

Języki XPath i XQuery

Patryk Czarnik

Instytut Informatyki UW

XML i nowoczesne technologie zarządzania treścią –
2007/08

Model danych XPath

Drzewo dokumentu
Sekwencje i atomy

Język XPath

Od podstaw
Ścieżki
XPath 1.0

Język XQuery

Struktura zapytania XQuery
Konstruktory węzłów
Funkcje

Dokument XML w modelu XPath

- ▶ Drzewo:
 - ▶ korzeń drzewa – węzeł dokumentu (nie element główny),
 - ▶ dzieci elementu – zawartość (nie atrybuty).
- ▶ Uwzględnianie przestrzeni nazw.
- ▶ Możliwość uwzględniania schematu.
- ▶ Rozwinięte sekcje CDATA oraz referencje do encji i znaków.
- ▶ Połączone sąsiednie węzły tekstowe.

Rodzaje węzłów w XPath

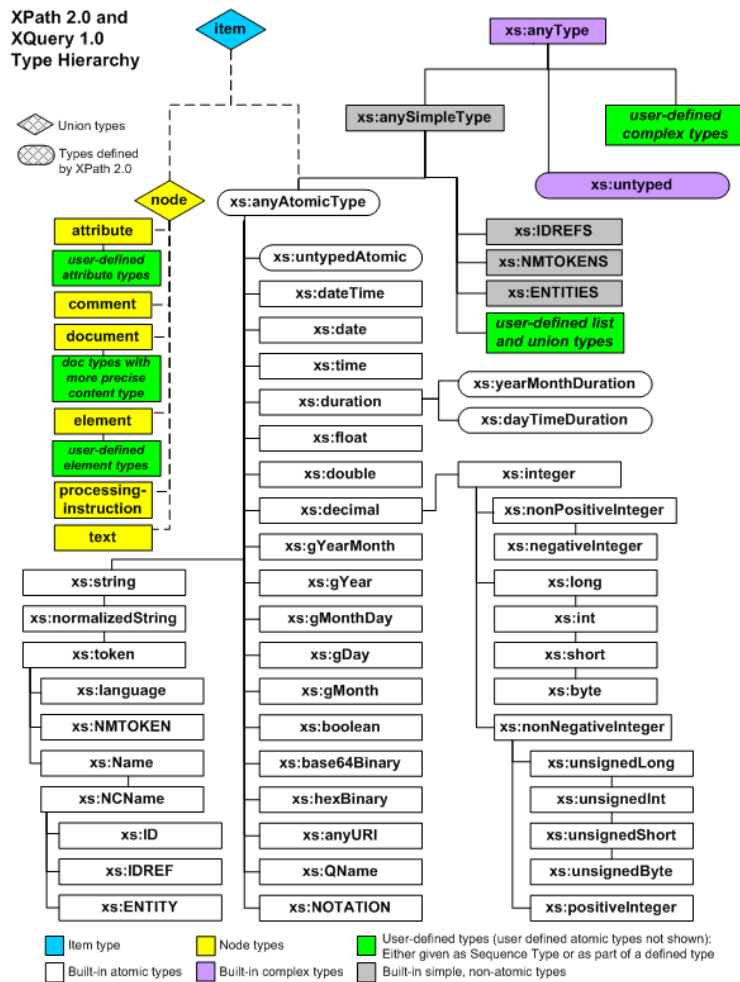
- ▶ Rodzaje węzłów:
 - ▶ węzeł dokumentu (korzeń),
 - ▶ element,
 - ▶ atrybut,
 - ▶ węzeł tekstowy,
 - ▶ instrukcja przetwarzania,
 - ▶ komentarz,
 - ▶ węzeł przestrzeni nazw.

Sekwencje

- ▶ Wartości w XPath – sekwencje.
- ▶ Elementy (*items*) sekwencji – węzły i wartości atomowe.
- ▶ Równoważność elementu i jednoelementowej sekwencji:
`(3.14) == 3.14`
- ▶ Spłaszczanie zagnieżdżonych sekwencji:
`(3.14, (1, 2, 3), 'Ala') == (3.14, 1, 2, 3, 'Ala')`

System typów

- ▶ Typy prymitywne XML Schema.
- ▶ Dodatkowo:
 - ▶ `xs:untyped`
 - ▶ `xs:untypedAtomic`
 - ▶ `xs:anyAtomicType`
 - ▶ `xs:dayTimeDuration`
 - ▶ `xs:yearMonthDuration`
- ▶ Możliwość używania typów zdefiniowanych w schemacie (prostych i złożonych).



Effective Boolean Value

- ▶ Częsta potrzeba traktowania dowolnej wartości jako wartości logicznej.
- ▶ Zasady zamiany:
 1. pusta sekwencja → **falsz**,
 2. sekwencja z węzłem na pierwszej pozycji → **prawda**,
 3. pojedyncza wartość boolowska → ta sama wartość,
 4. pojedynczy pusty napis → **falsz**,
 5. pojedynczy niepusty napis → **prawda**,
 6. pojedyncza liczba równa 0 lub NaN → **falsz**,
 7. inna pojedyncza liczba → **prawda**,
 8. inna wartość → błąd typu.

XPath – status

- ▶ XPath 1.0 (rekomendacja, listopad 1999)
 - ▶ używany m.in. w XSLT 1.0, XML Schema, XPointer.
- ▶ XPath 2.0 (kilka rekomendacji, styczeń 2007):
 - ▶ *XML Path Language (XPath) 2.0*,
 - ▶ *XQuery 1.0 and XPath 2.0 Data Model*,
 - ▶ *XQuery 1.0 and XPath 2.0 Functions and Operators*,
 - ▶ *XQuery 1.0 and XPath 2.0 Formal Semantics*,
 - ▶ używany w XSLT 2.0,
 - ▶ mocno związany z XQuery 1.0.

Ścieżki XPath – typowe zastosowanie

- ▶ `/lista/obiekt`
- ▶ `/lista/obiekt[@nazwa = 'pierwszy']`
- ▶ `//obiekt[@parzysty]`
- ▶ `//obiekt[position() mod 2 = 0]`
- ▶ `//obiekt[position() mod 2 = 0]/wyr[1]`
- ▶ `(//obiekt[position() mod 2 = 0]/wyr)[1]`

Literały i zmienne

Literały

- ▶ napisy: `'12.5'`, `"He said, "I don't like it.""`
- ▶ liczby: `12`, `12.5`, `1.13e-8`

Zmienne

- ▶ `$x` – referencja do zmiennej `x`,
- ▶ zmienne wprowadzane konstrukcjami:
 - ▶ XPath (`for`, `some`, `every`)
 - ▶ XQuery (`FLWOR`, `some`, `every`, parametry funkcji)
 - ▶ XSLT (`variable`, `param`)

Rzutowanie typów

Konstruktory typów

- ▶ `xs:date("2001-08-25")`
- ▶ `xs:float("NaN")`
- ▶ `adresy:kod-pocztowy("48-200")` (o ile schemat dostępny)

Operator `cast as`

- ▶ `"2001-08-25" cast as xs:date`
- ▶ ...

Funkcje

- ▶ Wywołania funkcji:
 - ▶ `fn:concat("abc", "xyz")`
 - ▶ `count(//obiekt/@parzysty)`
 - ▶ `moje:moja_funkcja(12, //jakieś_elementy)`
- ▶ 150 standardowych funkcji XPath:
 - ▶ w przestrzeni nazw
<http://www.w3.org/2005/xpath-functions>,
 - ▶ domyślna przestrzeń nazw dla funkcji.
- ▶ Definiowanie własnych funkcji
(zalecane w osobnej przestrzeni nazw):
 - ▶ w XQuery,
 - ▶ w XSLT (2.0),
 - ▶ w środowisku wykonania (np. EXSLT w Xalan).

Operatory

- ▶ Formalnie 68 operatorów XPath, przeciążona notacja.
- ▶ Arytmetyka:
 - ▶ `+`, `-`, `*`, `div`, `idiv`, `mod`
 - ▶ na datach i *duration* + i – zgodnie z typami.
- ▶ Sekwencje węzłów (w 1.0 „*node sets*”):
 - ▶ `union`, `|`, `intersect`, `except`
 - ▶ nie-węzły w sekwencjach – błąd typu,
 - ▶ wynik: sekwencja bez powtórzeń, zachowany porządek dokumentu.
- ▶ Wartości logiczne:
 - ▶ operatory `and`, `or`
 - ▶ `true()`, `false()`, `not(_)` to funkcje.

Operatory porównania

Porównania atomowe

- ▶ eq, ne, lt, le, gt, ge
- ▶ jeden z argumentów sekwencją pustą → wynik sekwencją pustą
- ▶ jeden z argumentów sekwencją >1 elementową → błąd typów,
- ▶ typy atomowe – intuicyjnie "normalne" porównanie,
- ▶ o ile typy pozwalają.

Porównania ogólne

- ▶ =, !=, <, <=, >, >=
- ▶ stosowane do sekwencji,
- ▶ *istnieje para elementów z lewej i prawej sekwencji, dla której zachodzi odpowiednie porównanie atomowe,*
- ▶ dla jednoelementowych zgodne z atomowymi.

Porównania ogólne – ciekawostki

- ▶ (Nie)Równość nie jest (nie)równością sekwencji:
 $(1, 2) = (2, 3)$
 $(1, 2) \neq (1, 2)$
- ▶ Równość nie jest przechodnia:
 $(1, 2) = (2, 3)$ – tak
 $(2, 3) = (3, 4)$ – tak
 $(1, 2) = (3, 4)$ – nie
- ▶ $X \neq Y$ nie jest równoważne $\text{not}(X = Y)$:
 $(1, 2) = (1, 2)$ – tak
 $(1, 2) \neq (1, 2)$ – tak
 $() = ()$ – nie
 $() \neq ()$ – nie

Wyrażenie warunkowe

```
if WARUNEK  
  then WYNIK1  
  else WYNIK2
```

- ▶ Liczy się *Effective Boolean Value*,
- ▶ obowiązkowo leniwa ewaluacja (tylko jedna gałąź).

Pętla po sekwencji

```
for $ZMIENNA in SEKWENCJA  
  return WYNIK
```

- ▶ *ZMIENNEJ* przypisywane kolejne wartości z *SEKWENCJI*,
- ▶ *WYNIK* obliczany z wybraną wartością *ZMIENNEJ*,
- ▶ wynik całości – sekwencja wyników częściowych.

Kwantyfikatory

some \$ZMIENNA in SEKWENCJA
satisfies WARUNEK

every \$ZMIENNA in SEKWENCJA
satisfies WARUNEK

- ▶ Liczy się *Effective Boolean Value*,
- ▶ możliwa leniwa ewaluacja,
- ▶ dowolna kolejność przechodzenia po sekwencji.

Ścieżki w XPath

- ▶ Najbardziej charakterystyczny rodzaj wyrażeń, przechodzenie po drzewie dokumentu.
- ▶ Ścieżka bezwzględna:
`/krok/krok ...`
- ▶ Ścieżka względna:
`krok/krok ...`
- ▶ Krok – składnia w pełni rozwinięta:
`oś::test-węzłów [predykat1] [predykat2] ...`
 - ▶ oś – kierunek w drzewie dokumentu,
 - ▶ test-węzłów – wybór węzłów po rodzaju, typie, nazwie,
 - ▶ predykaty – opcjonalne, dodatkowo filtrujące węzły.
- ▶
`/child::osoby/child::osoba[child::imię = 'Patryk']
/attribute::email`

Osie

- ▶ child
- ▶ descendant
- ▶ parent
- ▶ ancestor
- ▶ following-sibling
- ▶ preceding-sibling
- ▶ following
- ▶ preceding
- ▶ attribute
- ▶ namespace
- ▶ self
- ▶ descendant-or-self
- ▶ ancestor-or-self

Testy węzłów

- ▶ Rodzaj węzła, dodatkowo nazwa / typ:
 - ▶ `node()`
 - ▶ `document-node()`
 - ▶ `text()`
 - ▶ `comment()`
 - ▶ `processing-instruction()`,
`processing-instruction(xml-styleSheet)`,
 - ▶ `element()`, `element(osoba)`,
`element(*, osobaTyp)`, `element(osoba,`
`osobaTyp)`,
 - ▶ `attribute()`, `attribute(id)`,
`attribute(*, xs:integer)`, `attribute(id,`
`xs:integer)`,
- ▶ Nazwa:
 - ▶ `osoba`, `pre:*`, `*:osoba`, `*`,
 - ▶ rodzaj węzła: zależnie od osi element lub atrybut.

Predykaty

- ▶ Obliczane dla każdego węzła (węzeł na chwilę staje się w. kontekstowym).
- ▶ Każdy predykat „przesiewa” sekwencję.
- ▶ Typ wyniku:
 - ▶ liczba – porównywana z pozycją węzła w sekwencji (od 1),
 - ▶ nie liczba – decyduje *Effective Boolean Value*.
- ▶ Użycie poza ścieżkami (tzw. *filter expressions*):
(1 to 10) [. mod 2 = 0]

Skróty składniowe

- ▶ Oś `child` można pominąć.
- ▶ `@` przed nazwą zamiast osi `attribute`.
- ▶ `.` zamiast `self::node()`.
- ▶ `..` zamiast `parent::node()`.
- ▶ `//` zamiast `/descendant-or-self::node() /`.

Przykład

```
./para[@id = '123']
```

```
self::node()/descendant-or-self::node()  
  child::para[attribute::id = '123']
```

Sposób obliczania ścieżek

- ▶ Ścieżki obliczane od lewej do prawej.
- ▶ Dla każdego węzła z bieżącej sekwencji obliczany kolejny krok (wraz z predykatami).
 - ▶ `//obiekt/wyr[1]`
 - ▶ `(//obiekt/wyr)[1]`
- ▶ Po każdym predykanie „zebrana” sekwencja, przekazywana do kolejnego predykatu (zmiana kontekstu).
 - ▶ `//obiekt[@nazwa and position() = 5]`
 - ▶ `//obiekt[@nazwa][position() = 5]`

XPath 1.0 – różnice w modelu danych

- ▶ Typy danych:
 - ▶ boolean,
 - ▶ string,
 - ▶ number,
 - ▶ node-set,
 - ▶ w XSLT 1.0: result tree fragment.
- ▶ Brak sekwencji wartości prostych.
- ▶ Zbiory (a nie sekwencje) węzłów.

XPath 1.0 – różnice w języku

- ▶ Brak wyrażenia `if`.
- ▶ Brak wyrażen `for`, `some`, `every`.
- ▶ Brak porównań atomowych i `is`.
- ▶ Mniej testów węzłów:
 - ▶ `node()`
 - ▶ `text()`
 - ▶ `comment()`
 - ▶ `processing-instruction()`,
`processing-instruction(xml-style-sheet)`
 - ▶ `nazwa`, `*`, `pre:*`, `*:nazwa`.
- ▶ Nazwy funkcji nie w przestrzeni nazw.

XQuery – status

- ▶ XQuery 1.0 – rekomendacja (styczeń 2007).
- ▶ Idea XQuery – język zapytań nad XML.
- ▶ Wynik zapytania może być zapisany w postaci XML (serializacja zdefiniowana wspólnie dla XQuery i XSLT 2.0).

XQuery a XPath

- ▶ Model danych, funkcje – wspólne z XPath 2.0.
- ▶ Język zdefiniowany niezależnie.
- ▶ W praktyce rozszerzenie XPath 2.0.

Struktura zapytania XQuery

- ▶ Deklaracje i ciało.
- ▶ Deklaracje:
 - ▶ wersja (nagłówek zapytania / modułu),
 - ▶ import,
 - ▶ flagi i opcje (np. serializacji),
 - ▶ przestrzeń nazw,
 - ▶ zmienna / parametr całego zapytania,
 - ▶ funkcja.

Przykład

```
xquery version "1.0" encoding "utf-8";
declare namespace foo = "http://example.org";
declare variable $id as xs:string external;
declare variable $doc := doc("przyklad.xml");
```

```
$doc//foo:obiekt[@id = $id]
```

Wyrażenie FLWOR

- ▶ Od *For, Let, Where, Order by, Return*.
- ▶ Zamiast `for` z XPath.
- ▶ Wpływy `Select` z SQL :).

Przykład

```
for $obiekt in doc("przyklad.xml")/lista/obiekt
let $pop := $obiekt/preceding-sibling::element()
let $nazwa-pop1 := $pop[1]/@nazwa
where $obiekt/@nazwa
order by $obiekt/@nazwa
return
  <wynik>
    Obiekt o nazwie {xs:string($obiekt/@nazwa)}
    ma {count($pop)} poprzedników.
    Najbliższym poprzednikiem jest obiekt o nazwie
    {xs:string($nazwa-pop1)}.
  </wynik>
```

Konstruktory węzłów – bezpośrednie (*direct*)

Stały element wynikiem zapytania

```
<book isbn="isbn-0060229357">
  <title>Harold and the Purple Crayon</title>
  <author>
    <first>Crockett</first><last>Johnson</last>
    <?cel Wartość?>
    <!--Wszystko jest brane do wyniku-->
  </author>
</book>
```

Konstruktory i wyrażenia – nawiasy klamrowe

```
<wynik>{
  for $el in doc("przyklad.xml")//* return
    <elem głębokość="{count($el/ancestor::node())}">
      Element o nazwie: {name($el)}</elem>
}</wynik>
```

Konstruktory węzłów – obliczane (*computed*)

Ilustracja składni

```
element book {
  attribute isbn {"isbn-0060229357"},
  element {"title"} { "Harold and the Purple Crayon"},
  element author {
    element first { text { "Crockett" } },
    element last {"Johnson" }
    processing-instruction cel { "Wartość" }
    comment { "Wszystko jest brane do wyniku" }
  }
}
```

Przykład zastosowania – nazwa nadawana dynamicznie

```
<wynik>{
  for $el in doc("przyklad.xml")/* return
    element {concat("elem-", name($el))} {
      attribute głębokość {count($el/ancestor::node())},
      text {"Element o nazwie: "},
      text {name($el)}
    }
}
</wynik>
```

Definicje funkcji

Przykład

```
declare function
  local:podwoj($x)
{ 2 * $x };
```

Przykład ze specyfikacją typów

```
declare function
  local:podwoj($x as xs:double)
  as xs:double
{ 2 * $x };
```

Notacja dla typów

- ▶ Informacje o typie możliwe (ale nieobowiązkowe) dla:
 - ▶ zmiennych,
 - ▶ parametrów i wyników funkcji,
 - ▶ także w XSLT.
- ▶ Możliwości:
 - ▶ nazwa typu,
 - ▶ rodzaj węzła | `node()` | `item()` ,
 - ▶ określenie krotności (`?`, `*`, `+`, brak – dokładnie jeden).
- ▶ Przykłady:
 - ▶ `xs:double`
 - ▶ `element()`
 - ▶ `element()*`
 - ▶ `xs:integer?`
 - ▶ `item()+`