Poizvedovanje po XML dokumentih

Uvod v XQuery

Uvod v XQuery

- **Poizvedovalni jezik**, ki nam omogoča:
  - izbiro elementov/atributov iz XML dokumentov,
  - združevanje podatkov iz več dokumentov,
  - spreminjanje podatkov,
  - izpeljava novi podatkov,
  - sortiranje rezultatov,
  - ipd.

XQuery primer

vhodni dokument

```
<narocilo st="00299432" datum="2004-09-15" stranka="0221A">
   <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="rjava" />
   <postavka oddelek="ACC" stevilka="563" kolicina="1" />
   <postavka oddelek="ACC" stevilka="443" kolicina="2" />
   <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="modra" />
   <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="rdeča" />
   <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="vijolična" />
</narocilo>
```

poizvedba

```
for $oddelek in distinct-values(doc("narocila.xml")//postavka/@oddelek
let $postavke := doc("narocila.xml")//postavka[@oddelek = $oddelek]
order by $oddelek
return <oddelek ime="{$oddelek}" skupnaKolicina="{sum($postavke/@kolicina)}">
```

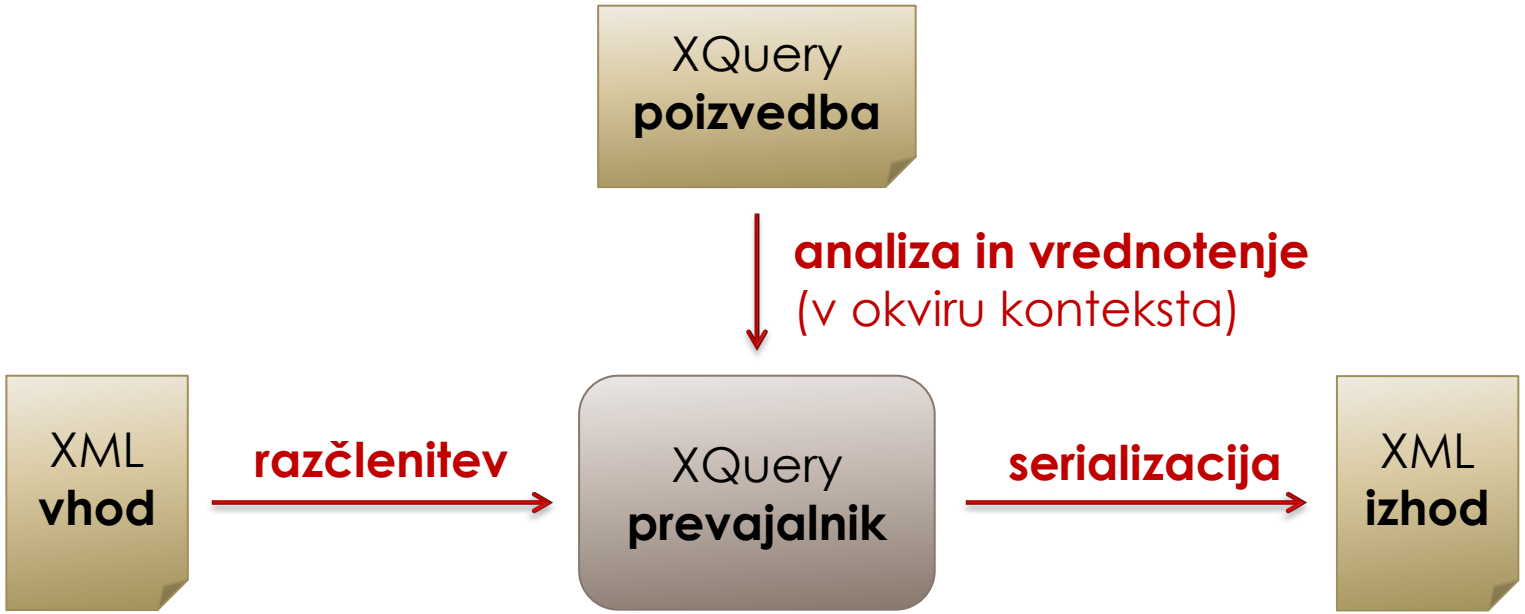
rezultat

```
<oddelek ime="ACC" skupnaKolicina="3" />
<oddelek ime="MEN" skupnaKolicina="2" />
<oddelek ime="WMN" skupnaKolicina="2" />
```

Scenariji uporabe

- izpostavljanje podatkov v obliki **spletnih storitev**,
- **iskanje** podatkov na spletu **in združevanje** z obstoječimi podatki,
- **pretvorba** XML **v XHTML** za objavo na spletu,
- itd.

XQuery model procesiranja



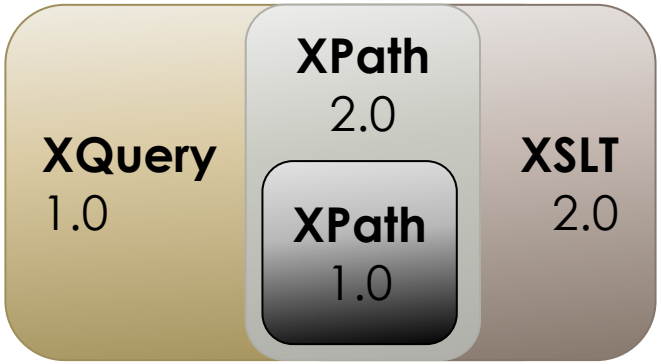
XML vhod

- lahko sprejme:
  - tekstovne XML dokumente,
  - množico XML dokumentov na URI naslovu,
  - podatke iz XML podatkovne baze,
  - podatki iz relacijske podatkovne baze z XML dostopno točko,
  - XML dokumente shranjene v pomnilniku.

XQuery, XSLT in XPath

FLWOR izrazi, XML konstruktorji, poizvedovalni prolog, uporabniško določene funkcije

pogojni izrazi, aritmetični izrazi, vgrajene funkcije in operatorji, podatkovni model



stili, predloge ipd.

omejitev poti, primerjalni izrazi, omejen nabor vgrajenih funkcij

Poizvedovanje po XML dokumentih

Testni XML dokumenti

Katalog izdelkov (katalog.xml)

```
<katalog>
   <izdelek oddelek="WMN">
      <stevilka>557</stevilka>
      <ime jezik="sl">lanena srajca</ime>
      <barve>rjava modra</barve>
   </izdelek>
   <izdelek oddelek="ACC">
      <stevilka>563</stevilka>
      <ime jezik="sl">kavbojski klobuk</name>
   </izdelek>
   …
   <izdelek oddelek="MEN">
      <stevilka>784</stevilka>
      <ime jezik="sl">kratka majica</ime>
      <barve>modra/bela modra/rdeča</barve>
      <opis>Naša <i>najbolje prodajana</i> majica!</opis>
   </izdelek>
</katalog>
```

Cene (cene.xml)

```
<cene>
   <cenik datumOd="2004-11-15">
      <izd st="557">
         <cena valuta="EUR">29.99</cena>
         <popust tip="CLR">10.00</popust>
      </izd>
      <izd st="563">
         <cena valuta="EUR">69.99</cena>
      </izd>
      <izd st="443">
         <cena valuta="EUR">39.99</cena>
         <popust tip="CLR">3.99</popust>
      </izd>
   </cenik>
</cene>
```

Naročilo (narocilo.xml)

```
<narocilo st="00299432" datum="2004-09-15" stranka="0221A">
   <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="rjava" />
   <postavka oddelek="ACC" stevilka="563" kolicina="1" />
   <postavka oddelek="ACC" stevilka="443" kolicina="2" />
   <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="modra" />
   <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="rdeča" />
   <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="vijolična" />
</narocilo>
```

Poizvedovanje po XML dokumentih

Enostavne poizvedbe

Dostop do strežnika

Na spletni učilnici so navodila za namestitev XML podatkovne baze **eXist** v okolje C9.

http://{C9 ime aplikacije}-{C9 uporabniško ime}.c9users.io:8080/exist/apps/eXide/

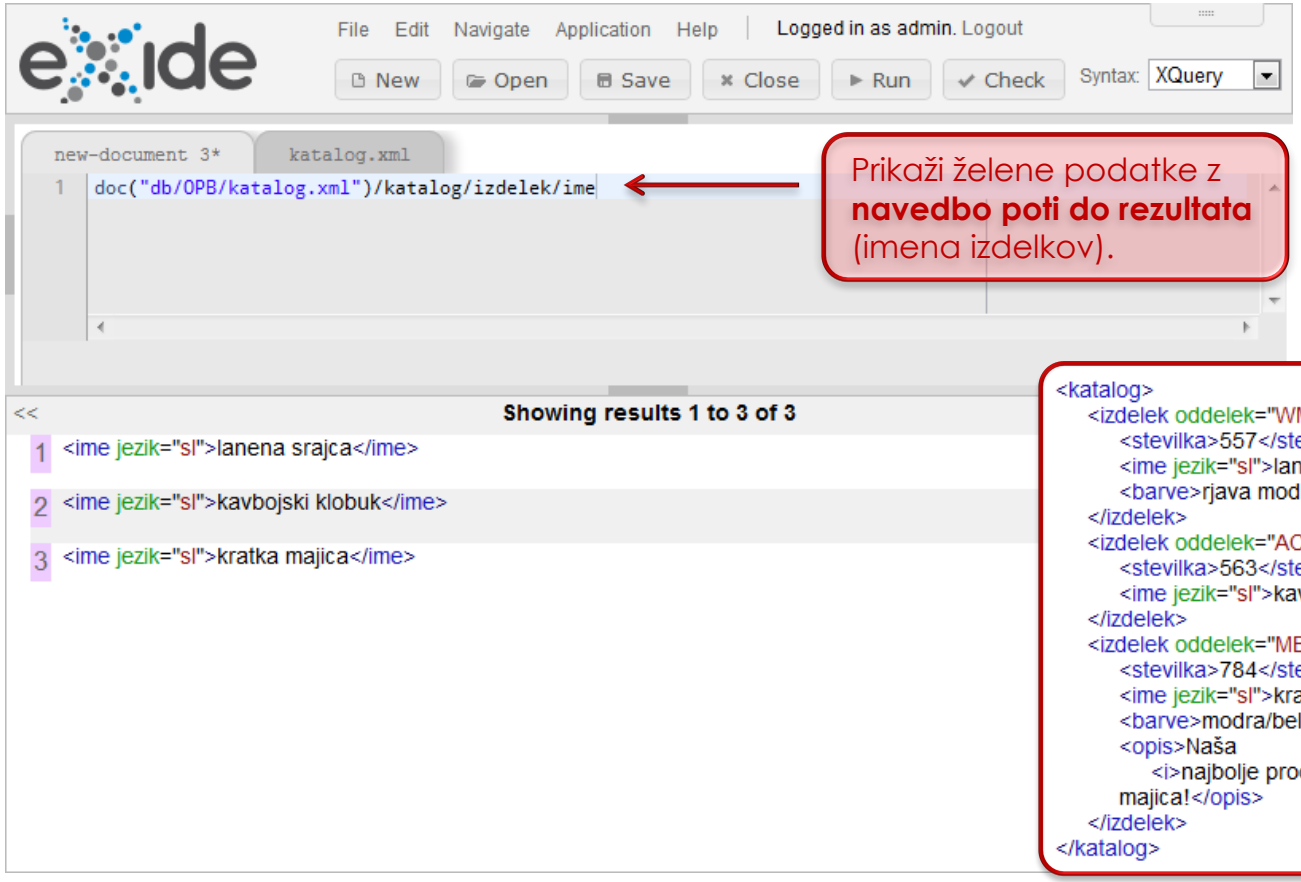
Ko boste namestitev uspešno izvedli, bo strežnik na voljo na zgornjem naslovu!

Izvajanje poizvedb (1)

Odpri XML datoteko

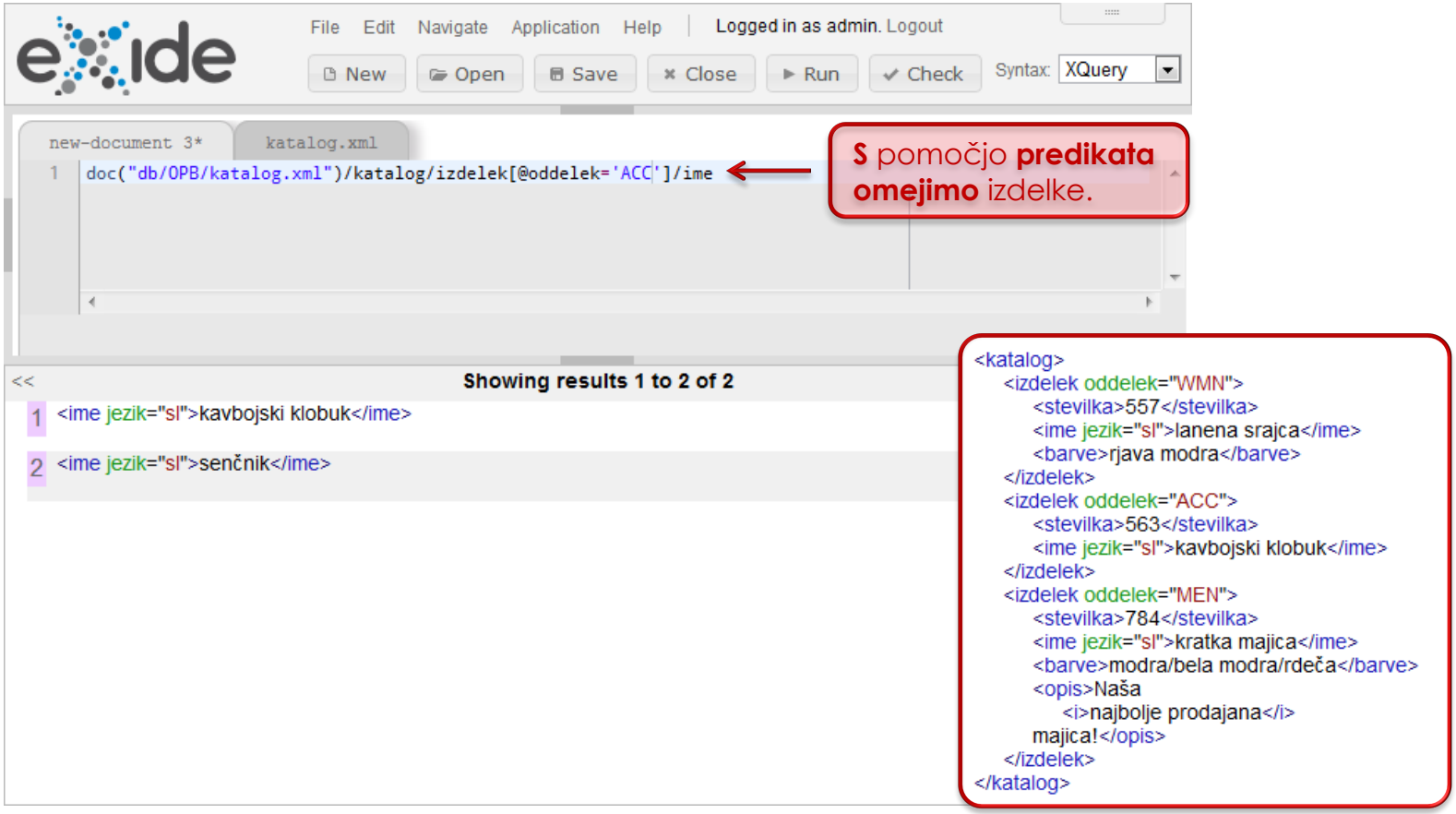
Izvajanje poizvedb (2)

XPath

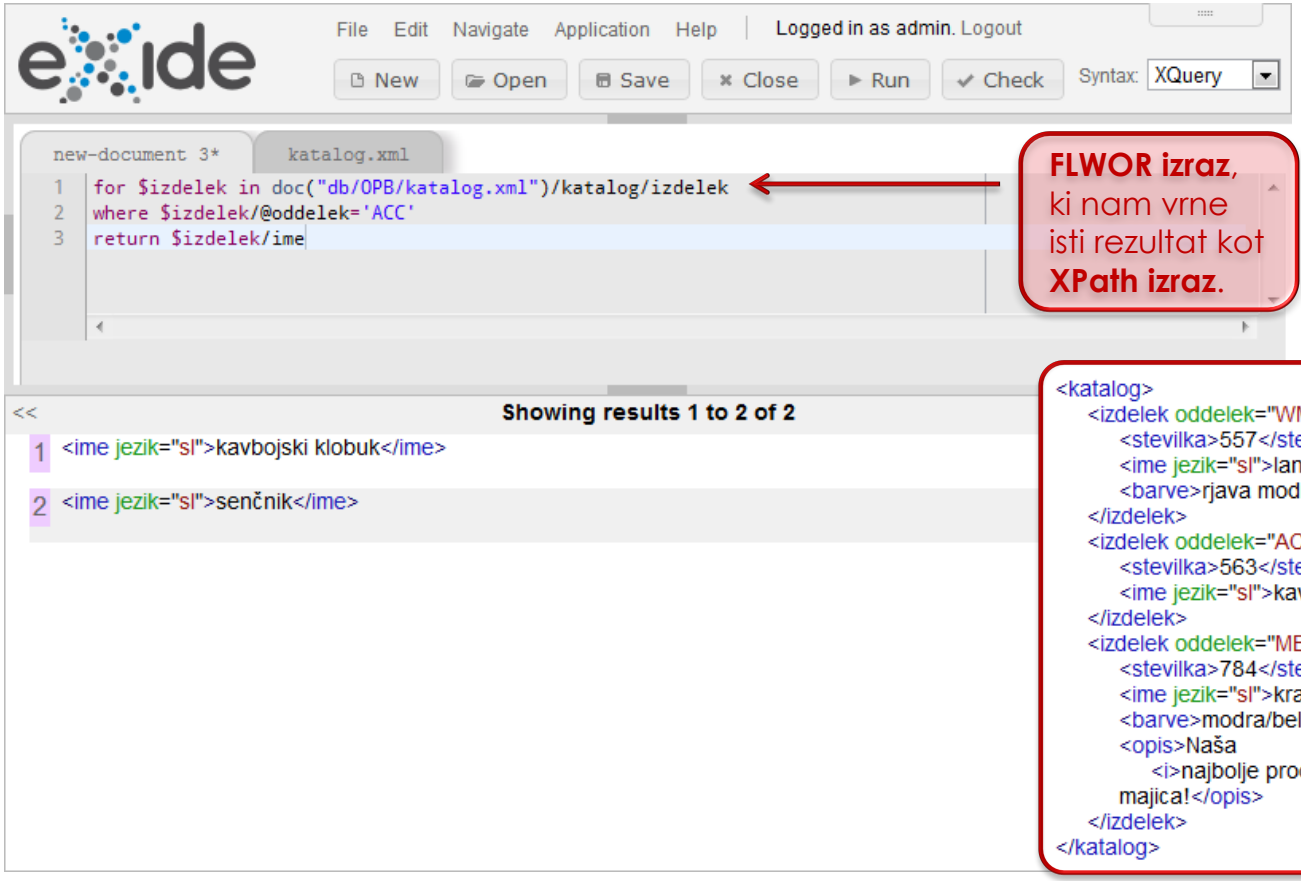


Izvajanje poizvedb (3)

XPath

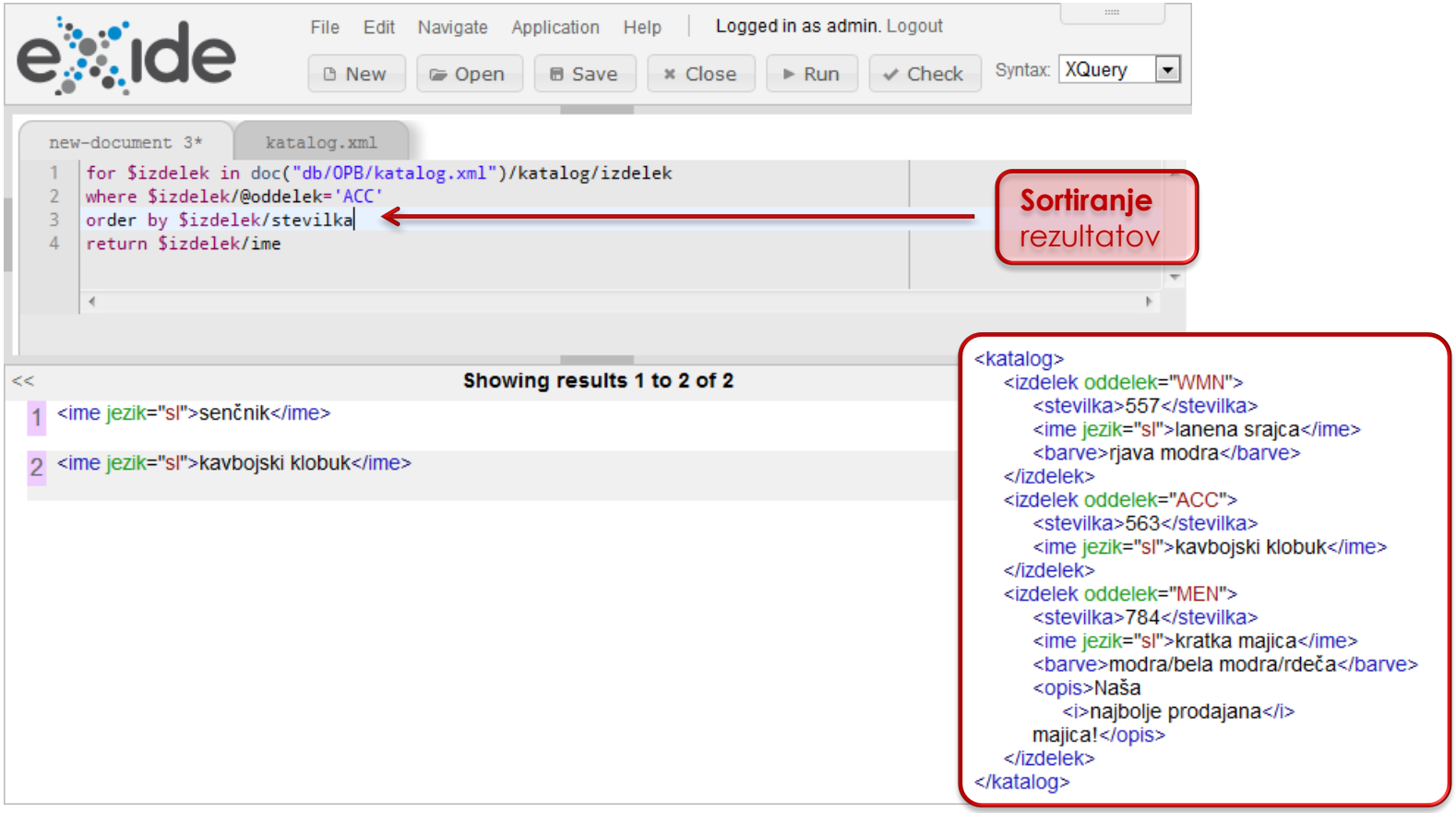


Izvajanje poizvedb (4)
FLWOR



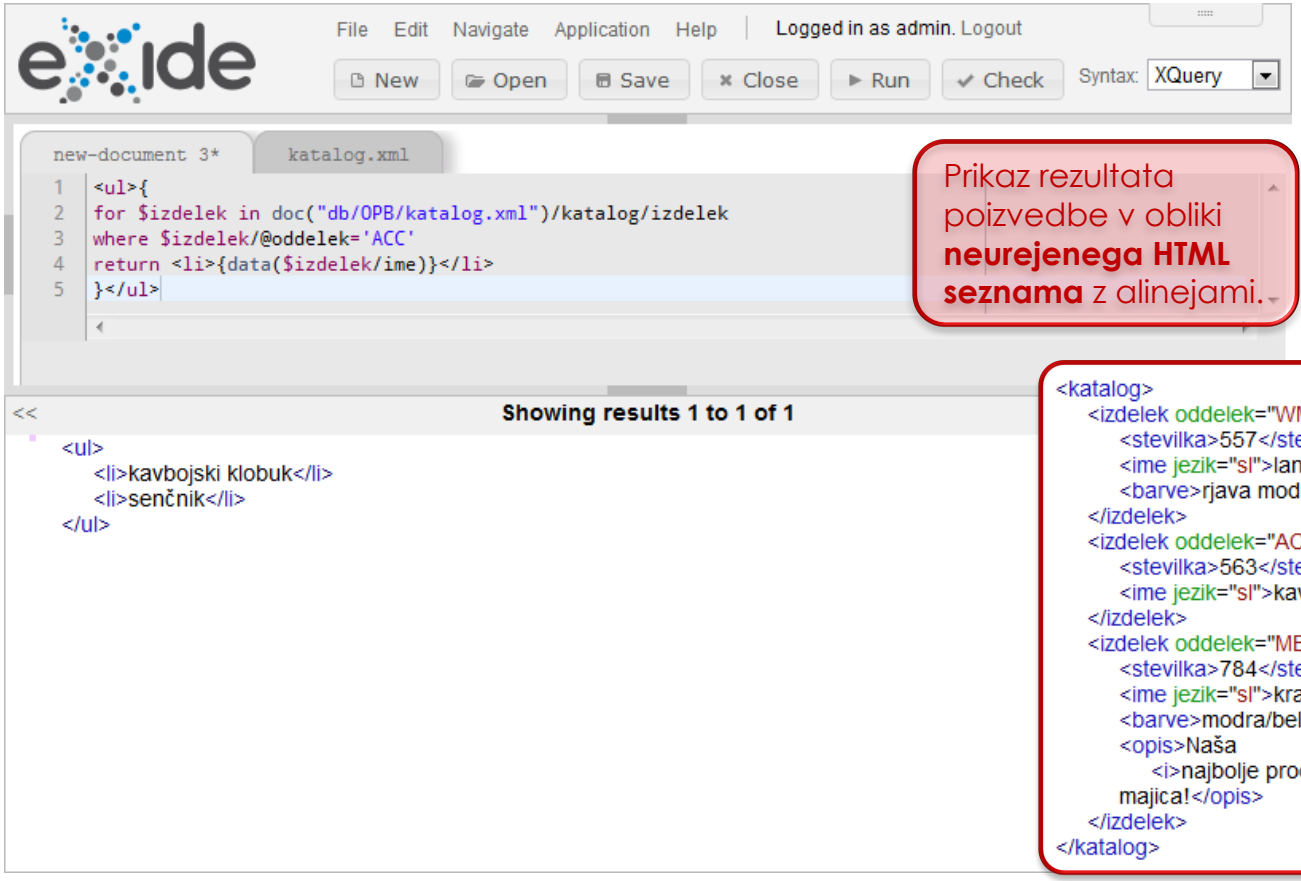
Izvajanje poizvedb (5)

Sortiranje rezultatov



Izvajanje poizvedb (6)

Pretvorba v HTML



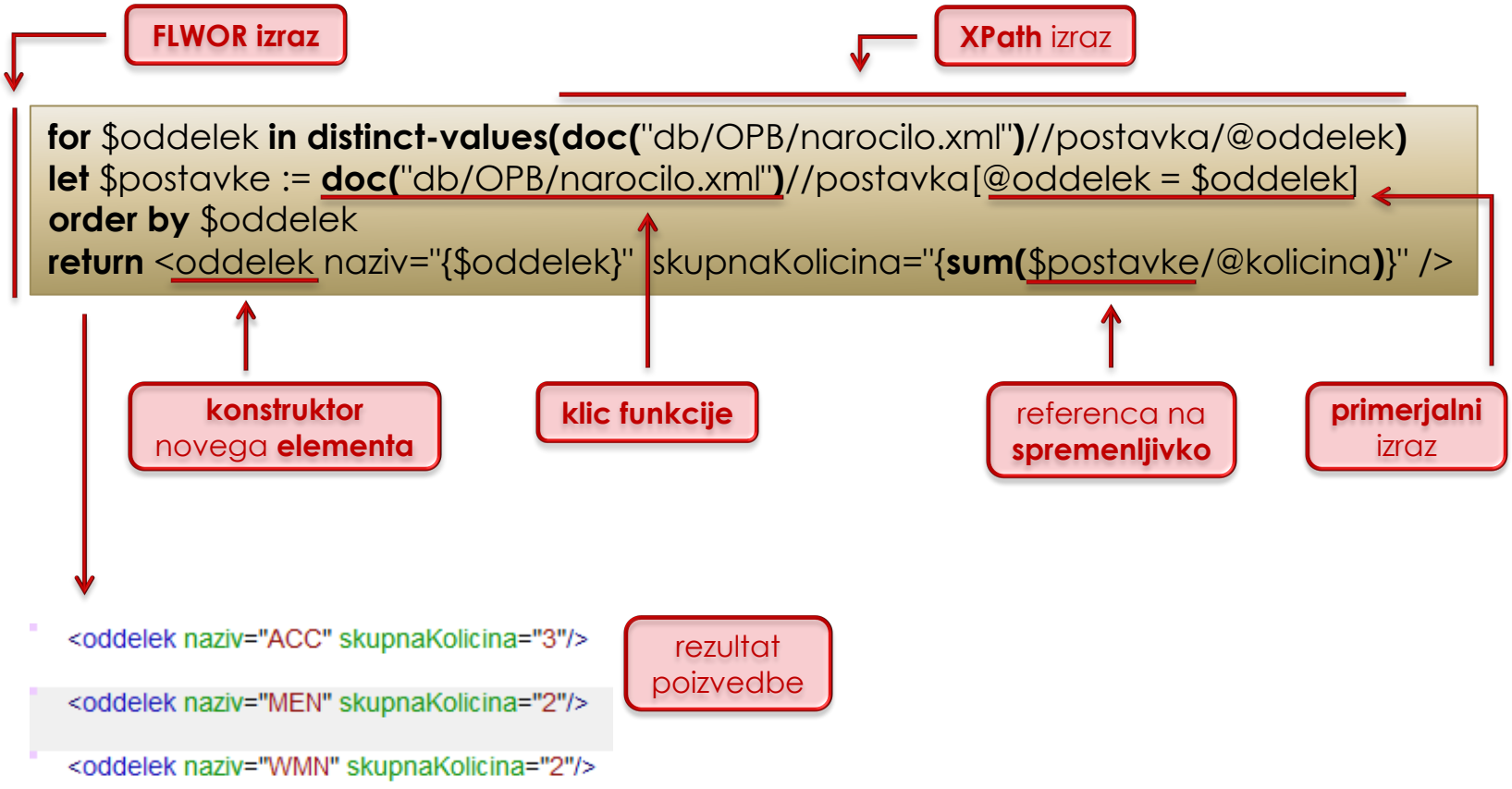
Prikaz rezultata poizvedbe v obliki **neurejenega HTML seznama** z alinejami.

```
<katalog>
    <izdelek oddelek="WMN">
        <stevilka>557</stevilka>
        <ime jezik="sl">lanena srajca</ime>
        <barve>rjava modra</barve>
    </izdelek>
    <izdelek oddelek="ACC">
        <stevilka>563</stevilka>
        <ime jezik="sl">kavbojski klobuk</ime>
    </izdelek>
    <izdelek oddelek="MEN">
        <stevilka>784</stevilka>
        <ime jezik="sl">kratka majica</ime>
        <barve>modra/bela modra/rdeča</barve>
        <opis>Naša
            <i>najbolje prodajana</i>
        majica!</opis>
    </izdelek>
</katalog>
```

Osnove jezika XQuery

Osnovni gradniki, sintaksa, spremenljivke, klici funkcij, primerjave, prolog

Osnovni gradniki XQuery izraza



Vrstni red računanja vrednosti

- Pri izračunavanju vrednosti je pomemben vrstni red:
  - **and** ima prednost pred **or**

    true() **and** true() **or** false() **and** false() → **true**
  - uporabimo lahko oklepaje

    true() **and** (true() **or** false()) **and** false() → **false**

Vrednosti in konstante

- Vrednosti lahko zapišemo v obliki
  - nizov

```
doc("db/OPB/katalog.xml")//izdelek/@oddelek = "WMN"
```

  - števil

```
doc("db/OPB/narocilo.xml")//postavka/@kolicina > 1
```

  - vrednosti določenega tipa

```
doc("db/OPB/cene.xml")//@datumOd > xs:date("2004-10-11")
```

Spremenljivke

- pred imenom uporabimo znak **$**
- uporabljamo jih lahko na več mestih
  - FLWOR izrazi,

```
for $izdelek in doc("db/OPB/katalog.xml")//izdelek
return $izdelek/stevilka
```

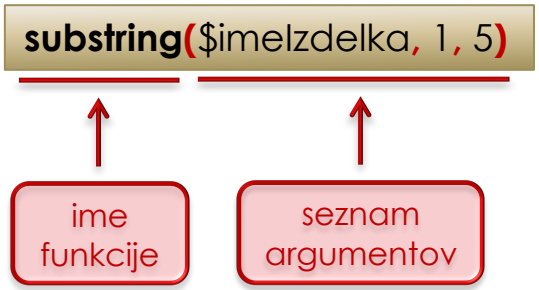
  - opredelitev funkcije,

```
declare function local:vrniStIzdelka($izdelek as element()) as element()
{ $izdelek/stevilka };
```

  - ipd.

Klic funkcije

- argument je lahko poljuben izraz
  - npr. referenca za spremenljivko, XPath ipd.
- pri argumentu lahko zahtevamo določen tip



Komentarji

- XQuery komentarji
  - začnejo in končajo se z nizoma **(:** in **:)**

    (: To je XQuery komentar … **:)**
  - se ne pojavljajo v rezultatu
- XML komentarji
  - se lahko pojavijo v rezultatu

    <!-- To je XML komentar … **-->**

Primerjalni operatorji

- dve skupini:
  - **primerjava po vrednosti**
    - **eq**, **ne**, **lt**, **le**, **gt**, **ge**
    - za primerjavo posameznih vrednosti
    - vsak operand mora biti atomarna vrednost
  - **splošna primerjava**
    - **=**, **!=**, **<**, **<=**, **>**, **>=**
    - lahko uporabimo tudi na zaporedju več elementov

Splošna primerjava vs. primerjava po vrednosti

```
doc("db/OPB/narocilo.xml")//postavka/@kolicina > 1
```

- vrne true, če ima **poljuben atribut** *kolicina* **vrednost večjo od 1**

```
doc("db/OPB/narocilo.xml")//postavka/@kolicina gt 1
```

- vrne true, če obstaja **le 1 atribut** *kolicina*, ki ima **vrednost večjo od 1**,
- če obstaja več atributov *kolicina*, pride do napake

Pogojni izrazi (1)

- **if-then-else** primer

```
for $izdelek in doc("db/OPB/katalog.xml")/katalog/izdelek
return if ($izdelek/@oddelek = 'ACC')
    then <acc>{data($izdelek/stevilka)}</acc>
    else <ostalo>{data($izdelek/stevilka)}</ostalo>
```

- oklepaji okrog if izraza so potrebni,
- else del je vedno potreben, lahko je prazen, npr. **else ()**

Pogojni izrazi (2)

- izraz **if** mora biti logična vrednost,
  - če ni, se uporabi **efektivna logična vrednost**,
    - efektivna logična vrednost vrne **false**:
      - xs:boolean vrednost false
      - vrednost 0 ali NaN
      - niz, dolžine 0
      - prazen niz
    - drugače vrne **true** (npr. seznam elementov)

```
if (doc("db/OPB/narocilo.xml")//postavka)
then "Seznam elementov: " else " "
```

Pogojni izrazi (3)

Gnezdenje

```
if ($izdelek/@oddelek = 'ACC')
then <dodatek>{data($izdelek/stevilka)}</dodatek>
else if ($izdelek/@oddelek = 'WMN')
   then <zenski>{data($izdelek/stevilka)}</zenski>
   else if ($izdelek/@oddelek = 'MEN')
      then <moski>{data($izdelek/stevilka)}</moski>
      else <ostalo>{data($izdelek/stevilka)}</ostalo>
```

Logični izrazi

- **and** in **or** operatorji
  - and ima prioriteto pred or

```
if ($jePopust and ($popust > 10 or $popust < 0))
then 10 else $popust
```

- **negacija**

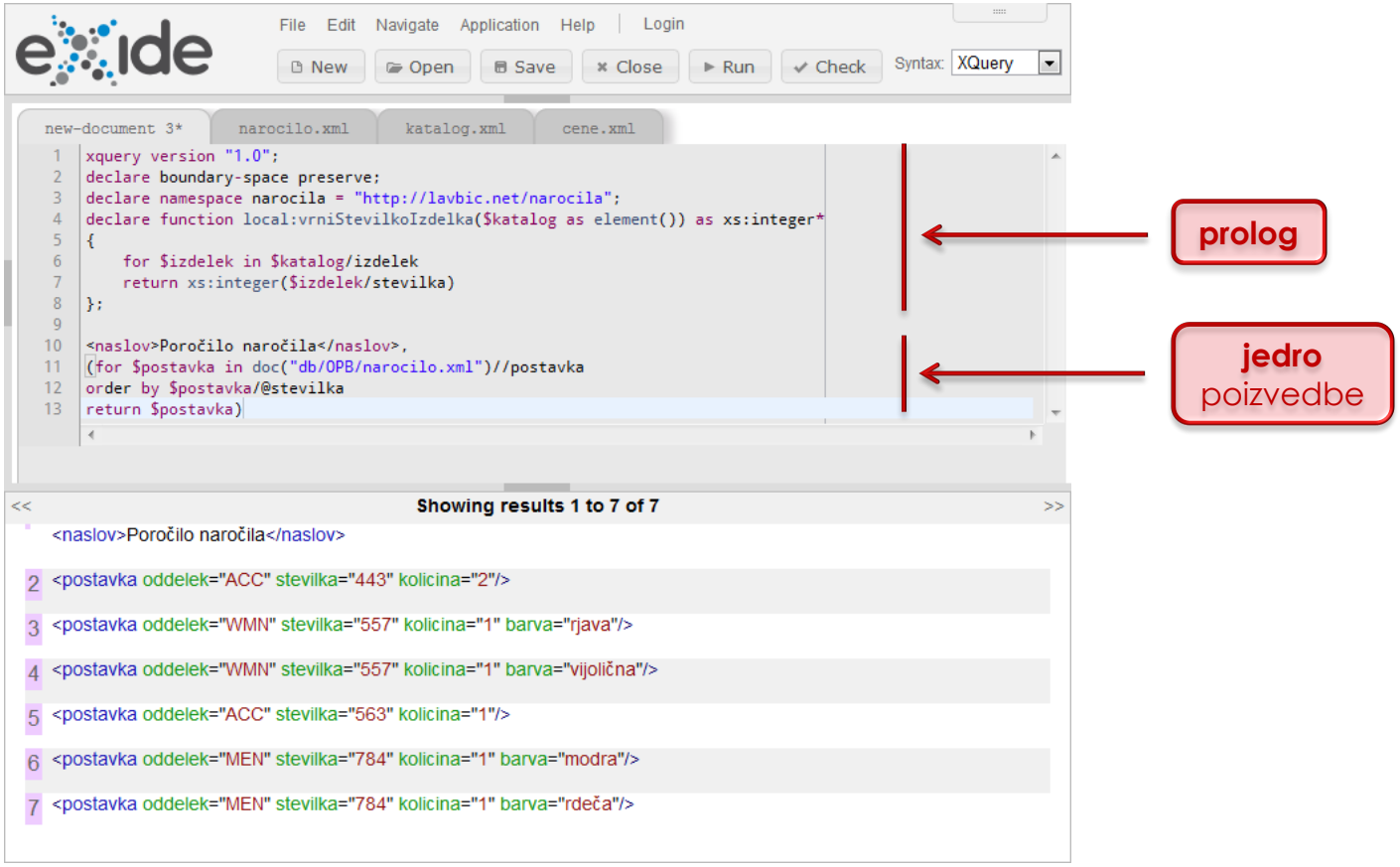
```
if (not($jePopust)) then 0 else $popust
```

- podobno kot pri pogojnih izrazih, velja tudi tukaj podobno za efektivne logične vrednosti

Poizvedovalni prolog (1)

- omogoča opredelitev številnih nastavitev:
  - imenski prostor,
  - funkcije,
  - uvoze ali zunanje module oz. sheme,
  - privzete operacije itd.
- pojavi se pred jedrom v poizvedbi
  - vsaka opredelitev je ločen s podpičjem

Poizvedovalni prolog (2)

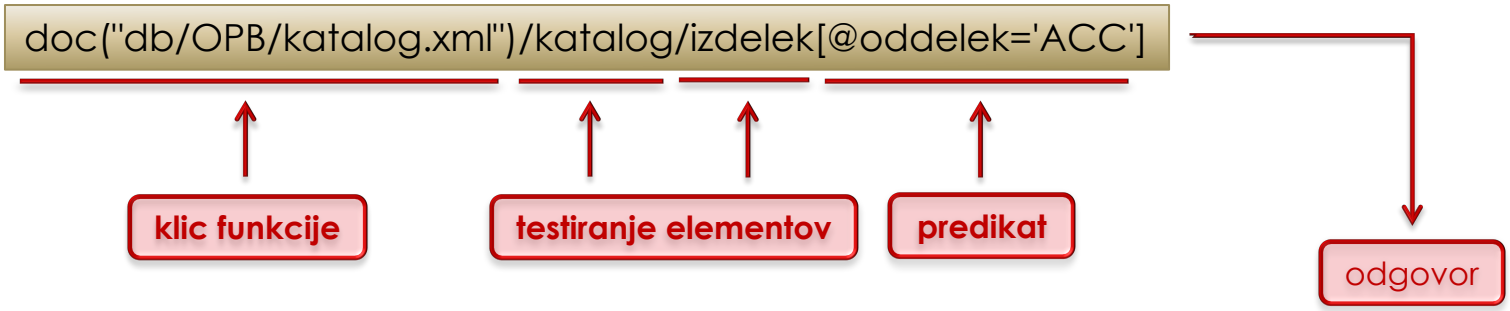


Izrazi za opredelitev poti (XPath)

Smer navigacije, testiranje vozlišč, predikati

Izrazi za opredelitev poti

- Uporabljajo se za sprehajanje po vhodnem dokumentu, kjer **izbiramo** želene **elemente** in **atribute**




```
1 <izdelek oddelek="ACC">
    <stevilka>563</stevilka>
    <ime jezik="sl">kavbojski klobuk</ime>
  </izdelek>
2 <izdelek oddelek="ACC">
    <stevilka>443</stevilka>
    <ime jezik="sl">senčnik</ime>
  </izdelek>
```

Smer navigacije

- sestavljajo naslednje komponente:

```
doc("db/OPB/narocilo.xml")//child::postavka[@oddelek='ACC']
```

  - **smer** (opcijska),
    - child:: (privzeta), parent::, descendant:: itd.
  - **testni pogoj elementa**,
    - imena oz. tipi elementov,
  - **predikat** (opcijski in ponavljajoč),
    - za filtriranje.

odgovor

```
1 <postavka oddelek="ACC" stevilka="563" kolicina="1"/>
2 <postavka oddelek="ACC" stevilka="443" kolicina="2"/>
```

Smer navigacije **child** (1)

Različni zapisi

- Primeri uporabe smeri **child**, ki vrnejo isti odgovor:

```
doc("db/OPB/narocilo.xml")/narocilo/postavka
doc("db/OPB/narocilo.xml")/child::narocilo/child::postavka
doc("db/OPB/narocilo.xml")/*/postavka
```

```
1 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="rjava"/>
2 <postavka oddelek="ACC" stevilka="563" kolicina="1"/>
3 <postavka oddelek="ACC" stevilka="443" kolicina="2"/>
4 <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="modra"/>
5 <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="rdeča"/>
6 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="vijolična"/>
```

odgovor

Smer navigacije **child** (2)

Rekurzija

- Če želimo vrniti **vse** (neposredne in posredne) **pod elemente**, lahko uporabimo:

```
doc("db/OPB/narocilo.xml")//postavka
doc("db/OPB/narocilo.xml")/descendant-or-self::postavka
```

```
1 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="rjava"/>
2 <postavka oddelek="ACC" stevilka="563" kolicina="1"/>
3 <postavka oddelek="ACC" stevilka="443" kolicina="2"/>
4 <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="modra"/>
5 <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="rdeča"/>
6 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="vijolična"/>
```

odgovor

Smer navigacije **parent**

- Vrne **nad element** izbranega elementa:

```
doc("db/OPB/narocilo.xml")/narocilo/postavka/..
doc("db/OPB/narocilo.xml")/narocilo/postavka/parent::narocilo
```

```
1 <narocilo st="00299432" datum="2004-09-15" stranka="0221A">
    <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="rjava"/>
    <postavka oddelek="ACC" stevilka="563" kolicina="1"/>
    <postavka oddelek="ACC" stevilka="443" kolicina="2"/>
    <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="modra"/>
    <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="rdeča"/>
    <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="vijolična"/>
</narocilo>
```

odgovor

Ostale smeri navigacije

- **self**
  - trenutni element (okrajšava je .)
- **ancestor, ancestor-or-self**
  - prednik, nad element nad elementa itd.
- **following, preceding**
  - nasledniki in predhodniki v dokumentu
- **following-sibling, preceding-sibling**
  - naslednji brat/sestra

Testiranje vozlišč

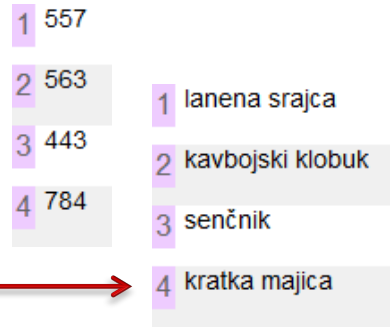
- pri testiranju lahko uporabimo:
  - **ime** elementa

    doc("db/OPB/narocilo.xml")/narocilo/**postavka**/**@oddelek**
  - **tip** elementa

    doc("db/OPB/katalog.xml")/katalog/**element()**

    doc("db/OPB/katalog.xml")//stevilka/**text()**

    doc("db/OPB/katalog.xml")//ime/**node()**

  - **poljubno ujemanje** (*)

    doc("db/OPB/narocilo.xml")/narocilo/***/@***

    vsi atributi vseh pod elementov naročila

Predikati (1)

- **Filtriranje elementov** na podlagi določenih kriterijev.
- **Primer**: Vrni vse izdelke s številko manjšo od 500.

```
doc("db/OPB/katalog.xml")//izdelek[stevilka < 500]
```

```
1 <izdelek oddelek="ACC">
    <stevilka>443</stevilka>
    <ime jezik="sl">senčnik</ime>
  </izdelek>
```

Predikati (2)

- Logično testiranje pogojev:

```
doc("db/OPB/narocilo.xml")//postavka
```

```
1 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="rjava"/>
2 <postavka oddelek="ACC" stevilka="563" kolicina="1"/>
3 <postavka oddelek="ACC" stevilka="443" kolicina="2"/>
4 <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="modra"/>
5 <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="rdeča"/>
6 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="vijolična"/>
```

Vrne **vse elemente** postavke naročila.

Predikati (3)

- Logično testiranje pogojev:

```
doc("db/OPB/narocilo.xml")//postavka[@barva]
```

```
1 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="rjava"/>
2 <postavka oddelek="ACC" stevilka="563" kolicina="1"/>
3 <postavka oddelek="ACC" stevilka="443" kolicina="2"/>
4 <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="modra"/>
5 <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="rdeča"/>
6 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="vijolična"/>
```

Vrne vse elemente postavke naročila, ki **vsebujejo atribut barva**.

Predikati (4)

- Logično testiranje pogojev:

```
doc("db/OPB/narocilo.xml")//postavka[@barva = 'rdeča']
```

```
1 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="rjava"/>
2 <postavka oddelek="ACC" stevilka="563" kolicina="1"/>
3 <postavka oddelek="ACC" stevilka="443" kolicina="2"/>
4 <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="modra"/>
5 <postavka oddelek="MEN" stevilka="784" kolicina="1" barva="rdeča"/>
6 <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="vijolična"/>
```

Vrne vse elemente postavke naročila, ki **vsebujejo atribut barva**, njegova vrednost pa je **rdeča**.

Predikati (5)

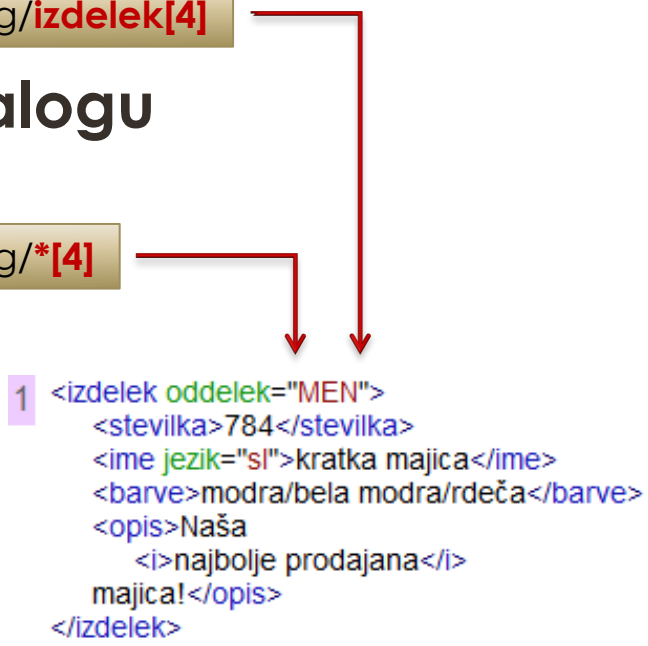
Položaj v zaporedju (1)

- Vrni **4. izdelek v katalogu**:

```
doc("db/OPB/katalog.xml")/katalog/izdelek[4]
```

- Vrni **4. element v katalogu** (ne glede na ime):

```
doc("db/OPB/katalog.xml")/katalog/*[4]
```



Predikati (6)

Položaj v zaporedju (2)

- Vrni **prve 3 izdelke v katalogu**:

```
doc("db/OPB/katalog.xml")/katalog/izdelek[position() < 4]
```

- Vrni **zadnji izdelek v katalogu**:

```
doc("db/OPB/katalog.xml")/katalog/izdelek[last()]
```

Predikati (7)

Več pogojev

- Pogoji se vedno preverjajo od leve proti desni!
- Vrni izdelke s **številko manjšo od 500** in **oddelka ACC**:

```
doc("db/OPB/katalog.xml")/katalog/izdelek[stevilka < 500][@oddelek = 'ACC']
```

- Izmed izdelkov **oddelka ACC vrni drugega**:

```
doc("db/OPB/katalog.xml")/katalog/izdelek[@oddelek = 'ACC'][2]
```

- Poglej 2. izdelek in če je iz ACC oddelka, ga vrni:

```
doc("db/OPB/katalog.xml")/katalog/izdelek[2][@oddelek = 'ACC']
```

Predikati (8)

Kompleksnejši primeri

- Vrni izdelke, katerih **oddelek vsebuje črko A**:

```
doc("db/OPB/katalog.xml")/katalog/izdelek[contains(@oddelek, 'A')]
```

- Vrni izdelke, ki imajo **poleg atributov stevilka in ime še vsaj 1 dodaten atribut**:

```
doc("db/OPB/katalog.xml")/katalog/izdelek[* except (stevilka, ime)]
```

- Vrni **izdelke na sodih mestih**:

```
doc("db/OPB/katalog.xml")/katalog/izdelek[position() mod 2 = 0]
```

Dodajanje elementov in atributov

3 načini, dinamični konstruktorji

Dodajanje elementov / atributov v rezultat

- Na voljo imamo 3 načine:
  - **vključevanje** iz vhodnega dokumenta,
    - kot prikazano na večini prejšnjih primerov
  - z uporabo **neposrednih konstruktorjev**,
    - XML zapis
  - z uporabo **dinamičnih konstruktorjev**,
    - poseben zapis znotraj { … },
    - omogoča dinamična imena.

1. način

Vključevanje iz vhodnega dokumenta

```
for $izdelek in doc("db/OPB/katalog.xml")/katalog/izdelek[@oddelek = 'ACC']
return $izdelek/ime
```

```
1 <ime jezik="sl">kavbojski klobuk</ime>
2 <ime jezik="sl">senčnik</ime>
```

- element ime je vključen v takšni obliki, kot je v vhodnem dokumentu:
  - skupaj z atributi (in podrejenimi elementi, če obstajajo),
  - ne samo atomarne vrednosti,
- ni možnosti spreminjanja atributov, podrejenih elementov ali imenskega prostora.

2. način (1)

Neposredni konstruktor » primer

```
<html><h1>Katalog izdelkov</h1>
<ul>{
for $izdelek in doc("db/OPB/katalog.xml")/katalog/izdelek
return <li>#{data($izdelek/stevilka)} je {data($izdelek/ime)}</li>
}</ul>
</html>
```

```
<html>
    <h1>Katalog izdelkov</h1>
    <ul>
        <li>#557 je lanena srajca</li>
        <li>#563 je kavbojski klobuk</li>
        <li>#443 je senčnik</li>
        <li>#784 je kratka majica</li>
    </ul>
</html>
```

2. način (2)
Neposredni konstruktor

- uporablja se XML zapis
  - in s tem povezava pravila (pravilno gnezdenje ipd.),
- lahko vsebuje:
  - atomarne vrednosti,
  - ostale konstruktorje,
  - izraze v zavitih oklepajih {…}
    - Za preverjanje elementov, atributov ali atomarnih vrednosti

2. način (3)

Neposredni konstruktor » atomarne vrednosti

- Vsi znaki zunaj zavitih oklepajev {…} gredo neposredno v rezultat

```
<li>Številka izdelka {data($izdelek/stevilka)}</li>
```

2. način (4)

Neposredni konstruktor » ostali konstruktorji

- Brez zavitih oklepajev {…} ali posebnih ločevalnih znakov

```
<html>
   <h1>Katalog izdelkov</h1>
   <p><i>Obsežen</i> seznam {
      count(doc("db/OPB/katalog.xml")//izdelek)
   } izdelkov.</p>
</html>
```

```
<html>
   <h1>Katalog izdelkov</h1>
   <p>
      <i>Obsežen</i>
   seznam 4 izdelkov.</p>
</html>
```

2. način (5)

Neposredni konstruktor » izrazi v {…}

```
for $izdelek in doc("db/OPB/katalog.xml")/katalog/izdelek
return <li>{$izdelek/@oddelek}
                {concat("št. ", ": ")}
                    {$izdelek/stevilka}</li>
```

Vozlišča atributov postanejo **atributi**

Atomarne vrednosti postanejo **znakovni podatki**

vozlišča elementov postanejo **pod elementi**

```
<li oddelek="WMN">št. : <stevilka>557</stevilka></li>
<li oddelek="ACC">št. : <stevilka>563</stevilka></li>
<li oddelek="ACC">št. : <stevilka>443</stevilka></li>
<li oddelek="MEN">št. : <stevilka>784</stevilka></li>
```

2. način (6)

Neposredni konstruktor » izrazi v {…}

- Lahko tudi neposredno opredelimo atribute:

```
<h1 class="izdelekGlava">Katalog izdelkov</h1>,
<ul>{
  for $izdelek in doc("db/OPB/katalog.xml")/katalog/izdelek
  return <li class="{$izdelek/@oddelek}">{data($izdelek/ime)}</li>
}</ul>
```

- Podobno kot konstruktor elementa, lahko vsebuje:
  - atomarno vrednost,
  - vgrajene izraze
    - ki pa se vedno prevedejo v atomarne vrednosti.

2. način (7)

Neposredni konstruktor » izrazi v {…}

- **Dodajanje atributa** izdelku:

```
for $izdelek in doc("db/OPB/katalog.xml")//izdelek[@oddelek = 'ACC']
return <izdelek id="{concat("I", $izdelek/stevilka)}">
            {$izdelek/@*}{$izdelek/*}</izdelek>
```

```
<izdelek oddelek="ACC" id="I563">
   <stevilka>563</stevilka>
   <ime jezik="sl">kavbojski klobuk</ime>
</izdelek>
<izdelek oddelek="ACC"  id="I443">
   <stevilka>443</stevilka>
   <ime jezik="sl">senčnik</ime>
</izdelek>
```

3. način (1)
Dinamični konstruktor

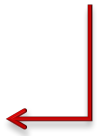
- Omogoča dinamično opredelitev imen in vrednosti in je uporaben za:
  - **kopiranje elementov** iz vhodnega dokumenta in izvajanje manjših sprememb,
    - npr. dodajanje atributov,
  - **pretvorba vsebine** vhodnega dokumenta v element ali atribut
  - **iskanje imen** elementov v **ločenem slovarju**
    - npr. podpora večjezičnosti

3. način (2)

Dinamični konstruktor » enostaven primer

```
element izdelek {
    attribute oddelek {"ACC"},
    element {concat("st", "evilka")} {563},
    element ime {attribute jezik {"sl"}, "kavbojski klobuk"}
}
```

```
<izdelek oddelek="ACC">
    <stevilka>563</stevilka>
    <ime jezik="sl">kavbojski klobuk</ime>
</izdelek>
```



3. način (3)

Dinamični konstruktor » pretvorba vsebine

```
for $oddelek in distinct-values(doc("db/OPB/katalog.xml")//izdelek/@oddelek)
return element {$oddelek}
            {doc("db/OPB/katalog.xml")//izdelek[@oddelek = $oddelek]/ime}
```

```
<WMN>
   <ime jezik="sl">lanena srajca</ime>
</WMN>
<ACC>
   <ime jezik="sl">kavbojski klobuk</ime>
   <ime jezik="sl">senčnik</ime>
</ACC>
<MEN>
   <ime jezik="sl">kratka majica</ime>
</MEN>
```

Izbiranje in filtriranje

XPath vs. FLWOR, for, let, where, return, enolične vrednosti

Na voljo sta 2 načina …

- **XPath** izrazi
  - primeren takrat, ko želimo kopirati vsebino izbranih elementov in atributov v takšni obliki, kot so v izvornem dokumentu
- **FLWOR** izrazi
  - omogočajo sortiranje,
  - omogočajo dodajanje elementov/atributov k rezultatu,
  - bolj zahtevna oblika, a bolj jasno opredeljena.

Gradniki FLWOR izrazov (1)

- **for**
  - iterativno poveže spr. $izdelek z XPath izrazom
- **let**
  - vrednost spr. $oddelekIzdelka nastavi na vrednost atributa spr. $izdelek
- **where**
  - izbere vozlišča, katerih atributi so "WMN" oz. "ACC"
- **return**
  - vrne pod element ime izbranega vozlišča

```
for $izdelek in doc("db/OPB/katalog.xml")//izdelek
let $oddelekIzdelka := $izdelek/@oddelek
where $oddelekIzdelka = "ACC" or
      $oddelekIzdelka = "WMN"
return $izdelek/ime
```

Gradniki FLWOR izrazov (2)

for

- iterativno poveže spremenljivko z vsakim elementom, ki ga vrne **in** del

```
for $izdelek in doc("db/OPB/katalog.xml")//izdelek
```

- sprehajamo se lahko tudi po določeni množici

```
for $i in (1 to $stIzdelkov)
```

Gradniki FLWOR izrazov (3)

for » večkratne spremenljivke

- večkratne pogoje v **in** delu ločimo z vejicami ali večkratnimi for zankami,

```
for $i in (1, 2), $j in (11, 12)
return <rezultat>i je {$i} in j je {$j}</rezultat>
```

- rezultat so izračunane vrednosti za vse kombinacije

```
<rezultat>i je 1 in j je 11</rezultat>
<rezultat>i je 1 in j je 12</rezultat>
<rezultat>i je 2 in j je 11</rezultat>
<rezultat>i je 2 in j je 12</rezultat>
```

Gradniki FLWOR izrazov (4)

for » položaj

- s pomočjo položaja, lahko določimo zaporedno številko v iteraciji
  - za to uporabljamo rezervirano besedo **at**

```
for $izdelek at $i in doc("db/OPB/katalog.xml")
                          //izdelek[@oddelek = "ACC" or @oddelek = "WMN"]
return <rezultat>{$i}. {data($izdelek/ime)}</rezultat>
```

```
<rezultat>1. lanena srajca</rezultat>
<rezultat>2. kavbojski klobuk</rezultat>
<rezultat>3. senčnik</rezultat>
```

Gradniki FLWOR izrazov (5)

let

- enostaven način za shranjevanje spremenljivk
  - ni potrebe po ponavljanjem istega izraza

```
for $i in (1 to 3)
return <rezultat>{1i}</rezultat>
```

↓

```
<rezultat>1</rezultat>
<rezultat>2</rezultat>
<rezultat>3</rezultat>
```

```
let $i := (1 to 3)
return <rezultat>{1i}</rezultat>
```

↓

```
<rezultat>1 2 3</rezultat>
```

Gradniki FLWOR izrazov (6)

večkratni **for** in **let**

```
let $dokumentKatalog := doc("db/OPB/katalog.xml")
for $izdelek in $dokumentKatalog//izdelek
let $oddelekIzdelka := $izdelek/@oddelek
where $oddelekIzdelka = "ACC" or $oddelekIzdelka = "WMN"
return $izdelek/ime
```

```
<ime jezik="sl">lanena srajca</ime>
<ime jezik="sl">kavbojski klobuk</ime>
<ime jezik="sl">senčnik</ime>
```

Gradniki FLWOR izrazov (7)

return

- vrednost, ki se vrača kot rezultat,

```
for $izdelek in doc("db/OPB/katalog.xml")//izdelek
return $izdelek/ime
```

- kot rezultat lahko vrnemo le 1 izraz
  - lahko pa združujemo več izrazov v zaporedje

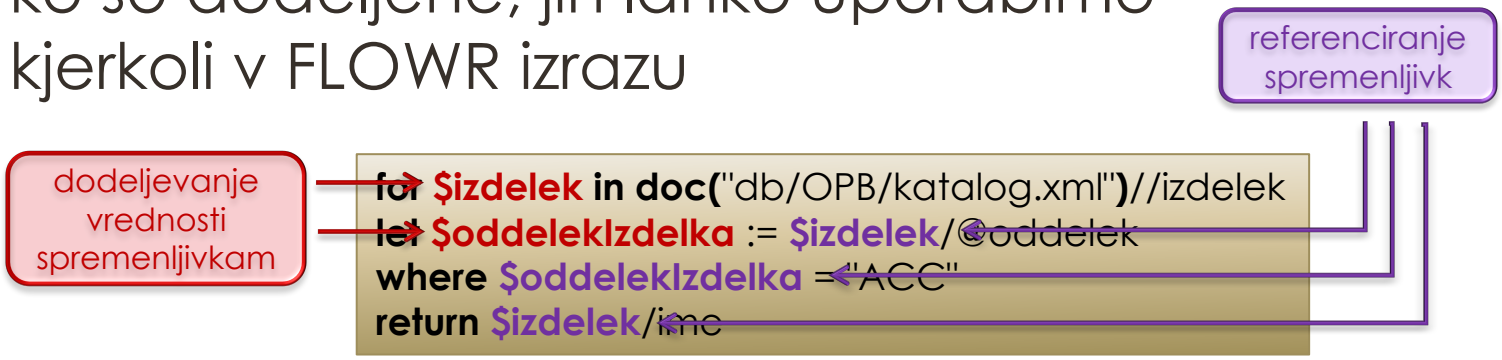
~~`return <a>{$i}</a> <b>{$j}</b>`~~

```
return (<a>{$i}</a>,
        <b>{$j}</b>)
```

Gradniki FLWOR izrazov (8)

dodeljevanje in referenciranje spremenljivk

- vrednosti se spremenljivkam dodelijo v let/for delu
- ko so dodeljene, jih lahko uporabimo kjerkoli v FLOWR izrazu



- ko spremenljivkam dodelimo vrednosti, jih ne moremo spremeniti
  - npr. let $stevec := $stevec + 1 ni dovoljeno

Gradniki FLWOR izrazov (9)
kvantifikatorji

- Uporabljajo se takrat, kjer vsi ali zgolj nekaj elementov zaporedja ustreza podanemu pogoju.
  - uporabljamo **some** ali **every** skupaj s **satisfies**

```
some $oddelek in doc("db/OPB/katalog.xml")//izdelek/@oddelek
satisfies ($oddelek = "ACC")
```
→ **true**

```
every $oddelek in doc("db/OPB/katalog.xml")//izdelek/@oddelek
satisfies ($oddelek = "ACC")
```
→ **false**

Gradniki FLWOR izrazov (10)

enolične vrednosti

- uporabimo funkcijo **distinct-values**

```
distinct-values(doc("db/OPB/katalog.xml")//izdelek/@oddelek)
```

WMN, ACC, MEN

- deluje le na atomarnih vrednostih oz. elementih z 1 atomarno vrednostjo,
- sprejme 1 argument,
  - za kombinacijo več vrednosti, lahko z uporabo for pokličemo funkcijo večkrat.

Gradniki FLWOR izrazov (11)

kombinacije enoličnih vrednosti

```
let $izdelki := doc("db/OPB/katalog.xml")//izdelek
for $oddelek in distinct-values($izdelki/@oddelek),
    $stevilka in distinct-values($izdelki[@oddelek = $oddelek]/stevilka)
return <odgovor oddelek="{$oddelek}" stevilka="{$stevilka}" />
```

```
<odgovor oddelek="WMN" stevilka="557"/>
<odgovor oddelek="ACC" stevilka="563"/>
<odgovor oddelek="ACC" stevilka="443"/>
<odgovor oddelek="MEN" stevilka="784"/>
```

spremenljivka **$oddelek** je dodeljena vsakemu enoličnemu oddelku

spremenljivka **$stevilka** je dodeljena vsaki enolični številki oddelka

Sortiranje rezultatov

Order by, vrstni red

Sortiranje z **order by** (1)

- edini način za sortiranje rezultatov v XQuery,
- rezervirani besedi order by moramo uporabiti pred return,
- lahko opredelimo več vrednosti, po katerih sortiramo

```
for $postavka in doc("db/OPB/narocilo.xml")//postavka
order by $postavka/@oddelek, $postavka/@stevilka descending
return $postavka
```

Sortiranje z **order by** (2)

- sortiranje ni omejeno zgolj na enostaven XPath izraz, ampak lahko uporabimo tudi:
  - klice funkcij,

```
order by substring($izdelek/@oddelek, 2, 2)
```

  - pogojne izraze,

```
order by (if ($izdelek/@barve) then $izdelek/@barve else "neznano")
```

  - poizvedbe v ločenih dokumentih

```
order by doc("db/OPB/katalog.xml")//izdelek[stevilka = $izdelek/@st]/ime
```

Sprememba vrstnega reda

- Vrstni red zaporedja lahko spremenimo z uporabo funkcije **reverse**,
- deluje nad vozlišči in atomarnimi vrednostmi

reverse((6, 3, 2)**)** → (2, 3, 6)

Združevanje rezultatov

Stiki, odprti stiki, grupiranje, agregacija

Združevanje s stikom (1)

```
<katalog>
   <izdelek oddelek="WMN">
      <stevilka>557</stevilka>
      <ime jezik="sl">lanena srajca</ime>
      <barve>rjava modra</barve>
   </izdelek>
   …
</katalog>
```

Dokument 1
Katalog izdelkov

Dokument 2
Naročilo

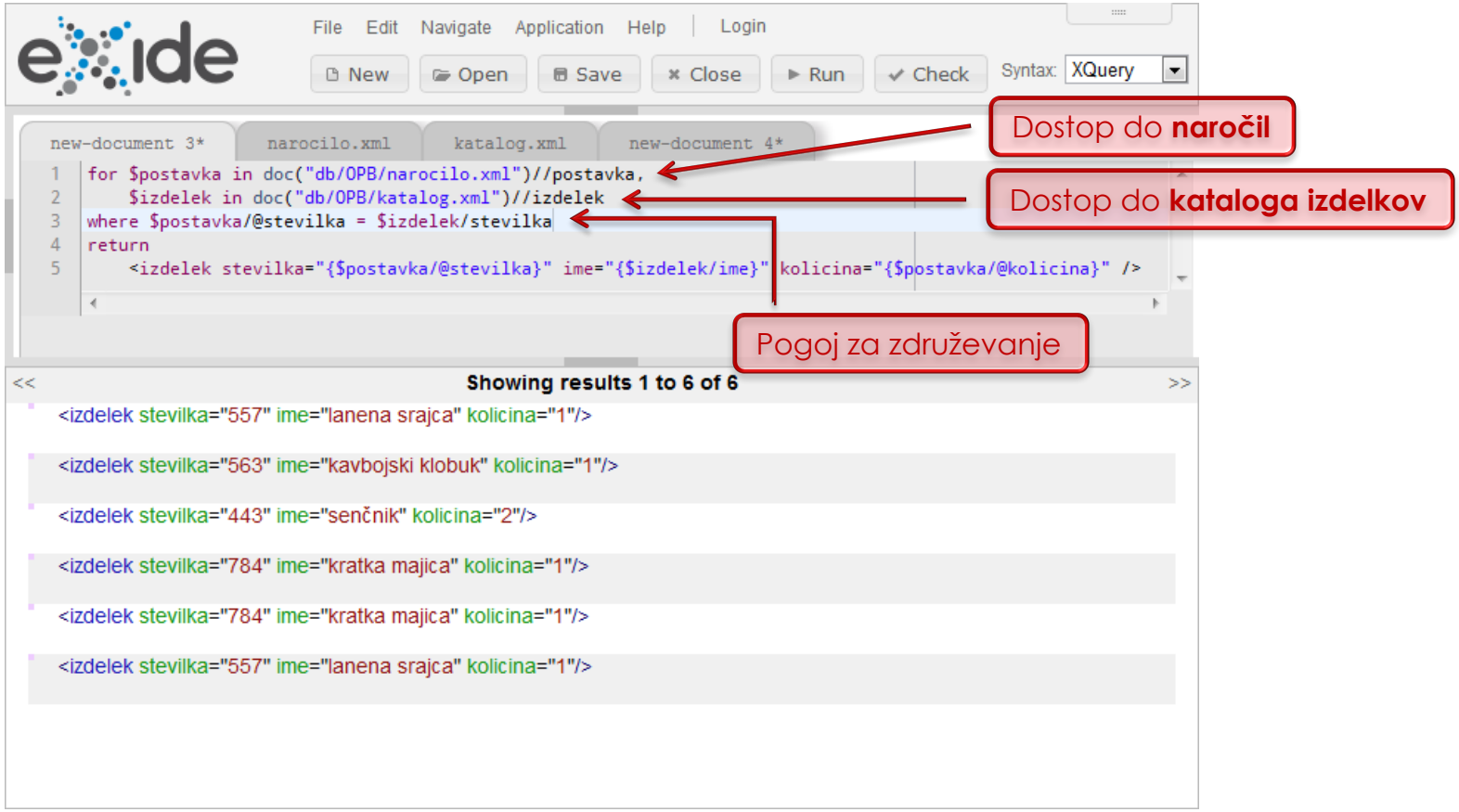
```
<narocilo st="00299432" datum="2004-09-15" stranka="0221A">
   <postavka oddelek="WMN" stevilka="557" kolicina="1" barva="rjava" />
   …
</narocilo>
```

↓ integracija

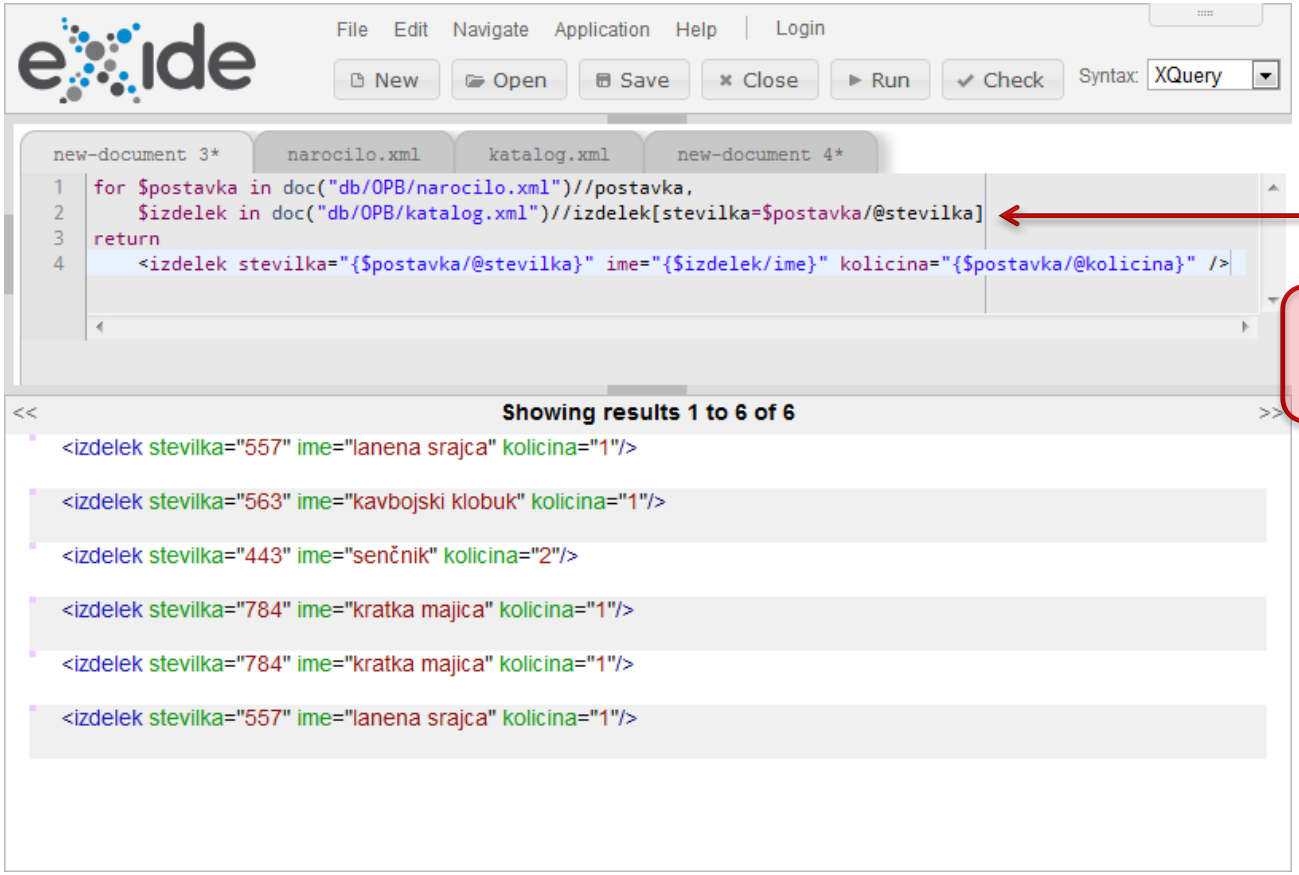
```
<izdelek stevilka="557" ime="lanena srajca" kolicina="1" />
…
```

Dokument 3
Združeni podatki naročila in kataloga izdelkov

Združevanje s stikom (2)

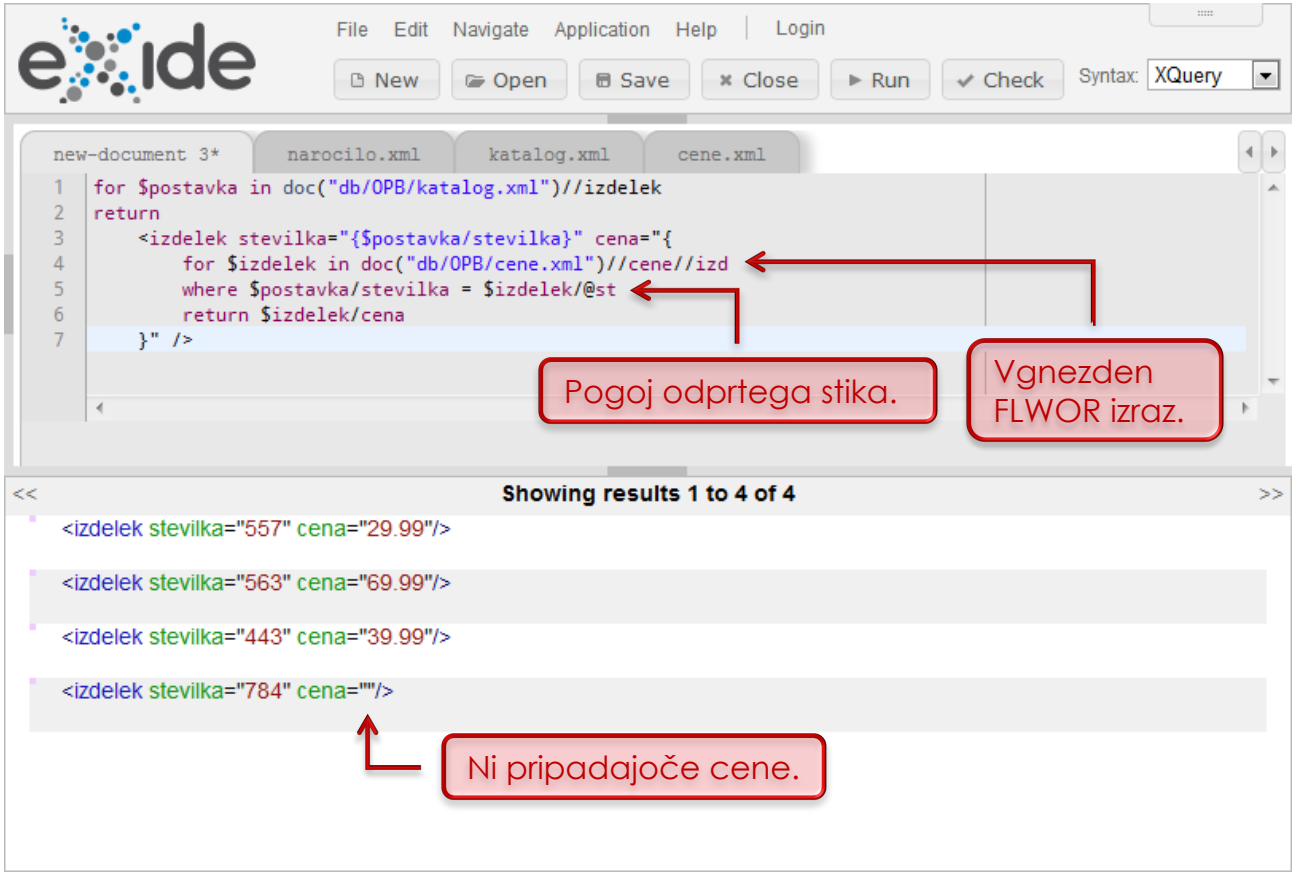


Združevanje s stikom (3)

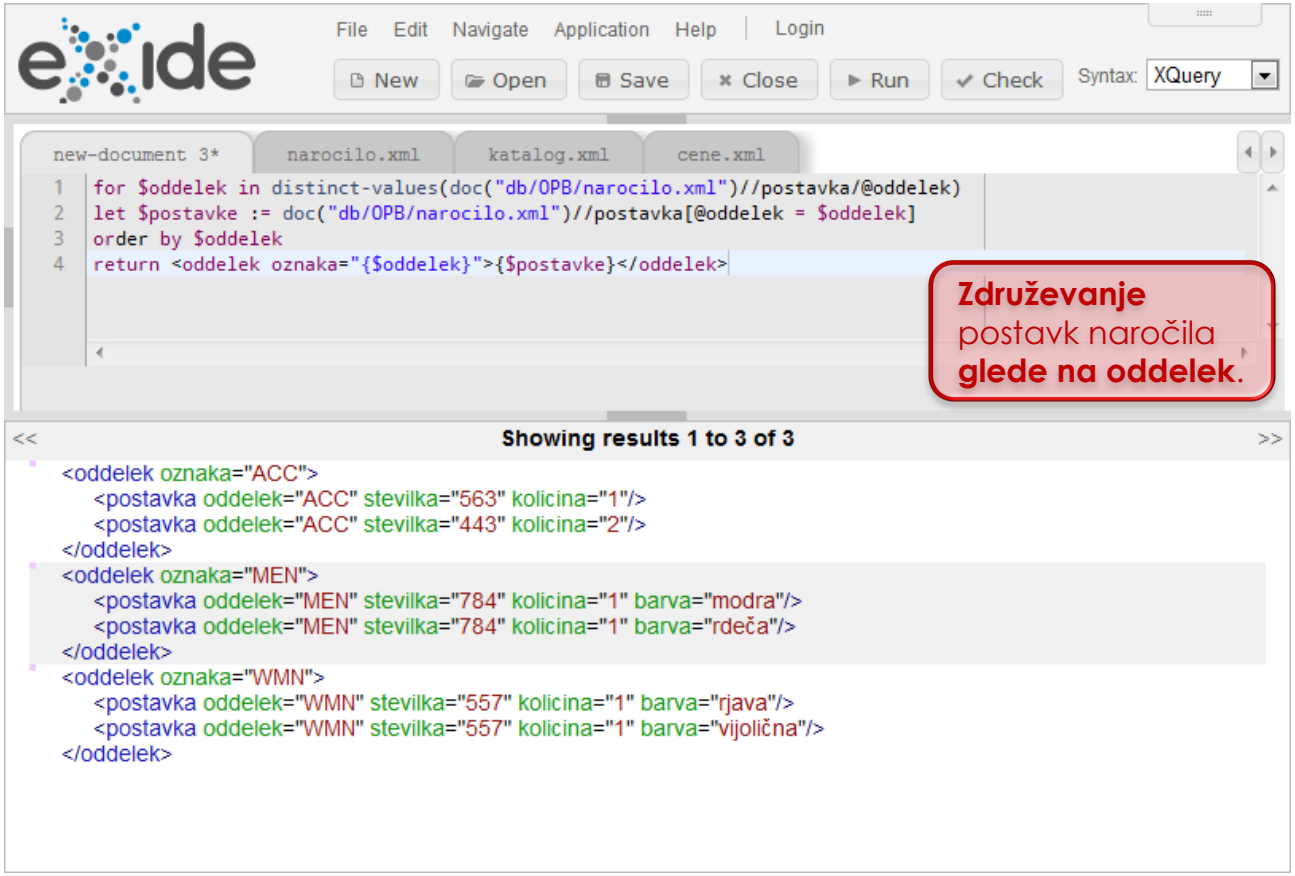


Drug način integracije istih podatkov.

Združevanje z odprtim stikom

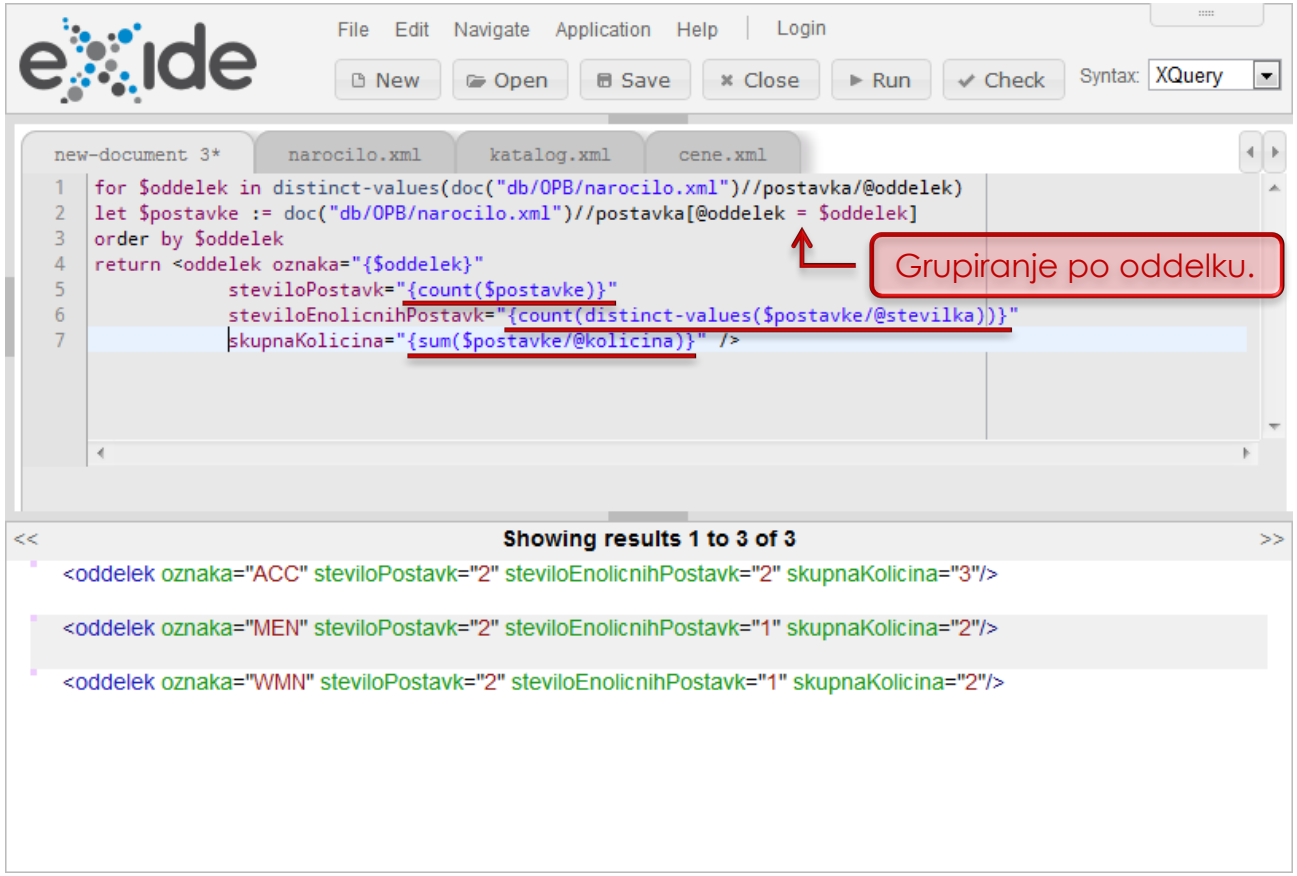


Grupiranje



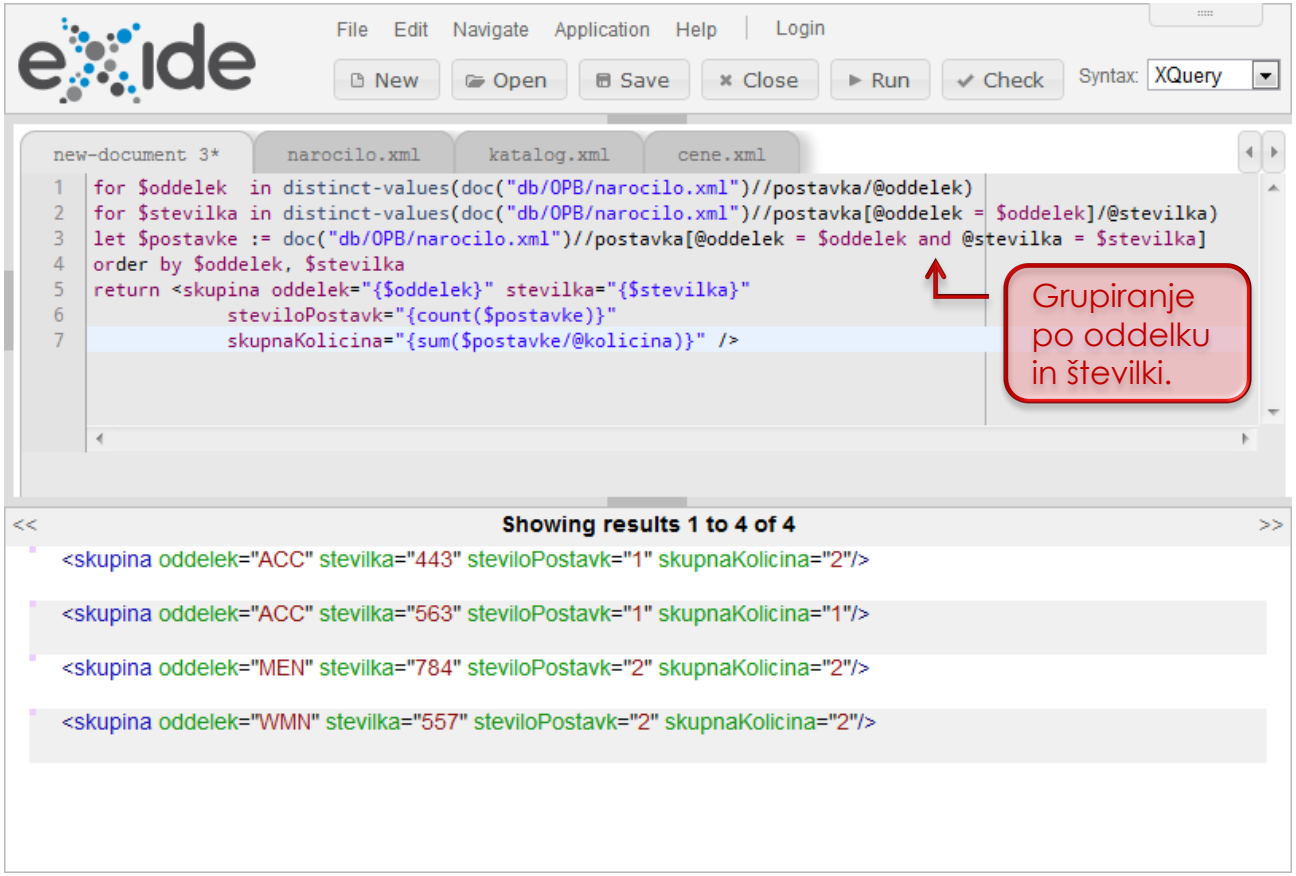
Agregacija (1)

... po oddelku



Agregacija (2)

... po oddelku in številki



Funkcije

Vgrajene, parametri

Funkcije

- **vgrajene** funkcije
  - jezik XQuery podpira več kot 100 vnaprej opredeljenih funkcij,
  - pri imenih ni potrebno uporabljati predpon.
- **lastne** funkcije
  - opredeljeni v poizvedbi ali v ločenem modulu,
  - pri imenih moramo uporabljati predpone.

Vgrajene funkcije (1)

- funkcije z **nizi**
  - substring, contains, matches, concat, normalize-space, tokenize
- **datumske** funkcije
  - current-date, month-from-date, adjust-time-to-timezone
- **številske** funkcije
  - round, avg, sum, ceiling

Vgrajene funkcije (2)

- funkcije **zaporedij**
  - index-of, insert-before, reverse, subsequence, distinct-values
- funkcije nad **elementi**
  - data, empty, exists, id, idref
- funkcije nad **imeni**
  - local-name, in-scope-prefixes, QName, resolve-QName

Vgrajene funkcije (3)

- funkcije **obvladovanja napak**
  - error, trace, exactly-one
- funkcije nad **dokumenti** in **URI naslovi**
  - collection, doc, root, base-uri

Kje uporabljamo klice funkcij?

- Povsod tam, kjer lahko uporabljamo izraze:
  - let delu FLWOR izraza

```
let $ime := substring($imeIzdelka, 1, 5)
```

  - na novo nastalem elementu

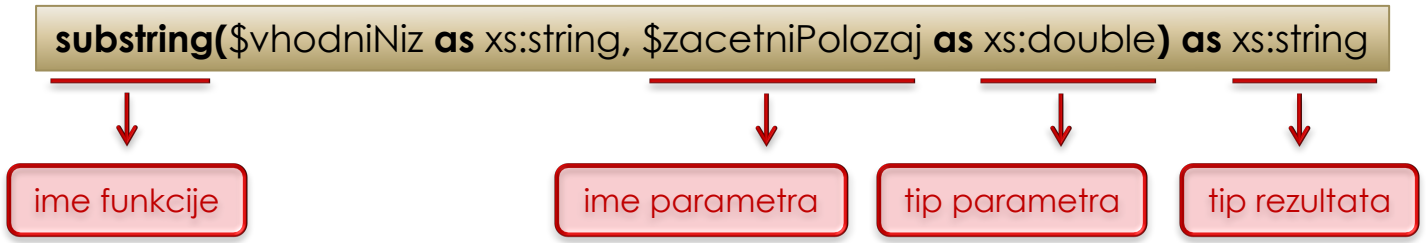
```
<ime>{substring($imeIzdelka, 1, 5)}</ime>
```

  - v predikatu izraza

```
doc("db/OPB/katalog.xml")/katalog/izdelek[substring(ime, 1, 3) = "lan"]
```

Opredelitev funkcije (1)

- Vsaka funkcija ima lahko več opredelitev:
  - ki opredeljuje ime, parametre in tip rezultata



Opredelitev funkcije (2)

- Če ima funkcija več opredelitev, potem mora imeti vsaka različno število parametrov:

```
substring($vhodniNiz as xs:string, $zacetniPolozaj as xs:double) as xs:string
```

```
substring($vhodniNiz as xs:string, $zacetniPolozaj as xs:double,
$dolzina as xs:double) as xs:string
```

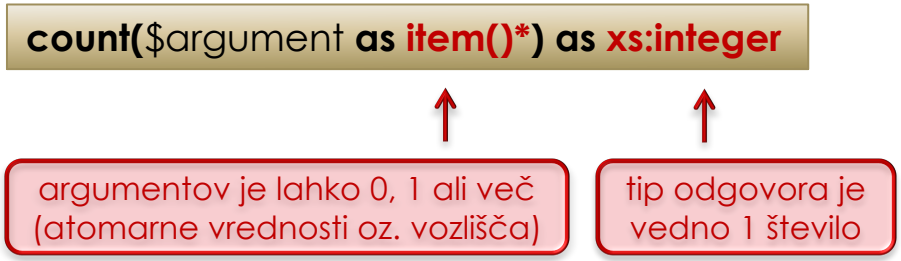
Druga opredelitev ima 1 dodaten parameter.

Zaporedje tipov (1)

- tipi parametrov in rezultata so predstavljeni kot zaporedje tipov,
- najbolj pogosto uporabljamo kar enostavne tipe
  - npr. xs:string, xs:integer, xs:double
- ostale možnosti zaporedja tipov
  - item(), node(), element(), attribute()

Zaporedje tipov (2)

- Indikator ponavljanja nam pove, koliko ponovitev dopuščamo:
  - **?** za 0 ali 1 element,
  - ***** za 0, 1 ali več elementov,
  - **+** za 1 ali več elementov,
  - **brez indikatorja** za natanko 1 element



Zaporedje tipov (3)

- Seznam argumentov mora ustrezati eni opredelitvi funkcije v številu in tipih

substring($imeIzdelka, 1, 5**)**

substring($imeIzdelka, 1**)**

substring($imeIzdelka, 1, **())** ← uporaba praznega zaporedja kot parameter ni isto kot izpuščanje parametra

substring($imeIzdelka, **"1")** ← neujemanje tipa parametra

Uporabniško opredeljene funkcije

Primeri, spremenljivke, rekurzivne funkcije …

Zakaj uporabljati uporabniško opredeljene funkcije?

- ponovna uporaba
  - ni potrebno ponavljati istih izrazov
    - poenostavitev poizvedbe,
    - vzdrževanje na enem mestu
- rekurzija,
  - brez uporabe funkcij je praktično nemogoča,
- itd.

Opredelitev funkcije

- lahko zapišemo v:
  - prolog ali
  - ločen modul,
- z uporabo **declare function** rezerviranih besed, katerim sledi
  - opredelitev,
  - jedro funkcije v zavitih oklepajih {…}.
- predpona
  - vnaprej opredeljena ali
  - **local**.

Primer opredelitve funkcije

```
declare function local:znizanaCena(
                $cena as xs:decimal?, $popust as xs:decimal?,
                $maxPopustIzdelek as xs:integer?) as xs:decimal?
{
   let $upostevanPopust := (if ($maxPopustIzdelek lt $popust)
      then $maxPopustIzdelek else $popust)
   return ($cena * (100 - $upostevanPopust)) div 100
};
```

opredelitev funkcije

jedro funkcije

```
for $izdelek in doc("db/OPB/cene.xml")//izd
return local:znizanaCena($izdelek/cena, $izdelek/popust, 5)
```

klic funkcije

Primer minimalne funkcije

- V jedru je lahko poljuben izraz,
  - ni nujno, da je FLOWR,
- parametri niso obvezni,
- tip rezultata ni obvezen.

```
declare function local:test() {2};
```

Kontekst funkcije

```
declare function local:izdelek2stevilka() as xs:string? {
   substring(stevilka, 2, 1)
};

for $izdelek in (doc("db/OPB/katalog.xml")//izdelek[local:izdelek2stevilka(.) > '5'])
return $izdelek
```

ni podanega konteksta za številko

```
declare function local:izdelek2stevilka($izdelek as element()?) as xs:string? {
   substring($izdelek/stevilka, 2, 1)
};

for $izdelek in (doc("db/OPB/katalog.xml")//izdelek[local:izdelek2stevilka(.) > '5'])
return $izdelek
```

kontekst se pošlje v argumentu

Rekurzivne funkcije

- Funkcije lahko kličejo samo sebe
- Primer:
  - vrni število pod elementov

```
declare function local:stPodElementov($element as element()) as xs:integer {
   sum(for $podElement in $element/*
         return local:stPodElementov($podElement) + 1)
};
```

Viri

W3C in ostalo dodatno gradivo

Dodatni viri (1)

- **W3C XML Query (XQuery)**
  - Osnove jezika XQuery.
  - http://www.w3.org/XML/Query/
- **W3C XQuery 1.0 and XPath 2.0 Function and Operators**
  - Seznam in primeri uporabe vgrajenih funkcij v poizvedovalni jezik XQuery in XPath.
  - http://www.w3.org/TR/xpath-functions/

Dodatni viri (2)

- **W3C XML Query Use Cases**
  - Primeri poizvedb na različnih primerih.
  - http://www.w3.org/TR/xquery-use-cases/
- **FunctX XQuery Functions**
  - Seznam več kot 100 primerov uporabnih XQuery funkcij.
  - http://www.xqueryfunctions.com/