

Języki XPath i XQuery

Patryk Czarnik

Instytut Informatyki UW

XML i nowoczesne technologie zarządzania treścią – 2009/10

Wprowadzenie

Status

Model danych XPath

Język XPath od podstaw

Od podstaw

Ścieżki

XPath 1.0

Język XQuery

Struktura zapytania XQuery

Konstruktory węzłów

Funkcje

XPath i XQuery

- ▶ Języki „zapytań” nad dokumentami XML
 - ▶ wygodny wybór określonych węzłów dokumentu,
 - ▶ intuicyjna składnia („ścieżki” jak w drzewie katalogów),
 - ▶ arytmetyka, porównania,
 - ▶ bogaty zestaw funkcji.
- ▶ XPath stosowany w ramach innych standardów
 - ▶ XSLT,
 - ▶ XML Schema,
 - ▶ XPointer.
- ▶ XQuery – samodzielny język
 - ▶ przetwarzanie/wyciąganie danych z dokumentów XML,
 - ▶ XML-owe bazy danych,
 - ▶ tradycyjne bazy danych ze wsparciem dla XML,
 - ▶ ...

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,
 - ▶ związany z XQuery 1.0.

Model danych XPath i XQuery

- ▶ Teoretyczna podstawa standardów XPath, XSLT i XQuery.
- ▶ Abstrakcyjna („po wczytaniu”) postać dokumentu XML.
- ▶ Typy danych oraz rzutowania między nimi.
- ▶ Różny w różnych wersjach XPath:
 - ▶ 1.0 – 4 typy danych, **zbiory** węzłów,
 - ▶ 2.0 (oraz XQuery 1.0) – typy proste XML Schema, **sekwencje** węzłów i wartości prostych.

Dokument XML w modelu XPath

- ▶ Dokument jako drzewo.
- ▶ Uwzględnianie przestrzeni nazw.
- ▶ Możliwość uwzględniania schematu (w wersji 2.0).
- ▶ Rozwinięte sekcje CDATA oraz referencje do encji i znaków.
- ▶ Brak sąsiadujących węzłów tekstowych.
- ▶ Atrybut nie jest dzieckiem elementu.
- ▶ Korzeń, zwany także „węzłem dokumentu”, jest osobnym węzłem, różnym od elementu głównego.

Rodzaje węzłów w XPath

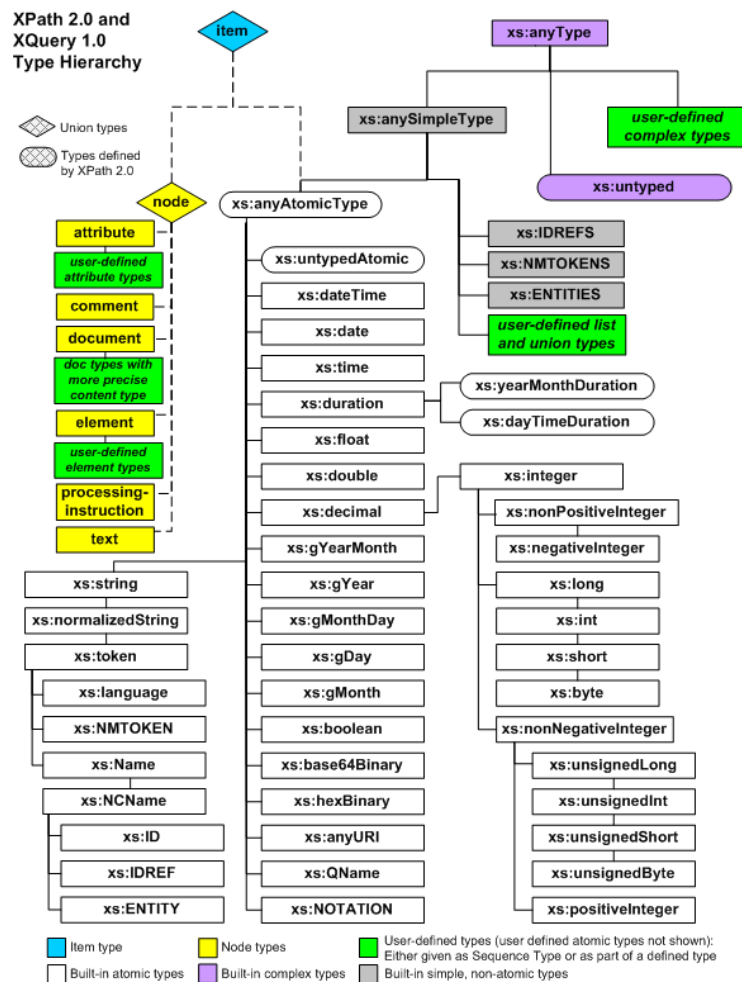
- ▶ Rodzaje węzłów:
 - ▶ węzeł dokumentu (korzeń),
 - ▶ element,
 - ▶ atrybut,
 - ▶ węzeł tekstowy,
 - ▶ instrukcja przetwarzania,
 - ▶ komentarz,
 - ▶ węzeł przestrzeni nazw.
- ▶ Brak m.in.:
 - ▶ sekcji CDATA,
 - ▶ encji i referencji do encji.

Sekwencje

- ▶ Wartości w XPath 2.0 – sekwencje.
- ▶ Elementy (*items*) sekwencji:
 - ▶ węzły (z drzew dokumentów),
 - ▶ 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 pierwotne XML Schema.
- ▶ Dodatkowo:
 - ▶ `xs:untyped`
 - ▶ `xs:untypedAtomic`
 - ▶ `xs:anyAtomicType`
 - ▶ `xs:dayTimeDuration`
 - ▶ `xs:yearMonthDuration`
- ▶ Możliwość używania zdefiniowanych w schemacie (prosty i złożony), a także innych typów wyprowadzonych
 - ▶ o ile przetwarzanie *schema aware*,
 - ▶ nie zawsze dostępne.



XPath 1.0 – różnice w modelu danych

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

Effective Boolean Value

- ▶ Często potrzeba traktowania dowolnej wartości jako wartości logicznej.
- ▶ Zasady zamiany:

pusta sekwencja	→	falsz
sekwencja z węzłem na pierwszej pozycji	→	prawda
pojedyncza wartość boolowska	→	ta sama wartość
pojedynczy pusty napis	→	falsz
pojedynczy niepusty napis	→	prawda
pojedyncza liczba równa 0 lub NaN	→	falsz
inna pojedyncza liczba	→	prawda
inna wartość	→	błąd typu

Atomizacja

- ▶ Operacja służąca traktowaniu dowolnej sekwencji jako sekwencji wartości prostych.
- ▶ Nie zawsze wykonalna (możliwy błąd typów).
- ▶ Dla każdego elementu sekwencji:

wartość atomowa	→	ta wartość
węzeł o (znanym) typie atomowym	→	wartość węzła
węzeł typu lista	→	sekwencja elementów listy
węzeł nieznanego typu prostego, z mieszaną zawartością lub typu <code>xs:untypedAtomic</code> lub <code>xs:anySimpleType</code>	→	zawartość tekstowa jako jeden atom
element o zawartości elementowej	→	błąd typu

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 o nazwie `x`,
- ▶ zmienne wprowadzane konstrukcjami:
 - ▶ XPath 2.0 (`for`, `some`, `every`)
 - ▶ XQuery (FLWOR, `some`, `every`, parametry funkcji)
 - ▶ XSLT 1.0 i 2.0 (`variable`, `param`)

Rzutowanie typów

Konstruktory typów

- ▶ `xs:date("2001-08-25")`
- ▶ `xs:float("NaN")`
- ▶ `adresy:kod-pocztowy("48-200")` (o ile schemat dostępny)
- ▶ `string(//obiekt[4])` (także w XPath 1.0)

Operator `cast as`

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

Funkcje

- ▶ Wywołania funkcji:
 - ▶ `fn:concat('Pani ', imię, ' ', nazwisko)`
 - ▶ `count(//obiekt/@parzysty)`
 - ▶ `moje:silnia(12)`
- ▶ 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).

Wybrane funkcje XPath

Napisy

concat(s1, s2, ...) substring(s, pos, len)
starts-with(s1, s2) contains(s1, s2)
string-length(s) translate(s, t1, t2)

Węzły

name(n?) local-name(n?) namespace-uri(n?)
id(s) nilled(n?) document-uri(doc)

Sekwencje

count(S) sum(S) min(S) max(S) avg(S)
empty(S) reverse(S) distinct-values(S)

Liczby

floor(x) ceiling(x)
round(x) abs(x)

Kontekst

current() position()
last() current-time()

Data i czas

month-from-date(t)
adjust-date-to-
timezone(t, tz)

Operatory

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

Operatory porównania

Porównania atomowe (tylko XPath 2.0)

- ▶ eq ne lt le gt ge
- ▶ na wstępie argumenty poddane atomizacji
- ▶ jeden z argumentów sekwencją pustą → wynik sekwencją pustą
- ▶ jeden z argumentów sekwencją wieloelementową → błąd typów,
- ▶ typy atomowe – intuicyjnie „normalne” porównanie,
- ▶ o ile typy pozwalają.

Porównania ogólne (XPath 1.0 i 2.0)

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

Porównania ogólne – ciekawostki

(Nie)Równość nie jest (nie)równością sekwencji

$(1, 2) = (2, 3)$ – prawda
 $(1, 2) != (1, 2)$ – prawda

Równość nie jest przechodnia

$(1, 2) = (2, 3)$ – prawda
 $(2, 3) = (3, 4)$ – prawda
 $(1, 2) = (3, 4)$ – fałsz

$x != y$ nie jest równoważne $\text{not}(x = y)$

$(1, 2) = (1, 2)$ – prawda
 $(1, 2) != (1, 2)$ – prawda
 $() = ()$ – fałsz
 $() != ()$ – fałsz

Wyrażenie warunkowe (XPath 2.0)

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

- ▶ Liczy się *Effective Boolean Value*.
- ▶ Obliczana tylko jedna gałąź.

Przykład

```
if informacje/cena
then
  if informacje/cena >= 1000
  then 'Przesyłka wartościowa'
  else 'Przesyłka zwykła'
else 'Brak danych'
```

Pętla po sekwencji (XPath 2.0)

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

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

Przykłady

```
for $i in (1 to 10)
  return $i * $i
```

```
for $o in //obiekt
  return concat('Nazwa obiektu:', $o/@nazwa)
```

Kwantyfikatory (XPath 2.0)

```
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.

Przykłady

```
some $i in (1 to 10) satisfies $i > 7
```

```
every $o in //obiekt satisfies $o/@nazwa
```

Ścieżki w XPath

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,
- ▶ *predykat* – opcjonalny, dodatkowo filtrujący węzły.

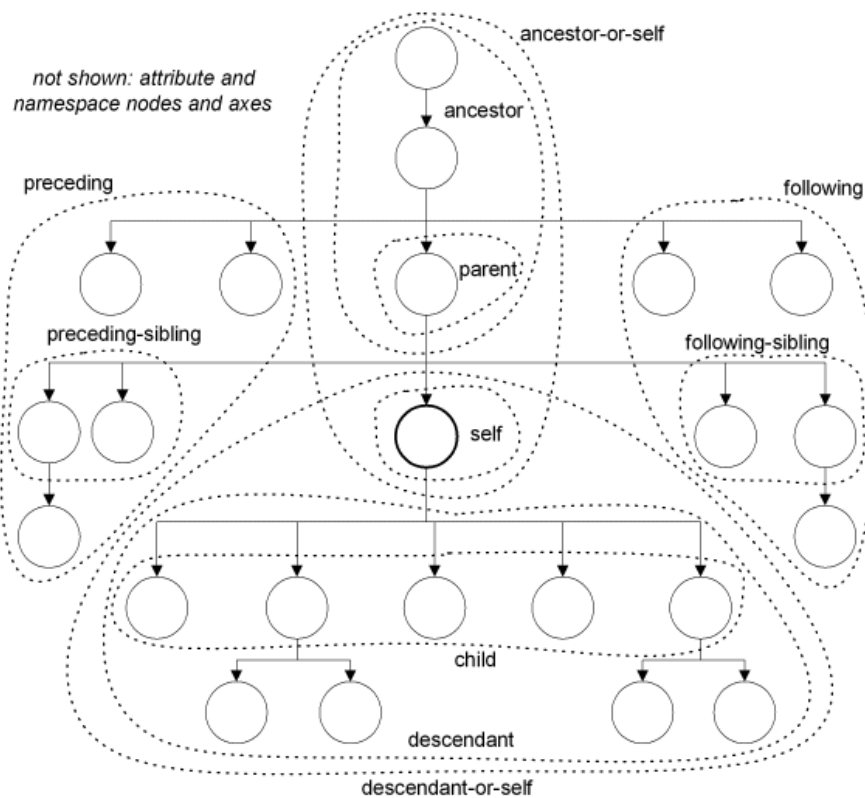
Przykład

```
/descendant::dział[attribute::id = 'ksi']/child::pracownik[1]  
/child::nazwisko/child::text()
```

Osie

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

Osie



Testy węzłów w XPath 1.0

Rodzaj węzła

- ▶ `node()`
- ▶ `text()`
- ▶ `comment()`
- ▶ `processing-instruction()`
- ▶ `processing-instruction(xml-style-sheet)`

Nazwa węzła (elementu bądź atrybutu, zależnie od osi)

- ▶ `osoba`
- ▶ `*`
- ▶ `pre:*`
- ▶ `*:osoba`

Dodatkowe testy węzłów w XPath 2.0

W XPath 2.0 dodatkowo

- ▶ `document-node()`
- ▶ `element()`
- ▶ `element(osoba)`
- ▶ `element(*, osobaTyp)`
- ▶ `element(osoba, osobaTyp)`
- ▶ `attribute()`
- ▶ `attribute(id)`
- ▶ `attribute(*, xs:integer)`
- ▶ `attribute(id, xs:integer)`

Predykaty

- ▶ Obliczane dla każdego węzła (węzeł na chwilę staje się węzłem kontekstowym).
- ▶ Każdy predykat „przesiewa” sekwencję.
- ▶ W zależności od typu wyniku predykatu:
 - ▶ liczba – porównywana z pozycją węzła w sekwencji (od 1),
 - ▶ nie liczba – decyduje *Effective Boolean Value*.
- ▶ Możliwe użycie poza ścieżkami (tzw. *filter expressions*).

Przykłady

```
/child::osoby/child::osoba[child::imię = 'Patryk']  
child::osoba[child::imię = 'Patryk']/child::nazwisko  
//obiekt[attribute::nazwa][3]  
(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

```
.///obiekt[@id = 'E4']  
self::node()/descendant-or-self::node() /  
    child::obiekt[attribute::id = 'E4']
```

Sposób obliczania ścieżek

- ▶ Ścieżki obliczane od lewej do prawej.
- ▶ Dla każdego węzła z bieżącej sekwencji obliczany kolejny krok (wraz z predykatami).
 - ▶ `//oddział/pracownik[1]`
 - ▶ `(//oddział/pracownik)[1]`
- ▶ Po każdym predykanie zbierana cała sekwencja, przekazywana do kolejnego predykatu (zmiana kontekstu).
 - ▶ `//pracownik[@kierownik and position() = 5]`
 - ▶ `//pracownik[@kierownik][position() = 5]`

XPath 1.0 – ograniczenia

- ▶ Prostszy model danych:
 - ▶ 4 (w XSLT 5) typy danych zamiast typów XML Schema,
 - ▶ zbiory węzłów zamiast sekwencji wszystkiego.
- ▶ Brak wyrażenia `if`.
- ▶ Brak wyrażeń `for`, `some`, `every`.
- ▶ Brak porównań atomowych `iis`.
- ▶ Mniej testów węzłów.
- ▶ Mniej standardowych funkcji.

XQuery – język zapytań nad XML

Status

- ▶ XQuery 1.0 – rekomendacja (styczeń 2007).

Niektóre możliwości

- ▶ Wyciąganie danych z dokumentów XML.
- ▶ Sortowanie, grupowanie, ...
- ▶ Zapis wyniku jak w XSLT (XML / HTML / tekst).
- ▶ Konstruowanie nowych węzłów.
- ▶ Definiowanie własnych funkcji.

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.
- ▶ Jak `SELECT` w 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 elementu 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 2.0.
- ▶ Możliwości:
 - ▶ nazwa typu,
 - ▶ rodzaj węzła | `node()` | `item()` ,
 - ▶ określenie krotności (? , * , + , brak – dokładnie jeden).
- ▶ Przykłady:
 - ▶ `xs:double`
 - ▶ `element()`
 - ▶ `node()*`
 - ▶ `xs:integer?`
 - ▶ `item()+`