „@”(ex: „@id”)
- **nod_test** = specifica fie numele nodului de identificat (ex: „student”) fie tipul acestuia (ex: „child::text()” identifica nodurile fii care sunt de tipul TEXT = acele noduri de tip „#PCDATA” specificat in DTD-ul unui document XML.
  OBSERVATIE: Intr-un document XML, un text este considerat si el un element. (ex: <e1></e1>Salut!<e2></e2>, aici „Salut!” este tot un element, de tip TEXT)
- **lista_de_predicate** = sunt functii folosite pentru a filtra dupa anumite conditii nodurile selectate in pasul curent

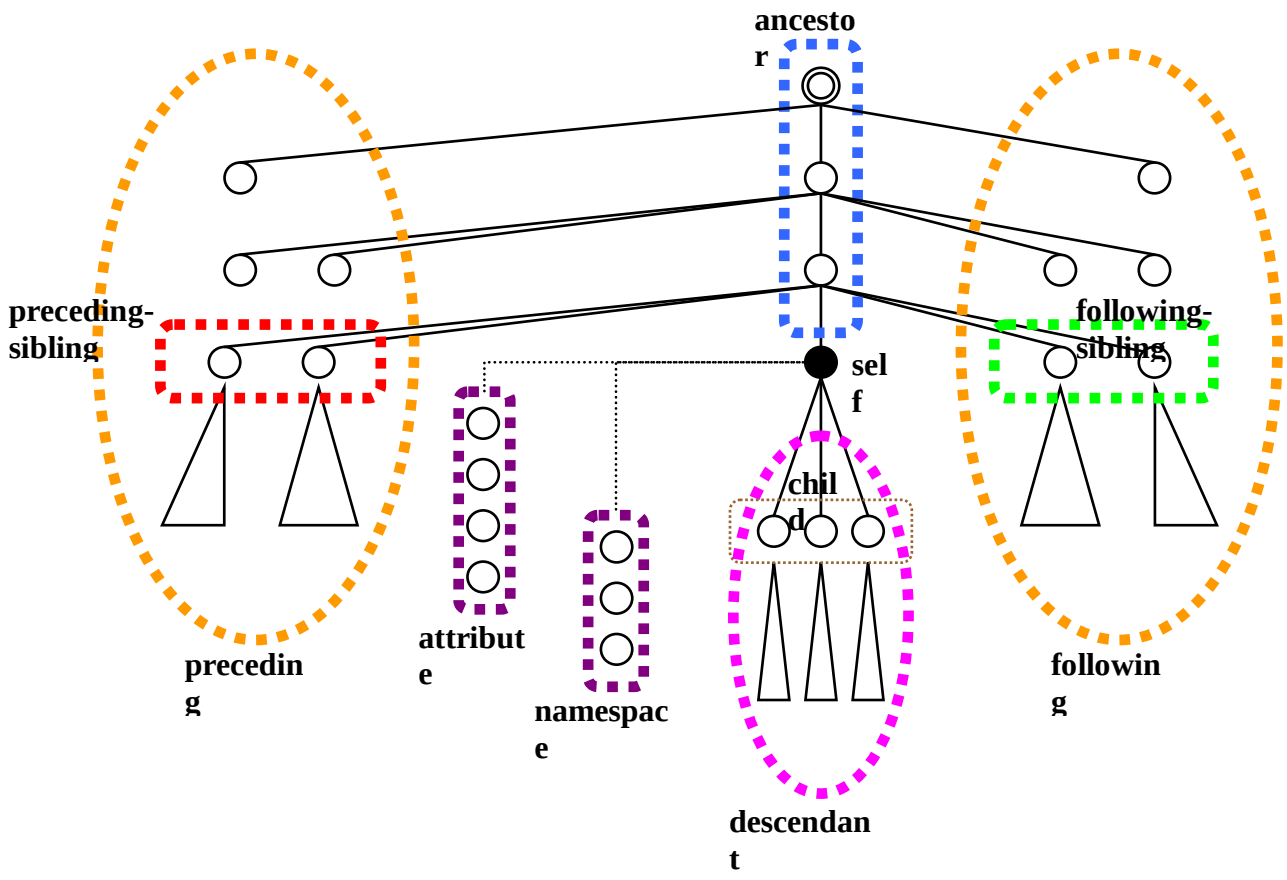
2.1 Axe XPath

In cadrul documentelor XML exista mai multe axe: ancestor, descendant, ancestor-or-self, descendant-or-self, preceding, following, preceding-sibling, following-sibling, parent, child, self, attribute, namespace. In tabelul de mai jos, se poate observa descrierea acestor axe, iar schema aflata mai jos intregeste imaginea, prezentand intr-o forma grafica cum sunt situate aceste axe in cadrul unui document XML:

Numele Axei	**Rezultatul evaluarii**
ancestor	Selecteaza toti strabunii nodului curent (parinti, bunici, etc.)
descendant	Selecteaza toti urmasii nodului curent (copii, nepoti, etc.)
ancestor-or-self	Selecteaza toti strabunii nodului curent, precum si nodul curent
descendant-or-self	Selecteaza toti urmasii nodului curent, precum si nodul curent
preceding	Selecteaza tot ce este in document inaintea tagului de inceput al nodului curent
following	Selecteaza tot ce urmeaza in document dupa tagul de inchidere al nodului curent
preceding-sibling	Selecteaza toate rudele nodului curent aflate inaintea acestuia
following-sibling	Selecteaza toate rudele nodului curent aflate dupa acesta
parent	Selecteaza parintele nodului curent
child	Selecteaza toti copii nodului curent

self	Selecteaza nodul curent
attribute	Selecteaza toate atributele nodului curent
namespace	Selecteaza toate nodurile cu namespace-urile nodului curent

Dupa cum se poate observa din schema de mai jos, unele dintre aceste axe descriu cate un singur nod (ex: parent, self), in timp ce altele descriu o multime de noduri (care uneori poate fi vida – de ex. ancestors pentru primul nod din fisierul XML).



Axele unui document XML
Poza este realizata de Arnaud Sahuguet si afost downladata pe data de 18.10.2007 de la adresa lambda.uta.edu/cse6331/spring02/**XML**2.ppt

Exemplu	**Rezultat**
child::student	Selecteaza toate nodurile student care sunt copii ai nodului curent
attribute::id	Selecteaza atributul id al nodului curent
child::*	Selecteaza toti copiii ai nodului curent
attribute::*	Selecteaza toate atributule ai nodului curent
child::text()	Selecteaza toate nodurile de tip text care sunt copii ai nodului curent
child::node()	Selecteaza toti copiii de tip nod ai nodului curent
descendant::student	Selecteaza toate nodurile student care sunt descendentii nodului curent
ancestor::nume	Selecteaza toate nodurile nume care sunt stramosi ai nodului curent

2.2 Noduri in XPath

Nod_test poate lua una din valorile de mai jos:

- text() returneaza un element de tip text, nu are taguri delimitatoare
- node() returneaza un element oarecare (chiar si de tip TEXT) – echivalent cu (* sau @* sau text())
- name() returneaza numele tagului curent

Cele mai utile expresii XPath pentru selectarea nodurilor sunt:

Expresie	**Descriere**
nodename	Selecteaza toate nodurile care sunt copii ai nodului curent
/	Selecteaza tot ce se afla in document (toate nodurile de dupa radacina)
//	Selecteaza toate nodurile din document care satisfac criteriile de selectie, indiferent unde se afla in document, dar dupa nodul curent
.	Selecteaza nodul curent
..	Selecteaza parintele nodului curent
@	Selecteaza atributele

Wildcards – sunt utilizate pentru a selecta noduri

- * returneaza orice element de tip nod
- @* returneaza orice atribut
- node() returneaza orice nod de orice tip

Exemple:

- nume – toti copiii cu numele „nume” (Popescu, Ionescu, Georgescu)
- ./nume – toti copii de tip nume ai nodului curent (echivalent cu nume)
- child::nume – toti copii de tip nume ai nodului curent (echivalent cu nume)
- student/nume – toti nepotii cu numele „nume” si care sunt copiii nodului student (Popescu, Ionescu)
- */nume toti nepotii cu numele „nume” (Popescu, Ionescu, Georgescu)
- . – nodul curent
- / – nodul radacina
- .. – parintele nodului curent
- .// – nodul curent si toti descendentii lui
- // – nodul radacina si toti descendentii lui
- text() – toti copiii de tip text ai nodului curent
- node() – toti copiii nodului curent (include nodurile de tip text si pe cele de tip atribut)
- //nume – toate nodurile de tip nume din document
- //text() – toate nodurile de tip text din document
- @id – atributul id al nodului curent
- /descendant::node()/child::nume – toti nepotii cu numele „nume”

OBSERVATIE: .//nume, nume, si //nume returneaza elemente diferite: prima expresie returneaza toti descendentii de tip nume ai nodului curent, incluzandu-l si pe el, a doua intoarce doar

nodurile de tip nume care sunt copii ai nodului curent, iar ultima intoarce toate nodurile de tip nume gasite in document. (ex: sa presupunem ca nodul curent este studenti. Atunci .//nume returneaza Popescu, Ionescu; nume nu returneaza nimic deoarece nodul studenti nu are copii de tip nume, iar //nume returneaza Popescu, Ionescu, Georgescu)

2.3 Predicate Xpath

Predicatele sunt folosite pentru a identifica un anumit nod sau un nod care are o anumita valoare. Intotdeauna predicatele sunt scrise intre paranteze drepte ([]) si sunt conditii care elimina nodurile care nu le satisfac.

Exemple:

- [2] – al doilea copil al nodului curent
- nume[2] – al doilea copil de tip nume al nodului curent
- [last()] – ultimul copil al nodului curent
- student [nume=„Popescu”] – copii de tip student ai nodului curent care au unul sau mai multi copii de tip nume a caror valoare este „Popescu”
- studenti[.//nume = „Popescu”] – copilul de tip studenti ai nodului curent care au intre descendentii lor un element de tip nume care are valoarea „Popescu”
- student[@id < „1213”] – studentii care au valoarea atributului id < 1213
- /descendant::node()/child::nume[parent/attribute::id = “1212”] – toti nepotii cu numele „nume” si care au parintele ce are valoarea atributului id 1212

2.4 Operatori Xpath

Operator	Descriere	Exemplu	Valoarea Returnata
|	sau intre seturi de noduri	//book | //cd	Returneaza un set de noduri ce contine nodurile din book si cd
+	adunare	6 + 4	10
-	scadere	6 - 4	2
*	inmultire	6 * 4	24
div	impartire	8 div 4	2
mod	modulo (restul impartirii)	5 mod 2	1
=	egal	price=9.80	adevarat daca pretul e 9.80 fals daca pretul e 9.90
!=	diferit	price!=9.80	adevarat daca pretul e 9.90 fals daca pretul e 9.80
<	mai mic	price<9.80	adevarat daca pretul e 9.00 fals daca pretul e 9.80
<=	mai mic sau egal	price<=9.80	adevarat daca pretul e 9.00 fals daca pretul e 9.90
>	mai mare	price>9.80	adevarat daca pretul e 9.90 fals daca pretul e 9.80
>=	mai mare sau egal	price>=9.80	adevarat daca pretul e 9.90 fals daca pretul e 9.70

or	sau	price=9.80 or price=9.70	adevarat daca pretul e 9.80 fals daca pretul e 9.50
and	si	price>9.00 and price<9.90	adevarat daca pretul e 9.80 fals daca pretul e 8.50

OBSERVATII:

- XPath converteste fiecare operand la un numar inainte de a face evaluarea operatiei.
- Daca se face verificarea de egalitate intre o valoare si un set de noduri, se returneaza "true" daca exista cel putin un nod din set egal cu valoarea respectiva
- Daca se face verificarea de inegalitate intre o valoare si un set de noduri, se returneaza "true" daca exista cel putin un nod din set inegal cu valoarea respectiva

2.5 Functii Xpath

Functii pe seturi de noduri:

Nume	**Descriere**
nr=count(node_set)	nr. de noduri din set
node_set=id(value)	un set de noduri care au ca ID value
number=last()	pozitia numerica a ultimului nod din lista procesata
string=name(node)	numele nodului
local-name(node)	numele nodului, fara prefix (prefix:nume)
namespace-uri(node)	namespace-ul nodului
position()	pozitia nodului curent procesat din lista de noduri

Functii pe siruri:

Nume	**Descriere**
concat(s1,..)	un sir format din sirurile concatenate
contains(sir,subsir)	"true" daca sir contine subsir
normalize-space(sir)	elimina spatiile de la inceput si sfarsit
starts-with(sir,subsir)	"true" daca sir incepe cu subsir
string(value)	converteste la sir de caractere
string-length(sir)	lungimea sirului de caractere sir
substring(sir,start,len)	subsirul din sir ce incepe la poz start si are lungimea len
substring-after(sir,subsir)	subsirul ce se afla in sir dupa subsir
substring-before(sir,subsir)	subsirul din sir aflat inainte de subsir
translate(sir,str1,str2)	inlocuieste fiecare sir de caractere str1 din sir cu sirul de caractere str2

Functii cu numere:

Nume	Descriere
ceiling(nr)	intregul mai mic cel mai apropiat de nr (sau =)
floor(nr)	intregul mai mare cel mai apropiat de nr (sau =)
number(string)	converteste la o valoare numerica
round(nr)	cel mai apropiat intreg
sum(nodeset)	suma tuturor valorilor dintr-un nodeset

Functii pe booleene:

Nume	Descriere
boolean(value)	converteste la boolean
false()	returneaza false
true()	returneaza true
not(cond)	returneaza inversul lui cond
lang(language)	returneaza true daca language e identic cu valoarea nodului xsl:lang

Pe langa aceste functii mai exista functii de acces, de detectare si urmarire a erorilor, de context, functii ce lucreaza cu timpi si durate, precum si functii care lucreaza pe secvente. Mai multe exemple gasiti la adresa http://www.w3schools.com/xpath/xpath_functions.asp

3. Cazuri de utilizare

Acest limbaj este utilizat in orice aplicatie care doreste sa lucreze cu elementele unui fisier XML. In limbaje ca XSLT este folosit pentru a naviga prin document si pentru a identifica diferitele elemente din cadrul acestuia pentru a putea fi prelucrate, iar in limbaje ca XQuery este folosit pentru a identifica si a returna diferite componente ale documentului XML. De asemenea este folosit si de catre XLink si XPointer precum si de catre parsere ca XML DOM sau SAX.

4. Instrumente

Exista diferite instrumente care permit folosirea sintaxei XPath. Printre acestea amintim XMLSpy si StylusStudio dar si multe altele (inclusiv Java).

La adresa http://www.zrinity.com/developers/xml/xpath/ se regaseste un instrument care permite testarea online a corectitudinii expresiilor XPath relativ la documentele XML pe care aveti de aplicat aceste expresii. Pentru a putea testa diversele expresii XPath, se copiaza continutul documentului XML pe care se doreste sa se lucreze in fereastra pusa la dispozitie pe site (exista un camp in care se afla initial un exemplu de-al lor), iar dupa aceea se introduce expresia XPath in campul de deasupra celui in care ati introdus documentul XML (de asemenea, initial acolo este o expresie XPath). Dupa introducerea acesteia, se apasa butonul de rulare XPath care va calcula corectitudinea documentului XML si a expresiei XPath si va returna rezultatul. **Atentie!** Acest rezultat este returnat in doua feluri. In susul paginii aveti evidentiata portiunea din documentul XML la care se refera expresia XPath, iar dedesubt, aveti rezultatul acestei expresii, sub forma unor obiecte ce sunt repezentari ale elementelor din documentul XML ce sunt returnate.

In continuare va prezentam o implementare in Java care permite rularea expresiilor XPath.

```
public class XMLTest {

	public static void main(String argv[])
	{
		if (argv.length != 1)
		{
			System.err.println("numele fisierului");
			System.exit(1);
		}
		try{

		// cod de la http://xml.apache.org/xalan-j/xpath_apis.html
		//1. Instantiate an XPathFactory.
		  javax.xml.xpath.XPathFactory xpfactory =
				javax.xml.xpath.XPathFactory.newInstance();

		  // 2. Use the XPathFactory to create a new XPath object
		  javax.xml.xpath.XPath xpath = xpfactory.newXPath();

		  // 3. Compile an XPath string into an XPathExpression
		  javax.xml.xpath.XPathExpression expression = xpath.compile("EXPRESIE
XPATH DE TESTAT");

		  // 4. Evaluate the XPath expression on an input document
		 String result = expression.evaluate(new org.xml.sax.InputSource(new
FileInputStream(new File(argv[0]))));
		 System.out.println("result is "+result);
		}catch (Exception e){System.out.println(e);}
    }//main
}// class
```

Mai multe detalii referitoare la implementarea XPath in java se pot gasi aici.

Ca exercitiu, aveti de testat urmatoarele expresii atat folosind programelul scris in Java cat si folosind softul online de la adresa de mai sus si comparati rezultatele obtinute:

- grupa
- /grupa
- /grupa/studenti/student
- /grupa/studenti/student[2]
- /grupa//student[2]
- //nume
- //@id
- /grupa//studenti/student[position()<2]
- //student[@id= „1213”]

5. Concluzii

Xpath este un limbaj folosit pentru navigarea prin documentele XML si pentru identificarea si accesarea diferitelor elemente si atribute ale acestuia. Cu ajutorul acestui limbaj se pot regasi foarte usor informatii intr-un document XML. Acest limbaj este extreme de important, el stand la baza altor limbaje, cum ar fi XSLT, XQuery, XLink si XPointer. De asemenea, parserele de XML sunt construite tot pe baza acestui limbaj.

6. Bibliografie

- http://www.w3.org/TR/xpath
- http://www.zvon.org/xxl/XPathTutorial/General/examples.html
- http://www.w3schools.com/xpath/
- http://lambda.uta.edu/cse6331/spring02/**XML**2.ppt
- http://xml.apache.org/xalan-j/xpath_apis.html
- http://www.zrinity.com/developers/xml/xpath/
- http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/xpath/XPathExpression.html#evaluate(org.xml.sax.InputSource,%20javax.xml.namespace.QName)
- http://xml.apache.org/xalan-j/xpath_apis.html#namespacecontext
- http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/xpath/XPath.html#evaluate(java.lang.String,%20java.lang.Object,%20javax.xml.namespace.QName)

XQUERY
Prezentare

1. Despre XQuery

XQuery este o recomandare W3C si a fost creat pentru a permite interogarea documentelor XML si extragerea unor componente ale acestora (elemente sau atribute). Cea mai buna definire a XQuery este obtinuta prin analogia cu SQL. Astfel, XQuery este pentru XML ceea ce este SQL pentru bazele de date. Acest limbaj se bazeaza pe XPath, cele doua limbaje folosind acelasi model al datelor si suportand aceiasi operatori si aceleasi functii. XQuery este suportat de catre toate motoarele mari ce actioneaza pe baze de date (IBM, Oracle, Microsoft, etc.)

2. Sintaxa

Cateva notiuni de baza ale sintaxei:

- XQuery este un limbaj case-sensitive
- Elementele, atributele si variabilele XQuery trebuiesc sa fie nume XML valide
- Valorile XQuery trebuiesc puse fie intre ghilimele, fie intre apostrofuri
- O variabila XQuery se defineste folosind $NUME, unde NUME este numele variabilei respective (ex: $doc)
- Comentariile XQuery sunt delimitate de (: si de :) (ex: (: XQuery Comentariu :)

Expresiile XQuery au o structura de FLWOR, care este un acronim pentru „For, Let, Where, Order by, Return”. Aceasta structura extrage niste noduri din fisierul XML, aplica un predicat pentru a elimina din nodurile selectate, iar apoi construieste un rezultat.

Instructiunile FOR si LET genereaza o lista de noduri extrase din document, mentinand ordinea acestora. Aceasta instructiune este similara cu FROM dintr-un query SQL. Diferenta dintre cele doua instructiuni este faptul ca in cazul instructiunii FOR, variabila var se leaga pe rand la fiecare element din expresia expr, facand posibila iterarea, in timp ce in cazul instructiunii LET, variabila var se leaga la intreaga lista definita de expresia expr, returnand o singura valoare.
Observatie: Instructiunea FOR se foloseste in forma *FOR $x **in** expr*, in timp ce instructiunea LET se foloseste in forma *LET $x = expr*.

Instructiunea ORDER BY defineste ordinea in care sunt considerate elementele extrase de FOR sau LET in cazul in care nu se doreste ordinea in care au fost acestea in documentul XML.

Instructiunea WHERE aplica un predicat (una sau mai multe restrictii) eliminand unele din nodurile extrase de catre FOR sau LET. Aceasta instructiune este echivalentul expresiei WHERE dintr-un query SQL.

Instructiunea RETURN este aplicata asupra fiecarui nod care a supravietuit conditiei/conditiilor impuse in WHERE, generand o lista ordonata de noduri la iesire. Aceasta instructiune este echivalentul expresiei RETURN dintr-un query SQL.

3. Cazuri de utilizare

Pentru a simplifica intelegerea sintaxei XQuery, vom folosi din nou documentul XML dat ca exemplu in documentul de prezentare a XPath. Vom presupune ca numele acestui document este facultate.xml.

```
<?xml version=„1.0” encoding=„ISO-8859-1”?>
<grupa id=„324CB”>
    <studenti>
        <student id=„1212”>
            <nume>Popescu</nume>
            <prenume>Ion</prenume>
        </student>
        <student id=„1213”>
            <nume>Ionescu</nume>
            <prenume>Dan</prenume>
        </student>
    </studenti>
    <advisor id=„323”>
        <nume>Georgescu</nume>
        <prenume>Alex</prenume>
    </advisor>
</grupa>
```

Pentru a putea deschide documentul asupra caruia sa se aplice expresiile XQuery se foloseste **doc("nume_document.xml")**. Aceasta instructiune permite mai departe referirea oricarui element din documentul respectiv prin expresii ce respecta sintaxa Xpath.

Exemple de utilizare a expresiilor XQuery:

- In cazul in care dorim sa referim valoarea campului nume al nodului student care are valoarea atributului id „1213”, putem sa scriem o expresie XPath de forma: doc(„facultate.xml”)//student[@id=„1213”]/nume. Folosind sintaxa FLWOR XQuery

  for $x **in doc**(„facultate.xml”)//student
  where $x/@id = 1213

return $x/nume

se obtine rezultatul: <nume>Ionescu</nume>.
Pentru o mai buna intelegere, in continuare se va explica pe scurt cum lucreaza expresia de mai sus. Initial, prin instructiunea FOR (for $x in doc(„facultate.xml")), variabila x (definita prin $x) se leaga la **fiecare** din nodurile de tip student pe care le gaseste in document //student) si se obtin doua noduri studenti – cel cu id=„1212" si cel cu id=„1213"). Apoi, aceste noduri sunt filtrate cu ajutorul conditiei din WHERE, din cele doua noduri ramanand numai cel cu id=„1213"($x este leagata la cel de-al doilea nod student). Acest nod este trimis mai departe catre instructiunea RETURN pentru a se genera iesirea. Aici, se extrage nodul nume care este copilul nodului student care a supravietuit conditiei din WHERE si se afiseaza, astfel obtinandu-se rezultatul <nume>Ionescu</nume>.

- Rezultatele care se obtin in urma aplicarii unei expresii XQuery pot fi **sortate** in functie de unul din elemente:

  for $x **in doc**(„facultate.xml")//student
  where $x/@id >1000
  order by $x/@id
  return $x/nume

  se obtine rezultatul: <nume>Popescu</nume><nume>Ionescu</nume>.

- In cazul in care nu se doreste ca expresia XQuery sa intoarca un nod, ci se doreste sa se intoarca un text, aceasta se poate realiza folosind instructiunea **text()**:

  for $x **in doc**(„facultate.xml")//student
  where $x/@id = 1213
  return $x/nume/text()

  Rezultatul acestui query va fi: Ionescu

- Daca se doreste ca rezultatul obtinut sa fie un **document valid XML sau HTML**, aceasta se poate realiza incadrand continutul expresiei XQuery in elementul care se doreste, punand continutul acesteia intre acolade ({}):

  <?xml version=„1.0" encoding=„ISO-8859-1"?>
  <studenti>
  {
  for $x **in doc**(„facultate.xml")//student
  where $x/@id >1000
  order by $x/@id
  return $x/nume
  }
  <studenti>

Scriptul de mai sus va avea ca efect crearea unui document XML valid care va arata in felul urmator:

```
<?xml version=„1.0” encoding=„ISO-8859-1”?>
<studenti>
    <nume>Popescu</nume><nume>Ionescu</nume>
<studenti>
```

- Daca se doreste modificarea numelor sau tipurilor tagurilor, se poate folosi o instructiune similara cu:

```
<html>
<body>
<h1> Studenti </h1>
{
    for $x in doc(„facultate.xml”)//student
    where $x/@id >1000
    order by $x/@id
    return <p ID = „{$x/@id }”>{$x/nume/text()}</p>
}
</body>
</html>
```

Rezultatul acestui query va fi:

```
<html>
<body>
<h1> Studenti </h1>
<p ID =”1212”>Popescu</p>
<p ID =”1213”>Ionescu</p>
</body>
</html>
```

Observati faptul ca **elementele ce se doresc a fi procesate sunt incluse intre acolade** ({$x/@id }, {$x/nume/text()}).

- Pentru a intelege **diferenta esentiala dintre FOR si LET**, urmariti urmatoarele doua query-uri si rezultatele acestora

Expresie XQuery	**Rezultat**
for $x **in doc**(„facultate.xml”)//student **where** $x/@id > 1000 **return** <out>$x/nume</out>	<out><nume>Popescu</nume></out> <out><nume>Ionescu</nume></out>
let $x **= doc**(„facultate.xml”)//student **where** $x/@id > 1000 **return** <out>$x/nume</out>	<out> <nume>Popescu</nume> <nume>Ionescu</nume> </out>

- In cadrul expresiilor XQuery se pot folosi **expresii conditionale** de tipul **„if-then-else”.**

for $x **in doc**(„facultate.xml”)//student
where $x/@id >1000
return if ($x/@id >1212)
then <ionescu>{$x/prenume/text()}</ionescu>
else < popescu >{$x/prenume/text()}</popescu >

Rezultatul acestui query este: <ionescu>Dan</ionescu> < popescu >Ion</popescu >.

OBSERVATIE: Expresia continuta in IF trebuie pusa intre paranteze rotunde (). Else trebuie sa fie prezenta, dar poate sa fie vida (else ()).

- **Comparatiile in XQuery pot fi de doua feluri:**
  - **Generale:** =, !=, <, <=, >, >= → expresia intoarce adevarat relatia este respectata de catre cel putin un element
  - **Comparari de valori:** eq, ne, lt, le, gt, ge → expresia intoarce adevarat daca este un singur element care este comparat cu o valoare si daca relatia este respectata. Daca sunt mai mule elemente care se doresc a fi comparate, se intoarce eroare

Expresie XQuery	**Rezultat**
doc(„facultate.xml”)//student/@id> 1000	**Adevarat** (sunt doi studenti ai caror atribute id au valoare mai mare ca 1000)
doc(„facultate.xml”)//student/@id> 1212	**Fals** (sunt tot doi studenti, dar unul din cei doi are vlaoarea atributului egala cu 1212, deci conditia nu este respectata)
doc(„facultate.xml”)//student/@id gt 1000	**EROARE** (sunt doi studenti si datorita acestui lucru se intoarce eroare)

4. Instrumente

Exista diferite instrumente care permit folosirea sintaxei XQuery. Printre acestea amintim Altova XMLSpy si StylusStudio (care au versiuni de trial) dar si QUIP (care are un GUI usor de utilizat dar care nu a mai fost updatat din 2002 – are nevoie de java 1.3 pentru a rula GUI). La adresa http://www.w3.org/XML/Query/#products gasiti o lista mai cuprinzatoare a produselor ce permit utilizarea XQuery.

5. Concluzii

XQuery este un limbaj ce poate fi utilizat pentru extragerea de informatii din documentele XML, generarea de rapoarte, transformarea documentelor XML in XHTML si cautarea in documentele

Web a unor informatii relevante. XQuery este compatibil cu diverse standarde W3C, cum ar fi XML, Namespaces, XSLT, XPath, si XML Schema.

6. Bibliografie

- http://www.w3.org/TR/xquery/
- http://www.w3.org/TR/query-semantics/
- http://www.w3.org/TR/xmlquery-use-cases
- http://xml.apache.org/xalan-j/xpath_apis.html