XPath, XQuery

Mladen Vidović
mvidovic@singidunum.ac.rs

Univerzitet Singidunum
Centar Novi Sad

31. mart 2022.

Sadržaj

XPath

- Jezik za označavanje delova XML dokumenata.
- Navigacija kroz stablo dokumenta.

XPath

- U stablu dokumenta postoji nekoliko različitih čvorova:
  - dokument (koren)
  - element
  - atribut
  - tekst
  - namespace
  - procesna instrukcija
  - komentar

XPath

- Primer: dokument sa kolekcijom autora

```
<authors>
  <author><name>...</name></author>
  ...
</authors>
```

XPath

- korenski čvor je document čvor
- čvor koji je njegovo dete je authors element
- authors čvor ima nekoliko dece –author
- author čvorovi su siblings –imaju istog roditelja i na istom su nivou (brat/sestra)
- authors je roditelj čvora author
- name je potomak za authors
- authors je predak za name

XPath

- XPath izrazi se pišu kao putanje: /korak/korak/korak...
- Ukoliko putanja počinje sa /, onda je apsolutna, odnosno kreće od document čvora, u suprotnom se evaluaira u odnosu na tekući čvor

XPath

- Opšti oblik izraza je: osa::test[predikat]
- Osa je osa kretanja (potomak, dete, roditelj...)

XPath

- Primer: XPath putanja koja pronalazi sve author potomke čvora authors - child::author
- U Oxygen-u uključiti view za XPath/XQuery
- Window->Show View-> XPath/XQuery builder
- Ako samo pokušamo child::author, ne dobijamo ništa, zato što document čvor nema dete author.

XPath

- Xpath putanja koja pronalazi sve elemente koji imaju verified atribut (nezavisno od lokacije)
- descendant::*/attribute::verified

XPath

- Xpath putanja koja pronalazi sve elemente koji imaju verified atribut (nezavisno od lokacije)
- descendant::*/attribute::verified
- Šta zapravo dobavlja ovaj izraz?
- Svi name elementi koji su unuci tekućeg čvora
- child::*/child::name

XPath - ose

- Ose:
  - ancestor : predak
  - ancestor-or-self : predak ili tekući čvor
  - attribute : atributi
  - child : deca
  - descendant : potomci, naslednici
  - descendant-or-self : naslednici ili tekući čvor
  - following : sve nakon zatvarajućeg taga
  - following-sibling : sva braća/sestre nakon zatvarajućeg taga tekućeg čvora
  - namespace : svi namespace čvorovi tekućeg elementa
  - parent : roditelj
  - preceding : sve pre otvarajućeg taga
  - preceding-sibling : sva braća/sestre pre tekućeg
  - self : tekući čvor

XPath - skraćeni zapis

- Ako se ne zapiše osa, podrazumevano je child::
- self .
- parent ..
- descendant //
- attribute @
- following-sibling ../

XPath

- Predikati - zadavanje uslova za pronalaženje čvora
- Pišu se unutar uglastih zagrada []
- //author[@verified="true"]
- //authors[attribute::verified]
- //author[yearOfBirth>1960]

XPath - funkcije

- Za dobavljanje određenog elementa iz sekvence, može se koristiti position(), last() ili indeksiranje pozicije
- Pretposlednji element //book[last()-1]
- Za proveru tekstualnog sadržaja koristi se text() funkcija
- //author/name[text()='Dan Abnett']
- predikat ne mora da označava kraj xpath-a

Zadaci

- U authors.xml koristeći xpath dobaviti:
- žanrove svih verifikovanih autora,
- države porekla svih autora rođenih pre 1960. godine,
- imena svih autora koji pišu dela za Warhammer setting,

Zadaci

- U books.xml pronaći:
- sve nemačke naslove knjiga
- imena svih autora knjiga koje imaju nemačke naslove
- cene svih knjiga kojih ima više od 15 na lageru
- broj knjiga na lageru za sve knjige koje koštaju više od 20 EUR

XPath

- Moguće je obuhvatiti rezultat selekcije jednim predikatom, tako što se selekcija obuhvati zagradama ()
- (//title[@lang='english'])[last()]
- Takođe je moguće kombinovati predikate sa and i or

Zadaci

- U books.xml pronaći:
- sve knjige kojih ima više od 20 na lageru i koštaju manje od 20
- poslednju navedenu knjigu koju je napisao Sandy Mitchell
- engleski naslov prve knjige koju je napisao Dan Abnett i čija je cena veća od 25 USD

Matches

- Koristeći matches(input, regex) funkciju u predikatu moguće je vršiti proveru poklapanja rezultata sa regularnim izrazom
- //book[title[matches(text(), "lambs$")]]
- konačan input koji se prosledi matches funkciji sme da sadrži samo 1 element, tako da je neispravno:
- //book[matches(title/text(), "lambs$")] - title/text() za neke knjige vraća 2 elementa, jer imaju više naslova

Sadržaj

XQuery

- Upitni jezik za XML
- Svaka vrednost je sekvenca
- Elementi sekvence su singletoni, odnosno sekvence dužine 1
- Sekvence mogu biti prazne, ali ne ugnježdene

XQuery

```
<html><body>
{
for $b in doc("books.xml")//book
return
<div>
<h1>Books</h1>
<div>
{$b//title[1]/text()} written by
{$b/author/name/text()}
</div>
</div>
}
</body></html>
```

XQuery

- html i body tagovi su statički
- for $b in doc("books.xml")//book - za svaku knjigu, kojoj će se pristupati preko promenljive b u sekvenci koju dobijamo tako što učitamo xml dokument books.xml i zatim dobavimo sve book elemente
- vraćamo (u petlji) divove koji sadrže po naslov i div
- Telo ovih divova stavljamo u vitičaste zagrade, da bi se evaluirali (interpolirali)
- U telu se pristupa promenljivoj koja predstavlja knjigu i dobavlja se tekst prvog naslova, pa se nadovezuje sa rečima written by i zatim ponovo interpolira ime autora knjige

XQuery

- Jednostavan primer:
  (: komentar :)
  let $i := 42
  return $i + 10
- Query može da ima prolog u kojem se definišu funkcije, globalne promenljive i namespace-ovi
- Query mora da ima telo u kojem se vrši neki upit i vraća neka vrednost, odnosno mora da ima FLWOR izraz

XQuery

- declare namespace ns = "foo";
  declare function ns:increment ($x as xs:integer) as xs:integer
  {
  if($x > 0)then $x + 1
  else $x -1
  };
  for $i in (1, 2, -1, -5, 16)
  return ns:increment($i)

XQuery FLWOR

- for - iteracija kroz sekvencu - for $var in sequence
- let - definisanje promenljivih - let $var := val
- where - filtriranje rezultata (predikat) - where $var > $x
- order by - sortiranje - order by $var/text() ascending (descending)
- return - konstrukcija rezultata - return $var
- čita se kao flower

XQuery FLWOR

- Jedan potpuni FLWOR izraz:
- for $b in doc("books.xml")//book
  let $row := concat($b/title[1], ', ', $b/author/name)
  where $b/author/name = "Dan Abnett"
  order by $b/title[1] descending
  return

  <knjiga jezik="{$b/title[1]/@lang}">
  {$row}
  </knjiga>

Zadaci

- Napisati XQuery za dobavljanje
- Svih nemačkih naslova knjiga, rezultate vratiti kao neuređenu listu
- Svih knjiga čija je cena veća od 20 eur, sortirano prema imenu autora
- Svih knjiga čija je cena veća od 20 usd, kojih ima više od 10 na lageru i imaju nemački naslov.

Kvantifikatori

- some - važi za barem 1 element sekvence
- every - važi za sve elemente sekvence
- Skraćeni flwor izraz
- every $b in doc("books.xml")//book
  satisfies $b/price > 20
- some $b in doc("books.xml")//book satisfies $b/@available> 20

Zadaci

- Kreiranjem 'inner join' operacije povezati books.xml i authors.xml i napisati XQuery za:
- Prikaz svih knjiga zajedno sa podacima njihovih autora
  Naslov i ISBN knjige, ime, godina rođenja i glavni (prvi navedeni) žanr autora
- Prikaz svih autora koji pišu science fiction knjige zajedno sa spiskom knjiga koje su napisali
  Ime i godina rođenja autora, engleski naslov knjige

XQuery

- Aritmetičke operacije:
- +, -, *, div, mod
- min, round, round-half-to-even
- Poređenje singleton-a
- eq, ne, gt, ge, lt, le
- Poređenje dve sekvence
- =, !=

XQuery - poređenje

- Kod poređenja sekvenci se primenjuje kvantifikator 'some' a ne 'every'
- (1, 2, 3) = 4 -> false
- (1, 2, 3) = 3 -> true
- (1, 2) = (3, 4) -> false
- (1, 2) != (3, 4) -> true
- (1, 2) = (2, 3) -> true
- (1, 2) != (2, 3) -> true
- (1, 2) != (1, 2) -> ?

Xquery - if

- if - then - else konstrukcija
- svi elementi su obavezni, ali ako else nije potreban, koristi se else()
- Za sve knjige, ispisati engleski naslov ukoliko je knjiga jeftinija od 20EUR a nemački ako je skuplja

Xquery - if

```
for $b in doc("books.xml")//book
return
if($b/price[@currency="EUR"] > 20)
then <p>$b/title[@lang="english"]</p>
else <p>$b/title[@lang="german"]</p>
```

Zadaci

- Vratiti uređenu listu svih autora
- Za svakog autora vratiti ime i zemlju porekla (odvojeno zarezom), a ispod toga novu uređenu listu
- Nova lista sadrži naslov knjige i broj dostupnih primeraka, odvojene zarezom -naslov, available:20
- Ukoliko knjiga ima nemački naslov, njega ispisati u zagradi - naslov (ger: naslov), available 10

JSON

- JavaScript Object Notation
- String reprezentacija objekata (tekstualni prikaz)
- Popularno rešenje za serijalizaciju objekata za razmenu i skladištenje

JSON

- Elementi JSON dokumenta
- objekti
- nizovi
- stringovi
- brojevi
- boolean vrednosti
- null

JSON Schema

- Služi za definisanje strukture JSON dokumenta
- Omogućuje validaciju JSON-a naspram šeme koja mu je dodeljena
- Specifikacija dostupna na
  `https://json-schema.org/specification.html`

JSON Schema

- File -> New -> JSON Schema
- $schema - dokument kojim je definisana šema
- $id - URI šeme koju definišemo
- title - naslov šeme
- description - opis šeme
- type - ograničenje tipa za samu šemu, mora da bude objekat
- properties - svojstva koja će biti definisana u šemi

JSON Schema

- Uplatnica ima "globalni element"uplatnica
- Podelementi su joj podaciODuzniku, podaciOPoveriocu, svrhaPlacanja, podaciOUplati
- svrha plaćanja je string, ostali su object
- svrha plaćanja ima minimalnu dužinu 1, a maksimalnu dužinu 200 karaktera
- u svrhaPlacanja dodajemo svojstva minLength i maxLength
- postaviti da su sva ova svojstva obavezna

JSON Schema

- Podaci o poverioci i uplatiocu mogu da se odnose na pravno ili na fizičko lice
- Oba imaju adresu, koja se sastoji iz ulice, broja, grada i države
- Adresa je obavezna
- Ako je u pitanju fizičko lice, ima ime i prezime
- Ako je u pitanju pravno lice, ima naziv

JSON Schema

- Jedan način kako možemo da ostvarimo izbor između fizičkog ili pravnog lica je da prvo navedemo sva svojstva
- Nakon toga, koristeći oneOf svojstvo definišemo izbor - ili je obavezno ime i prezime, ili je obavezan naziv
- oneOf - tačno jedan od navedenih uslova mora biti ispunjen
- allOf - svi navedeni uslovi moraju biti ispunjeni
- anyOf - barem jedan od uslova mora biti ispunjen

JSON Schema

- Drugi način podrazumeva da definišemo odvojene JSON šeme za fizičko i za pravno lice
- Zatim za opis podataka o uplatiocu/poveriocu koristimo $ref: $id eksterne šeme
- Takođe koristeći oneOf možemo da tako omogućimo da bude ili fizičko ili pravno lice
- Nije neophodno napraviti odvojenu šemu za definisanje tipova
- Moguće je i koristiti svojstvo definitions i referencirati ga

JSON Schema

- Dodati u uplatnicu podaciORacunu sa podelementima:
- brojRacuna (string 3 cifre - 13 cifara - 2 cifre)
- model (integer)
- pozivNaBroj (string, max 20 karaktera)

JSON Schema

- podaciOUplati:
- iznos (decimal 2 decimalne cifre) - definisano preko multipleOf svojstva (deljivo sa) 0.01
- valuta (string, enumeracija RSD, USD i EUR) - definisano preko enum svojstva čija je vrednost niz dozvoljenih vrednosti

JSON Schema

- Pri validaciji sa ovom šemom, ako JSON ima još neka dodatna svojstva, validacija će proći
- Da bismo to izbegli, odnosno zabranili dodavanje dodatnih svojstava, na nivoima gde želimo to da zabranimo postavimo svojstvo additionalProperties na false

JSON Schema

- Kreirati JSON šemu za ličnu kartu (V1)

JSON Schema

- Kreirati JSON šemu za knjižaru.
- Knjižara ima naziv, adresu (ulica, broj, grad, država), vlasnika (ime, prezime, adresa) i kolekciju knjiga
- Svaka knjiga ima naslov (string, max 100 karaktera), autora(ime, prezime), broj stranica(broj >0), godinu izdavanja(broj) i žanr (string, enumeracija), isbn (string, 3 cifre - 1 cifra - 2 cifre - 6 cifara - 1 cifra)
- Za definisanje tipa za knjigu, pogledati `https://json-schema.org/understanding-json-schema/structuring.html`

JSON Schema

- Kreirati JSON šemu za kolekciju knjižara