



Interneto technologijos

XPath išraiškos



XPath išraiškos

- XPath keliai yra dalinis XPath išraiškų atvejis
- Išraiškos papildomai leidžia:
 - vietoj žingsnių naudoti funkcijas (kurios grąžina mazgų aibę)
 - su XPath kelių rezultatais atlikti **aritmetines, lyginimo ir aibių sąjungos** operacijas
 - išraiškos rezultatas gali būti ne tik mazgų aibė, bet ir **boolean**, **string**, **number** tipo reikšmė
 - Taigi XPath turi viso 4 tipus: **mazgų aibė, loginis, eilutės ir skaičiai**
- Išraiškoms, kurios grąžina mazgų aibę, galima taikyti predikatus



Išraiškų pavyzdžiai

- `/descendant::knyga[2]` |
`/descendant::knyga[4]`
aibių sąjunga: gražina aibę iš dviejų elementų
- `/descendant::knyga[3]/metai > 2005`
grąžina `true` arba `false`
- `/descendant::knyga[3]/metai - 1`
grąžina skaičių
- `count(//knyga)`
suskaiciuoja, kiek žymių "knyga" yra dokumente
- `sum(//knyga/kaina) - 0.01`
susumuoja bendrą kainą, ir atima vieną centą
- `(//knyga)[1]` – grąžina pirmą dokumento knygą (palyginkite su "spąstais")



Operacijų prioritetas

■ Prioriteto mažėjimo tvarka:

1. Skliausteliai: (ir)
2. Žingsnių skirtukas: / (tai ne dalyba!)
3. Aibių sąjungos operatorius: | (dirba tik su aibėmis)
4. Daugyba/dalyba: *, \div (slankiu kableliu), mod (liekana)
5. Sudėtis/atimtis: +, - (dirba tik su skaičiais!!!)
6. Mažiau/daugiau: <, <=, >, >= (dirba tik su skaičiais!!!)
7. Lygu/nelygu: =, != (su aibėmis nedirba!!!)
8. Loginis "ir": and
9. Loginis "arba": or

■ Pastaba: loginiam neigimui yra funkcija `not()`, funkcijos `true()` ir `false()` grąžina konstantas



Predikatai, taikomi išraiškoms

- Išraiškų rezultatams galima taikyti predikatus
 - Išraiška privalo grąžinti mazgų aibę (ne skaičių, eilutę ar loginį tipą)
 - Predikatai išraiškos rezultatą filtruoja taikydami `child` ašį pagal nutylėjimą
- Pavyzdys:
 - `((//knyga | //žurnalas)[1]`
grąžins pirmą dokumento knygą arba pirmą žurnalą, priklausomai kas bus sutiktas pirmiau
 - `((//knyga)[last()]/preceding::knyga)[1]`
grąžins **pirmą** (ne paskutinę) dokumento knygą (nepaisant to, kad paskutinis žingsnis grąžins visų knygų aibę išrikiuotą priešinga tvarka nei dokumentas (ašis yra einanti atgal))



Tipų konvertavimai

- Jei operacijos operandai yra skirtingų tipų, tipų konvertavimas atliekamas automatiškai/priverstinai
 - Niekada nebus klaidų pranešimų dėl blogų tipų
- Pavyzdys: išraiška $3 > 2 > 1$
 - Operacijos apdorojamos visada iš kairės į dešinę, taigi perrašome kaip $(3 > 2) > 1$
 - $(3 > 2)$ grąžina *true*, gauname *true* > 1
 - Operacija > reikalauja, kad abu operandai būtų skaičiai, taigi *true* yra automatiškai verčiamas į skaičių kviečiant funkciją `number(true)`, kuri grąžina 1, gauname: $1 > 1$
 - Atsakymas: *false*
- Uždavinukai: koks bus rezultatas:
 - $1 < 2 < 3$
 - `"kuku" = 1 + //knyga - (1 < 2)`



Tipų konvertavimai

- XPath turi 4 tipus:
 - Struktūrinį:
 - aibė
 - Paprastus:
 - loginis (boolean),
 - eilutė (string)
 - skaičius (number)
- Aibės tipą galima paversti į bet kokį paprastą tipą
- Bet kokį paprastą tipą galima paversti į bet kokį kitą paprastą tipą
- Vertimams (konversijai) yra trys funkcijos:
 - `boolean()`, `string()`, `number()`



Funkcija `boolean()`

- Argumentas verčiamas į loginę konstantą *true* arba *false* tokiu būdu:
 - mazgų aibė verčiama į *true*, jei ji ne tuščia
 - Pvz.: `boolean(//knyga)` grąžins *true* jei dokumente yra bent viena knyga; priešingu atveju – *false*
 - skaičiai: nulis ir NaN (not a number) verčiami į *false*, kiti skaičiai – *true*
 - Pvz.: `boolean(5)` grąžins *true*
 - eilutė verčiama į *true*, jei jos ilgis > 0
 - Pvz.: `boolean("kuku")` grąžins *true*
 - `boolean("")` grąžins *false*



Funkcija `string()`

- Jei argumentas yra mazgų aibė:
 - Jei ji tuščia, grąžinama tuščia eilutė (ilgis 0)
 - Jei netuščia, imamas pirmas aibės elementas (mazgas) ir jei tai yra:
 - žymė, tai imami jos visi įpėdiniai tekstiniai mazgai (`descendant::text()`) ir jie sujungiami į vieną eilutę
 - atributas, tai imama jo reikšmė kaip eilutė
 - tekstinis mazgas, tai imama jo reikšmė
- Jei argumentas yra skaičius:
 - NaN yra verčiamas į eilutę "NaN"
 - teigiama begalybė verčiama į "Infinity"
 - neigiama begalybė verčiama į "-Infinity"
 - kiti skaičiai – į natūralų jų atvaizdavimą eilute
- Jei argumentas yra loginė konstanta:
 - *true* verčiamas į "true",
 - *false* verčiamas į "false"
- Jei argumentas praleistas, automatiškai sukuriamas mazgų aibė, `self::node()` ir paduodama kaip argumentas



Funkcija `number ()`

- Argumentas verčiamas į skaičių tokiu būdu:
 - eilutė verčiama į artimiausią matematinę skaitinę išraišką; jei paversti į skaičių neįmanoma (ne skaičius), tai grąžinamas skaičius NaN (not a number)
 - loginis *true* verčiamas į 1, *false* verčiamas į 0
 - mazgų aibė pirmiausia paverčiama į eilutę kviečiant funkciją `string ()`, tada verčiama į skaičių kaip nusakyta pirmame punkte
- Jei argumentas praleistas, automatiškai sukuriamas mazgų aibė `self::node ()` ir paduodama kaip argumentas



Loginės išraiškos

- Tai išraiškos, naudojančios operatorius `or`, `and`, funkcijas `true()`, `false()`, `not()` bei palyginimo operacijas `<`, `<=`, `>`, `>=`
- Jei operatorių `or` ir `and` argumentai nėra loginio tipo, jei priverstinai paverčiami į loginį tipą, kviečiant funkciją `boolean()`
- Pvz.:
 - `/descendant::knyga[1]/kaina > 50 and /descendant::žurnalas[2]/kaina < 30`
 - `//knyga or 0 or "kuku,,`
 - Ar aibė tuščia: `boolean(//kuku) = false()`



Lyginimo operacijos $<$, $<=$, $>$, $>=$ be aibių

- Jei lyginami operandai yra ne aibės
 - Pvz.: `true() > „kuku“`
- Tai:
 - abu operandai yra verčiami į skaičius kviečiant funkciją `number()`

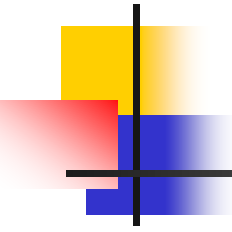
Lyginimo operacijos $<$, $<=$, $>$, $>=$ su aibėmis (1)

■ Pavyzdžiai

- `//knyga > 5`
- `//knyga <= //žurnalas`

■ Sutrumpinimai:

- A – aibė (aibė gali susidėti tik iš dokumento mazgų, negali susidėti iš skaičių, eilučių, loginių reikšmių)
- m – mazgas (pvz.: `/`, žymė, tekstas, atributas)
- l – loginė reikšmė (*boolean*)
- s – skaičius (*number*)
- e – eilutė (*string*)
- ° - lyginimo operacija $\{<, <=, >, >=\}$



Lyginimo operacijos $<$, $<=$, $>$, $>=$ su aibėmis (2)

- Atsikratome aibių, perrašydami taip:
 - $A_1 \circ A_2 \leftrightarrow \exists m_1 \in A_1 \text{ ir } \exists m_2 \in A_2 : \text{number}(m_1) \circ \text{number}(m_2)$
 - $A_1 \circ s \leftrightarrow \exists m_1 \in A_1 : \text{number}(m_1) \circ s$
 - $A_1 \circ e \leftrightarrow \exists m_1 \in A_1 : \text{number}(m_1) \circ \text{number}(e)$
 - $A_1 \circ l \leftrightarrow \text{number}(\text{boolean}(A_1)) \circ \text{number}(l)$
- Pastaba: pirma trys punktai atlieka aibių Dekarto sandaugą – imame visas galimas poras, jei bent viena pora duoda rezultatą true, tai galutinis rezultatas irgi true.



Lyginimo operacijos = ir != be aibių

- Jei lyginami operandai yra ne aibės
 - Pvz.: 5 = „kuku“
- Tai:
 - Jei bent vienas operandas yra loginio tipo, tai kitas operandas yra paverčiamas į loginį tipą kviečiant funkciją boolean()
 - Priešingu atveju, jei bent vienas operandas yra skaičius, tai kitas operandas yra verčiamas į skaičių kviečiant funkciją number()
 - Priešingu atveju abu operandai yra verčiami į eilutes kviečiant funkciją string()

Lyginimo operacijos = ir != su aibėmis (1)

■ Pavyzdžiai

- `//knyga != 5`
- `//knyga = //žurnalas`

■ Sutrumpinimai:

- A – aibė (aibė gali susidėti tik iš dokumento mazgų, negali susidėti iš skaičių, eilučių, loginių reikšmių)
- m – mazgas (pvz.: /, žymė, tekstas, atributas)
- l – loginė reikšmė (*boolean*)
- s – skaičius (*number*)
- e – eilutė (*string*)
- ° - lyginimo operacija {=, !=}

Lyginimo operacijos = ir != su aibėmis (2)

- Atsikratome aibių, perrašydami taip:
 - $A_1 \circ A_2 \leftrightarrow \exists m_1 \in A_1 \text{ ir } \exists m_2 \in A_2 : \text{string}(m_1) \circ \text{string}(m_2)$
 - $A_1 \circ s \leftrightarrow \exists m_1 \in A_1 : \text{number}(m_1) \circ s$
 - $A_1 \circ e \leftrightarrow \exists m_1 \in A_1 : \text{string}(m_1) \circ e$
 - $A_1 \circ l \leftrightarrow \text{boolean}(A_1) \circ l$
- Pastaba: pirma trys punktai atlieka aibių Dekarto sandaugą – imame visas galimas poras, jei bent viena pora duoda rezultatą true, tai galutinis rezultatas irgi true.



Naudingos funkcijos

- *number* **last**()
 - Grąžina konteksto dydį (šiuo metu predikato apdorojamos mazgų aibės elementų skaičių)
- *number* **position**()
 - Grąžina einamojo mazgo poziciją predikato apdorojamoje mazgų aibėje, atsižvelgiant į ašies ėjimo kryptį
- *number* **count**(*mazgų aibė*)
 - Grąžina argumento (mazgų aibės) elementų skaičių
- *number* **sum**(*mazgų aibė*)
 - Pirmiausia kiekvienas argumento (mazgų aibės) elementas verčiamas į eilutę (kviečiant `string()` funkciją), tada kiekviena eilutė verčiama į skaičių (kviečiant `number()` funkciją), ir visi skaičiai susumuojami
- *boolean* **not**(*boolean*)
 - Atlieka neigimo operaciją
- *string* **concat**(*string*, *string*, *string**)
 - Atlieka parametrų konkatenciją (eilučių apjungimą)



Naudingos funkcijos

- *string* **normalize-space**(*string*?)
 - Ištrina eilutės–parametro pradžioje ir pabaigoje esančius tarpus (ir kitus nematomus simbolius – whitespace) bei viduje esančių tarpų grupes keičia vienu tarpu, ir grąžina gautą rezultatą



Uždavinys – unikalumo tikrinimas

- Sakykime, atributą `id` galima rašyti bet kurioje dokumento žymėje. Kaip patikrinti, ar visame dokumente visų šių atributų reikšmės unikalios?
 - "Špargalkė": jei operacijos "=" abu operandai yra mazgų aibės, tai ji padaro tų aibių Dekarto sandaugą ir lygina kiekvieną porą atskirai. Jei bent vienai porai būna lygu, tai ir atsakymas `true`.



Atsakymas

Kyla noras parašyti taip (kairėje ir dešinėje visų dokumento atributų `id` aibės):

```
//@id = //@id
```

Tačiau čia tas pats atributas kairėje bus palygintas su savimi pačiu dešinėje, ir bus blogai. Išėjis – atsistojus "ant" kažkokios einamosios žymės, pereiti visas kitas žymes, einančias dokumente žemiau jos, ir palyginti jų atributus su jos atributu:

```
//*[ @id = following::* /@id]
```

Ši išraiška grąžins žymes, kurių atributas `id` turi reikšmę, pasikartojančią kažkur žemiau dokumente.

Bet čia liko dar viena problemėlė, kad `following` neima žymių, kurios yra einamosios žymės įpėdiniai, todėl pridedam dar ir `descendant`:

```
//*[ @id = following::* /@id or @id=descendant::* /@id]
```



Uždavinys – didžiausio/mažiausio paieška

- Rasti mažiausią knygos kainą
- `//knyga[kaina < //knyga/kaina]`
 - Deja neveikia – operatorius `<` grąžina, ar kaina yra mažesnė už bent vieną kitą kainą, o mums reikia, kad būtų mažesnė už visas kitas
- Rasti kainą, kuri nėra didesnė nei už vieną kitą kainą:
- `//knyga[not(kaina > //knyga/kaina)]`