UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Elektronske lekcije o XML formatu podataka i upotrebi XPath i XQuery izraza za obilazak stabla XML dokumenta

MASTER RAD

Student
Jelena Ilić

Mentor
dr Miroslav Marić

Beograd
17.9.2020.

Sadržaj

Uvod ... 3
1 **Elektronske lekcije** ... 4
2 **Struktura XML dokumenta** ... 7
2.1 „Stablo“ XML dokumenta ... 7
2.2 Deklaracija ... 9
2.3 Komentari ... 9
2.4 Instrukcije za obradu ... 9
3 **Etikete** ... 10
3.1 Sintaksa etiketa ... 10
4 **Elementi** ... 11
4.1 Imenovanje elemenata ... 11
4.2 Proširivost XML dokumenata ... 12
4.3 Korenski element ... 13
5 **Reference entiteta** ... 14
6 **Atributi** ... 15
7 **Prostori imena** ... 18
7.1 Praktična primena prostora imena ... 19
8 **Prikaz XML dokumenta** ... 22
8.1 Prikaz XML dokumenta sa greškom ... 23
9 **Validacija XML dokumenta** ... 24
10 **Definicija tipa dokumenata - XML DTD** ... 26
10.1 Implementacija DTD dokumenta ... 26
10.2 Deklarisanje elemenata ... 28
10.3 Deklarisanje atributa ... 29
10.4 Deklarisanje entiteta ... 31
11 **XML šema** ... 33
11.1 Deklarisanje elemenata ... 33
11.1.1 XML elementi prostog tipa ... 33
11.1.2 XML elementi složenog tipa ... 34
11.2 Deklarisanje atributa ... 34
11.3 Implementacija XML Scheme ... 35
12 **Upotreba i struktura XPath izraza** ... 38
12.1 Čvorovi XML dokumenta ... 38
12.2 XPath operatori ... 39

12.3 XPath izrazi 39
12.3.1 Lokacione putanje 40
12.3.2 Logički izrazi 43
12.3.3 Znakovni nizovi 43
12.3.4 Brojevni izrazi 43
12.3.5 „Džokerski“ znakovi 43
12.3.6 Pronalaženje više skupova čvorova različite vrste istovremeno 43
12.4 XPath funkcije 44
12.4.1 Funkcije skupa čvorova 45
12.4.2 Funkcije znakovnih nizova 45
12.4.3 Logičke funkcije 46
12.4.4 Numeričke funkcije 46
13 **Upotreba i struktura XQuery izraza** 47
13.1 XQuery izrazi 47
13.1.1 Izrazi putanja 48
13.1.2 FLWOR izrazi 49
13.1.3 Konstruktori elemenata 50
13.1.4 Uslovni izrazi 51
13.1.5 Kvantifikovani izrazi 52
13.1.6 Izrazi u kojima se koriste operatori 52
13.1.7 Izrazi u kojima se koriste funkcije 53
Zaključak 54
Literatura 55

Uvod

Predstavljajući naizgled neiscrpan izvor informacija, sredstvo komunikacije i sastavni deo posla ljudi širom sveta, internet je postao neophodna karika ljudske svakodnevnice. To dovodi do neprestanog napretka veb tehnologija za izradu i razvoj različitih veb aplikacija. Iz potrebe za sticanjem praktičnog znanja iz ove oblasti, nastale su brojne platforme za učenje veb tehnologija. Jedna od njih je i elektronska platforma „eŠkola veba“ na kojoj se nalaze kursevi različitih veb tehnologija. Platforma je javno dostupna na adresi **http://edusoft.matf.bg.ac.rs/eskola_veba** [1]. Kursevi se sastoje iz elektronskih lekcija. Ovaj rad je namenjen elektronskim lekcijama o XML formatu podataka i upotrebi XPath i XQuery izraza za obilazak stabla XML dokumenta.

XML (eXtensible Markup Language) je standard kojim se generiše struktura podataka koje treba čuvati u memoriji, razmeniti ili dalje obrađivati. Nastao je iz potrebe za postojanjem standarda za obeležavanje koji je jednostavan za parsiranje i obradu. Preporučen je od strane W3C (World Wide Web Consortium) 1998. godine.

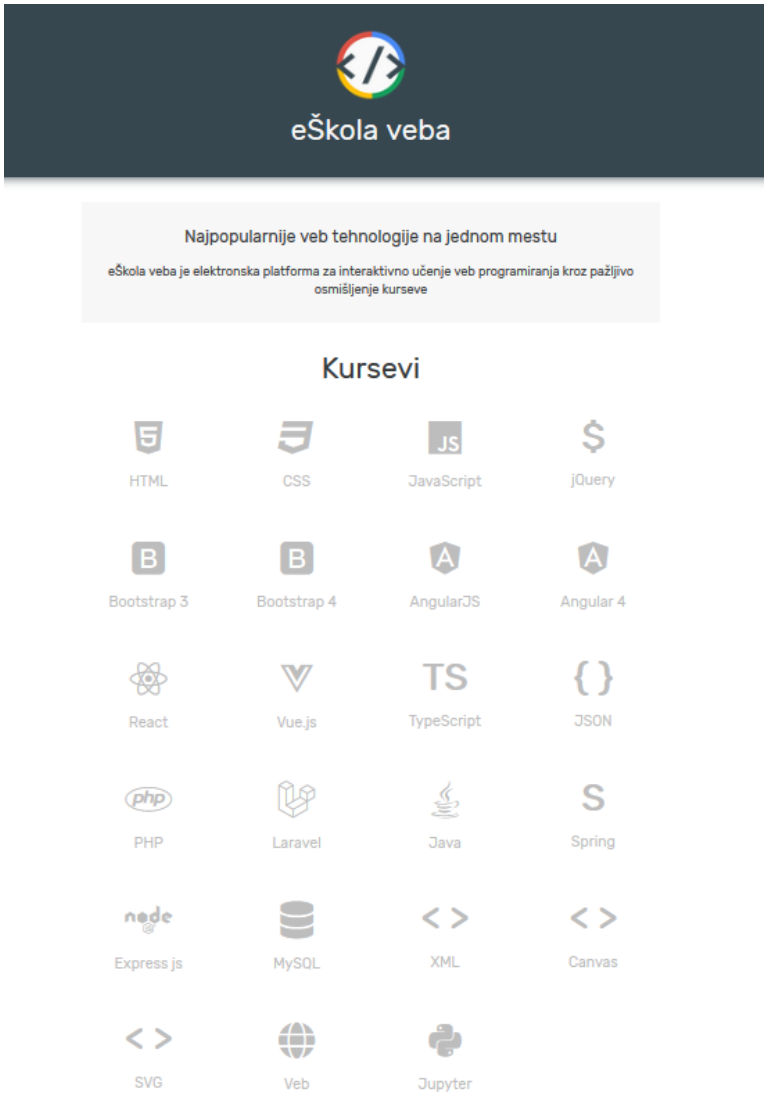
Ovaj standard nudi mogućnost obeležavanja i opisivanja podataka lako razumljivim i jednostavnim oznakama (etiketama). Na taj način, informacije se mogu lakše identifikovati i koristiti. XML je veoma važan u današnjem svetu informacionih tehnologija pre svega zbog svoje jednostavnosti i fleksibilnosti i kao takav predstavlja neizostavan deo veb tehnologija. Većina veb aplikacija koristi upravo XML za čuvanje i razmenu podataka. Važna karakteristika XML formata podataka jeste proširivost. Zahvaljujući toj osobini, pogodan je za korišćenje u različitim oblastima jer se može prilagoditi terminologiji i formi koju data oblast zahteva. Stoga, osim u informacionim tehnologijama, XML ima primenu i u multimediji, robotici, pravu, finansijama, novinarstvu, fizici, ekonomiji i još mnogim oblastima. XML je standard za meta označavanje. To znači da ne postoji unapred definisan konačan skup etiketa i elemenata koji se mogu koristiti za pisanje XML dokumenata, već je autorima data sloboda da svoje oznake i elemente kreiraju i nazovu u skladu sa potrebama svog rada.

XML datoteke su nezavisne od platforme na kojoj se čitaju ili kreiraju, što olakšava razmenu podataka kroz različite informacione sisteme.

U ovom radu, detaljno će biti objašnjena struktura XML dokumenta i njegovi gradivni blokovi: etikete, elementi, atributi, prostori imena, instrukcije za obradu, komentari. Biće obrađena sintaksna pravila koja se moraju ispoštovati da bi XML dokument bio dobro formiran. Potom će pažljivo biti razmotrena validacija XML dokumenata uz opis tehnologija za specifikovanje strukture XML dokumenata: XML DTD i XML Schema. Pored upoznavanja sa osnovnim karakteristikama XML formata podataka, biće obrađeni i XPath i XQuery izrazi za obilazak stabla XML dokumenta. Pomenuti izrazi će biti ilustrovani odgovarajućim primerima. Za potrebe rada kreirane su odgovarajuće elektronske lekcije namenjene budućim veb programerima. Nalaze se na platformi „eŠkola veba“ i namenjene su budućim veb programerima i svima onima koji prvi put dolaze u kontakt sa XML standardom. Lekcije su dostupne na adresi: **http://edusoft.matf.bg.ac.rs/eskola_veba/#/course-details/xml**.

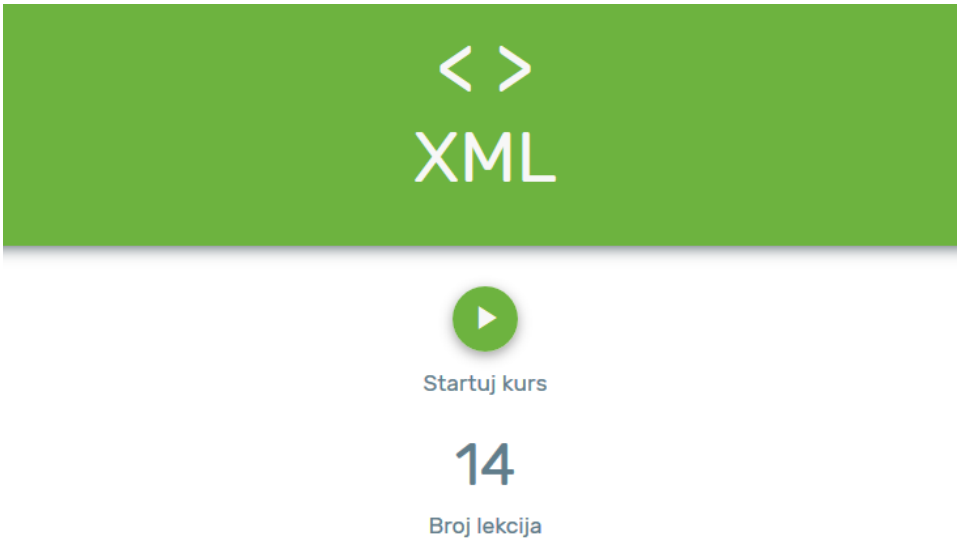
1 Elektronske lekcije

Elektronske lekcije o XML formatu podataka i upotrebi XPath i XQuery izraza za obilazak stabla XML dokumenta imaju za cilj da olakšaju savladavanje osnovnih pojmova XML standarda. Napisane su na srpskom jeziku, a uz to su besplatne i javno dostupne. Može im se pristupiti putem linka: http://edusoft.matf.bg.ac.rs/eskola_veba/#/course-details/xml. Izgled početne strane platforme „eŠkola veba“, na kojoj se lekcije nalaze, prikazan je na slici 1.



Slika 1: eŠkola veba

Kao što se može primetiti, na platformi se nalaze kursevi za učenje različitih veb tehnologija. Među njima je i kurs posvećen XML standardu, kreiran za potrebe ovog rada. Izborom ovog kursa otvara se početna strana čiji izgled prikazuje slika 2.

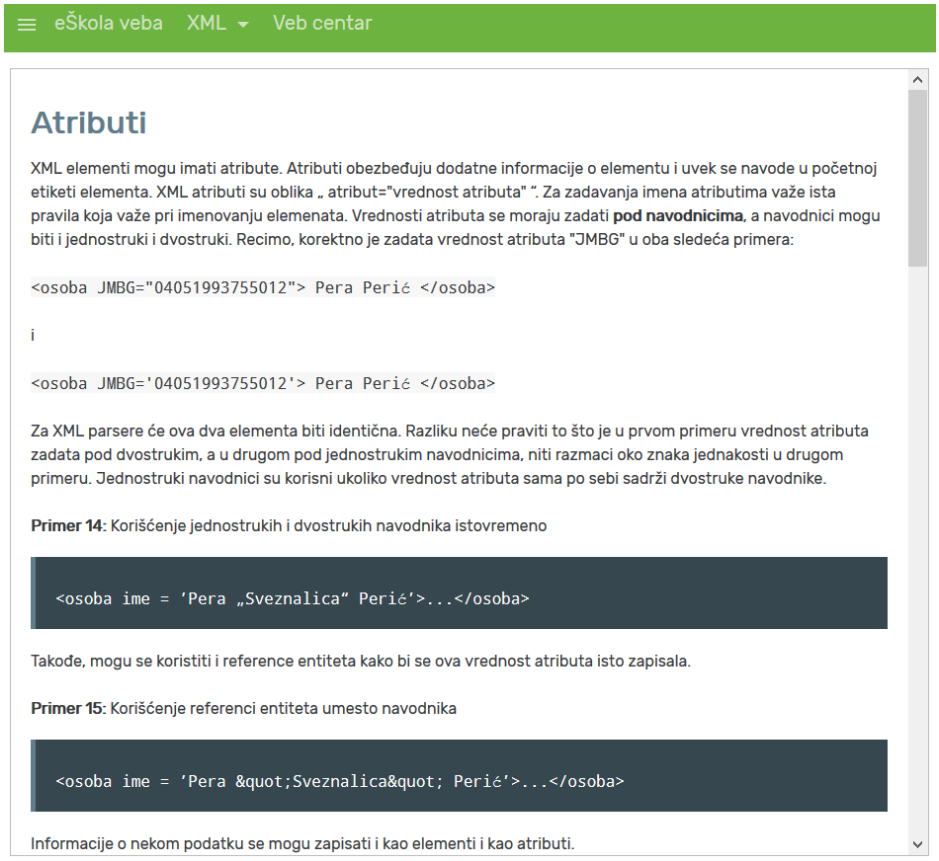


Slika 2: Naslovna strana kursa

Lekcije koje čine ovaj kurs su:

- Uvod u XML;
- Struktura stabla;
- Etikete;
- Elementi;
- Reference entiteta;
- Atributi;
- Prostori imena;
- Deklaracija, komentari i instrukcije za obradu;
- Prikaz;
- Validacija dokumenata;
- Definicija tipa dokumenta - XML DTD;
- XML Šema;
- XPath izrazi;
- XQuery izrazi.

Svaka lekcija sadrži definicije i detaljne opise ključnih pojmova. Sve strukture, pravila i upitni izrazi koji se obrađuju ilustrovani su velikim brojem kraćih primera. Primeri su detaljno objašnjeni u cilju dobrog razumevanja konkretne teme. Slika 3 pokazuje izgled lekcije iz XML kursa, a slika 4 prikazuje jedan primer iz lekcije.



Slika 3: Izgled lekcije

Primer 6: Sadržaj elementa

```
<bioskop>
       <film zanr="komedija">
         <naziv> Ghostbusters </naziv>
         <reditelj>Ivan Reitman</reditelj>
         <trajanje>1h 45min</trajanje>
         <cena>300 dinara</cena>
       </film>
</bioskop>
```

Slika 4: Primer iz lekcije

2 Struktura XML dokumenta

XML dokumenti sadrže podatke u obliku znakovnih niski, obeležene tekstualnim etiketama. Etikete počinju znakom „<“ i završavaju se znakom „>“, a između pomenutih znakova se navode njihova imena. Postoje početne i završne etikete. Svaka početna etiketa ima odgovarajuću završnu koja je nazvana istim imenom i navodi se između znakova „</“ i „>“. Primeri nekih etiketa su <element>, <osoba>, <naziv>, <karakteristika> i slično. One podsećaju na HTML etikete, ali je razlika u tome što HTML ima definisan skup etiketa koje se smeju koristiti, na primer : <p> - paragraf, <table> - tabela, <ul> - neuređena lista (eng. unordered list), itd. XML sintaksa omogućava meta označavanje, odnosno daje slobodu autorima da sami kreiraju etikete.

2.1 „Stablo“ XML dokumenta

XML dokumenti su formirani u obliku „stabla elemenata“. XML element čine jedna početna etiketa, odgovarajuća završna etiketa i sadržaj koji se nalazi između njih. „Stablo“ počinje od korenskog elementa i grana se preko ostalih elemenata „roditelja“, „dece“, sve dalje do, može se reći, elemenata „listova“. Svi elementi imaju svoj tekstualni deo i mogu sadržati druge elemente (podelemente) i atribute, o čemu će više reči biti kasnije. Nazivi „element roditelj“, „element dete“ i slično, koriste se da bi se objasnila veza između elemenata u XML stablu. Element „roditelj“ je onaj koji ima svoje podelemente „decu“. Svaki element može imati svoje podelemente. „Braća“ i „sestre“ elementi su elementi istog nivoa. Da bi ova struktura bila jasnija, ona se može i slikovito predstaviti. Uopšten primer strukture stabla XML dokumenta može se videti u primeru 1.

Primer 1: Struktura stabla XML dokumenta

```
<koren>
    <roditelj>
      <dete_brat>
        <dete_deteta>...</dete_deteta>
      </dete_brat>
      <dete_sestra>
        <dete_deteta>...</dete_deteta>
      </dete_sestra>
    </roditelj>
</koren>
```

Jednostavan XML dokument prikazan je u primeru 2.

Primer 2: Jednostavan XML dokument „obavestenje.xml“

```
<obavestenje>
        <od>Autor</od>
        <za>Čitalac</za>
        <naslov>Obaveštenje</naslov>
        <poruka>
          Počinje kurs. Srećno!
        </poruka>
</obavestenje>
```

Etikete u primeru 2: <obavestenje>, <od>, <za>, <naslov>, <poruka> nisu definisane XML-standardom, već su osmišljene od strane autora. Međutim, XML zadaje stroga sintaksna pravila po

kojima se dokumenti oblikuju. Jedno od tih pravila je i to da svaka početna etiketa mora imati odgovarajuću završnu etiketu.

XML dokumenti imaju samoopisivu strukturu, zahvaljujući etiketama. One ukazuju na vrstu podataka koje obeležavaju. Primera radi, dovoljno je pročitati etikete koje označavaju određeni element i primetiti da on predstavlja ime osobe, cenu ili datum. Dakle, iz same strukture se jasno vidi šta opisuje dati XML dokument i kakve informacije sadrži. Prema tome, može se odmah primetiti da XML dokument „bioskop.xml“ prikazan u primeru 3 sadrži informacije o repertoaru bioskopa.

Primer 3: XML dokument „bioskop.xml“

```
<?xml version="1.0" encoding="UTF-8"?>
    <bioskop>
      <film>
        <naziv> It </naziv>
        <reditelj>Andy Muschietti</reditelj>
        <trajanje>2h 15min</trajanje>
        <cena>350</cena>
      </film>
      <film>
        <naziv> Ghostbusters </naziv>
        <reditelj>Ivan Reitman</reditelj>
        <trajanje>1h 45min</trajanje>
        <cena>300</cena>
      </film>
      <film>
        <naziv> Interstellar </naziv>
        <reditelj> Christopher Nolan</reditelj>
        <trajanje>2h 49min</trajanje>
        <cena>400</cena>
      </film>
    </bioskop>
```

Na samom početku dokumenta navedena je XML deklaracija:

`<?xml version="1.0" encoding="UTF-8"?>`.

Naredna linija predstavlja početak stabla o kome je bilo reči. Zapravo, ova linija otvara korenski element „bioskop“. Svi elementi unutar elementa „bioskop“ su njegova „deca“ koja će sadržati sve potrebne informacije da bi se dovoljno detaljno opisao repertoar bioskopa. Prvi podelement (dete) elementa „bioskop“ je „film“. Primećuje se da u datom primeru postoje tri elementa pod istim tim nazivom i to su elementi istog nivoa („braća i sestre“ elementi). Svaki od njih ima četiri podelementa. Dakle, element „film“ je roditeljski element i ima četvoro dece: „naziv“, „reditelj“, „trajanje“ i „cena“.

Primer 4: Roditeljski element film i njegovi podelementi

```
<film>
          <naziv> Interstellar </naziv>
          <reditelj> Christopher Nolan</reditelj>
          <trajanje>2h 49min</trajanje>
          <cena>400 dinara</cena>
</film>
```

Može se primetiti da svaki element ima svoju početnu i završnu etiketu. Za razliku od HTML standarda, XML ne dozvoljava elemente kao što je
, koji nema završnu etiketu. Poslednja linija datog XML

dokumenta predstavlja kraj korenskog elementa. Tu se „bioskop“ zatvara i ovim se završava XML dokument.

S obzirom na to da je celokupan sadržaj (i podaci i etikete) XML dokumenata tekstualan, oni se mogu pisati pomoću uređivača teksta, koji su svim korisnicima lako dostupni. Važno je da kreirani XML dokumenti imaju ekstenziju „ .xml “. Više o kreiranju i strukturi XML dokumenata, ali i prenosu podataka iz XML dokumenata u već postojeće relacione baze podataka može se pročitati u [2].

2.2 **Deklaracija**

XML dokumenti najčešće počinju XML deklaracijom. Deklaracija može sadržati atribute *version, encoding* i *standalone*:

```
<?xml version="1.0" encoding="UTF-8"? standalone="yes">.
```

Atributom *version* zadaje se verzija XML standarda, atributom *encoding* znakovni kod (u ovom slučaju UTF-8), a atributom *standalone* ukazuje se na to da li XML dokument zavisi od nekog spoljašnjeg DTD dokumenta. Često se dešava da atribut *standalone* nije naveden u dekaraciji. Više o DTD dokumentima može se pročitati u glavi 10. XML deklaracija može postojati u XML dokumentu i ne mora, ali ako postoji, mora se prva navesti. Dakle, ispred nje se ne smeju nalaziti komentari, elementi, razmaci i slično. Podrazumevani znakovni kod za XML jeste UTF-8.

2.3 **Komentari**

Ono što je potpuno zajedničko za XML i HTML jeste sintaksa komentara. U oba standarda se navode na isti način.

```
<!-- Ovo je jedan komentar -->.
```

Važno je napomenuti da se unutar komentara ne smeju pojaviti dve crtice „--“ pre znaka za kraj „-->“. Takođe, nije dozvoljeno završiti komentar sa tri crtice „--->“. Kako komentari nisu elementi, njihova pozicija u dokumentu neće narušiti strukturu XML dokumenta. Prema tome, oni se mogu navesti u bilo kom delu dokumenta osim ispred deklaracije i unutar etiketa ili drugih komentara.

Komentare navodi autor ili koautori jednog zajedničkog XML dokumenta kako bi pratili svoj i međusobni rad, obeležili gde bi nešto moglo da se doda ili oduzme, naveli potrebna objašnjenja, napomene i slično. Oni nisu namenjeni programima koji koriste XML dokumente, već isključivo korisnicima, kako bi dati dokument ili neki njegov deo učinili razumljivijim.

2.4 **Instrukcije za obradu**

Nekada je potrebno aplikacijama koje koriste XML dokument proslediti posebne informacije. Za prosleđivanje informacija aplikacijama, XML koristi instrukcije za obradu. Znak za njihov početak je „<?“ a za kraj „?>“. Instrukcija za obradu sadrži ime aplikacije kojoj je namenjena i tekst zapisan u formatu koji odgovara toj aplikaciji. Instrukcije se, kao i komentari, mogu navesti bilo gde pre ili nakon korenskog elementa, ali ne ispred deklaracije, unutar etiketa ili komentara. Instrukcije za obradu mogu biti nazvane bilo kojim imenom koje poštuje pravila koja XML zadaje, jedini naziv koji nije dozvoljen je „XML“ u bilo kakvoj kombinaciji malih i velikih slova. Primer instrukcije za obradu koja govori robotima da li da indeksiraju određenu stranicu, preuzet iz [3]:

```
<? robots index="yes" follow="no" ?>.
```

3 Etikete

XML etikete dosta podsećaju na HTML etikete. Početna etiketa počinje znakom „<“ a završava se znakom „>“. Završna etiketa počinje znakom „</“ i završava se znakom „>“. Između pomenutih znakova se navodi ime etikete. Imena etiketa zadaje autor kako bi opisao sadržaj elementa koji one označavaju.

3.1 Sintaksa etiketa

Svaka početna XML etiketa mora imati odgovarajuću završnu etiketu. Dakle, za razliku od HTML standarda, XML ne dozvoljava etikete kao što su
 i <p> za koje ne mora da postoji odgovarajuća završna etiketa.

Iako je vrlo jednostavan, element „ime“ predstavlja primer potpunog i dobro formiranog XML elementa:

```
<ime> Pera </ime>.
```

Primećuje se da XML deklaracija nema završnu etiketu. Međutim, to nije pogrešno zato što se ona ne smatra delom XML dokumenta u strogom smislu.

XML etikete su osetljive na veličinu slova (eng. case sensitive). Prema tome, etikete <Ime>, <IME> i <ime> nisu iste. Ovo može predstavljati problem ukoliko se ne zadaju ispravno početna i završna etiketa istog elementa. Stoga bi sledeći element bio neispravno kreiran:

```
<Ime> Pera </ime>.
```

Dakle, autor može da bira da li će koristiti mala ili velika slova kada zadaje imena etiketa, ali prilikom kreiranja jednog elementa mora da vodi računa da početna i završna etiketa budu odgovarajuće zapisane.

Elementi moraju biti pravilno ugnježdeni. To znači da, ukoliko je neki element otvoren unutar nekog drugog, mora prvo da se zatvori unutrašnji element, a zatim spoljašnji, što pokazuje sledeći primer:

```
<film><naziv>Interstellar</naziv></film>.
```

XML ne dozvoljava sledeći način ugnježđivanja (koji se često može videti u HTML standardu):

```
<film><naziv>Interstellar</film></naziv>.
```

4 Elementi

Elementi su osnovni gradivni blokovi XML dokumenta. XML element je celina koju čine početna i završna etiketa i naveden sadržaj između njih.

Primer 5: Element

```
<element>Sve ovo je jedan element.</element>
```

Tekstualni sadržaj ovog elementa je „Sve ovo je jedan element.“ Važno je da razmaci (beline) takođe čine deo tekstualnog sadržaja i da se više uzastopnih razmaka neće sažeti u jedan. Dakle, ukoliko je tekst unet na sledeći način:

```
<element> Ovde se nalazi                    velika praznina </element>.
```

On će tako i ostati sačuvan u XML dokumentu.

Element može sadržati tekst, atribute i druge elemente.

Primer 6: Sadržaj elementa

```
<bioskop>
        <film zanr="komedija">
          <naziv>Ghostbusters</naziv>
          <reditelj>Ivan Reitman</reditelj>
          <trajanje>1h 45min</trajanje>
          <cena>300 dinara</cena>
        </film>
</bioskop>
```

Primećuje se da elementi „bioskop“ i „film“ sadrže druge elemente; element „film“ sadrži i atribut (zanr="komedija"); a elementi „naziv“, „reditelj“, „trajanje“, „cena“ sadrže samo tekst. Elemente koji nemaju sadržaj između početne i završne etikete nazivamo praznim elementima. Prazan element se može zapisati na dva načina, što je prikazano u primerima 7 i 8.

Primer 7: Prazan element

```
<prazno> </prazno>
```

Primer 8: Prazan element

```
<prazno />
```

Prazni elementi mogu imati atribute. Upravo u atributima su sadržane informacije koje prazni elementi nose.

4.1 Imenovanje elemenata

Ime elementa se zadaje odgovarajućim etiketama. XML imena mogu da sadrže sva velika i mala slova engleskog jezika (dakle sve karaktere od *a do z* i od *A do Z*) kao i cifre, 0 – 9. I ne samo to, dozvoljena su i neengleska slova i ideogrami, kao na primer: *š, π,ب* , itd. Što se znakova interpunkcije tiče, u imenima se mogu naći samo sledeći: „ . “ (tačka), „ - “ (crta) i „ _ “ (donja crta). Ostali znakovi interpunkcije nisu dozvoljeni.

Potrebno je obratiti pažnju na sledeća pravila prilikom imenovanja elemenata:

- Imena elemenata su osetljiva na veličinu slova (eng. case sensitive);
- Imena elemenata moraju početi slovom, ne brojem ili znakom interpunkcije;
- Imena elemenata ne smeju početi slovima *xml, XML, Xml* i slično;
- Imena elemenata ne smeju sadržati razmake niti znake za prelazak u novi red, vraćanje na početak reda i slično;
- Ne postoje rezervisani nazivi (osim *xml*), tako da bilo koja reč može biti naziv elementa.

Dobra praksa je davati kratke nazive koji dobro opisuju element. Na primer: <ime>, <prezime> <boja>, i slično. U slučaju da naziv mora da se sastoji iz više reči, najbolje je birati najkraći jasan opis. Recimo bolje je element nazvati <otvorena_vrata> umesto <vrata_koja_su_otvorena>. Iako su dozvoljeni, treba izbegavati karaktere kao što su "-", ".", ":" jer nazive koji sadrže ove karaktere neki softveri mogu pogrešno da protumače. Recimo, <ime.osobe> se može protumačiti kao da je „osobe“ svojstvo objekta „ime“; <ime-osobe> se može protumačiti kao oduzimanje osobe od imena; znak „:“ je rezervisan za prostore imena o kojima će biti reči kasnije. Takođe, treba biti oprezan sa specifičnim karakterima kao što su š,ć,č,đ i slični, jer ih neki softveri ne mogu pročitati.

Ne postoji stil koji je definisan za imenovanje elemenata, ali neki od predloga su: <imeosobe>, <IMEOSOBE>, <imeo_sobe>, <imeosobe>, <ImeOsobe>, <imeOsobe>. Za koji god stil se autor opredeli, trebalo bi da se njega drži kroz ceo dokument. Često za XML dokument postoji odgovarajuća baza, dobra praksa je elementima davati nazive po pravilu po kojem su oni nazvani u toj bazi.

4.2 Proširivost XML dokumenata

XML dokumenti mogu slobodno da se proširuju novim podacima i, isto tako, iz njih mogu da se brišu postojeći podaci, bez bojazni da će to izazvati grešku kod softvera koji koristi taj dokument. Proširivost XML dokumenata se može ilustrovati na primeru 9.

Primer 9: XML dokument „stan.xml“

```
<stan>
        <soba1> Kuhinja </soba1>
        <soba2> Kupatilo </soba2>
        <poruka> Dobro došli!</poruka>
 </stan>
```

Primera radi, neka postoji aplikacija koja iz ovog XML dokumenta može da izdvoji informacije o sobama i poruku, i interpretira to kao što je prikazano na slici 5.

Slika 5: Interpretacija XML dokumenta „stan.xml“

Može se desiti da autor XML dokumenta u nekom trenutku želi da doda još novih informacija. Na primer, još jednu sobu. Proširiće svoj XML dokument kao što pokazuje primer 10.

Primer 10: Proširen XML dokument „stan.xml“

```
     <stan>
          <soba1> Kuhinja </soba1>
          <soba2> Kupatilo </soba2>
          <soba3> Dnevni boravak </soba3>
          <poruka> Dobro dosli!</poruka>
      </stan>
```

Važno je da pri ovom dodavanju informacija XML dokumentu, aplikacija koja izdvaja i interpretira informacije neće prijaviti grešku. Zapravo, aplikacija će i dalje moći da pronađe i interpretira tražene elemente, bez obzira na informacije koje su naknadno dodate.

4.3 **Korenski element**

Svaki XML dokument mora imati korenski element (eng. root element). Ovaj element nema svoje roditeljske elemente, ali je on roditelj svim ostalim elementima u posmatranom XML dokumentu, tj. sadrži sve ostale elemente tog dokumenta. Nekada se on naziva i element dokumenta (eng. Document element). S obzirom na to da XML ima samoopisujuću sintaksu, na osnovu imena korenskog elementa se može uvideti šta opisuje posmatrani XML dokument. U primeru 11, korenski element XML dokumenta „osoba.xml“ je element „osoba“.

Primer 11: XML dokument „osoba.xml“

```
<?xml version="1.0" encoding="UTF-8"?>
<osoba>
          <ime> Pera </ime>
          <prezime> Perić </prezime>
          <datum_rođenja>
                          <dan> 1 </dan>
                          <mesec> januar</mesec>
                         <godina> 2001 </godina>
          </datum_rođenja>
</osoba>
```

Ostali elementi su deca elementa „osoba“ i ceo dokument iz primera 11 zapravo predstavlja informacije o toj osobi.

5 Reference entiteta

Neki karakteri imaju posebno značenje u XML standardu, stoga treba biti pažljiv prilikom korišćenja specijalnih karaktera. Recimo, ukoliko bi se unutar tekstualnog sadržaja elementa ili unutar vrednosti atributa naveo samo karakter „<“, došlo bi do greške jer ga XML parser prepoznaje kao oznaku za početak novog elementa. Takođe, karakter „&“ se ne sme navesti kao takav jer je on rezervisan za oznaku početka reference entiteta.

Greška opisana primerom 12 se može ispraviti na način koji prikazuje primer 13.

Primer 12: Neispravan zapis poređenja

```
<poređenje> 700 < 1500  </poređenje>
```

Primer 13: Ispravan zapis poređenja

```
<poređenje> 700 &lt; 1500  </poređenje>
```

U pravilno napisanom primeru 13 je korišćena referenca entiteta „<“ umesto specijalnog karaktera „<“. Reference entiteta imaju tri obavezna svoja dela, a to su ampersand (&), naziv entiteta i znak „tačka-zarez“ (;). Predefinisane reference entiteta u XML standardu su:

- < za znak „<“ (skraćeno od less than - manje od);
- > za znak „>“ (skraćeno od greater than - više od);
- & za znak „&“ (skraćeno od ampersand);
- ' za znak „ ' “ (skraćeno od apostrophe - apostrof);
- " za znak „ " “ (skraćeno od quotation mark - navodnici).

Osim navedenih referenci mogu da se koriste i numeričke i heksadekadne numeričke reference entiteta. Recimo, znak „<“ osim što se može navesti pomoću reference „<“, može se navesti i kao „&#60“ ili „&#x3C“. Svaku od ovih referenci će parser prilikom čitanja zameniti znakom „<“ i neće ih protumačiti kao početak etikete.

Samo karaktere „<“ i „&“ je striktno zabranjeno koristiti prilikom pisanja tekstualnog sadržaja elemenata i vrednosti atributa XML dokumenata, ali je dobra praksa koristiti reference entiteta i za ostale. Karakteri kao što su apostrof i navodnici koriste se za zadavanje vrednosti atributa, o čemu će reči biti kasnije.

Pomenute reference smeju da se navode samo unutar tekstualnog sadržaja elemenata i vrednosti atributa. Ne smeju se pojaviti u imenima elemenata i atributa. Takođe, ukoliko se reference navedu u komentaru, njih parser neće zameniti odgovarajućim znakom, jer on prepoznaje reference samo unutar sadržaja elemenata ili vrednosti atributa.

6 Atributi

XML elementi mogu imati atribute. Atributi obezbeđuju dodatne informacije o elementu i uvek se navode u početnoj etiketi elementa. XML atributi su oblika „ atribut="vrednost atributa" “. Za zadavanja imena atributima važe ista pravila koja važe pri imenovanju elemenata. Vrednosti atributa se moraju zadati pod navodnicima, a navodnici mogu biti i jednostruki i dvostruki. Recimo, korektno je zadata vrednost atributa "JMBG" u oba sledeća primera:

```
<osoba JMBG="04051993750012"> Pera  Perić </osoba>
```

i

```
<osoba JMBG = '04051993750012'> Pera Perić </osoba>.
```

Za XML parsere će ova dva elementa biti identična. Razliku neće praviti to što je u prvom primeru vrednost atributa zadata pod dvostrukim, a u drugom pod jednostrukim navodnicima, niti razmaci oko znaka jednakosti u drugom primeru. Jednostruki navodnici su korisni ukoliko vrednost atributa sama po sebi sadrži dvostruke navodnike.

Primer 14: Korišćenje jednostrukih i dvostrukih navodnika istovremeno

```
<osoba ime = ’Pera „Sveznalica“ Perić’>...</osoba>
```

Takođe, mogu da se koriste i reference entiteta kako bi se ova vrednost atributa isto zapisala.

Primer 15: Korišćenje referenci entiteta umesto navodnika

```
<osoba ime = ’Pera &quot;Sveznalica&quot; Perić’>...</osoba>
```

Informacije o nekom podatku se mogu zapisati i kao elementi i kao atributi.

Primer 16: Pol naveden kao atribut

```
<osoba pol= "muški">
        <ime> Pera </ime>
        <prezime> Perić </prezime>
</osoba>
```

Primer 17: Pol naveden kao element

```
<osoba>
        <pol> muški </pol>
        <ime> Pera </ime>
        <prezime> Perić </prezime>
</osoba>
```

U XML standardu ne postoje stroga pravila koja određuju kada treba koristiti elemente, a kada atribute, ali postoje važne razlike između elemenata i atributa koje naizgled elementima daju malu prednost. Na primer:

- Atributi ne mogu imati više od jedne vrednosti, dok elementi mogu;
- Atributi nemaju strukturu stabla, a elementi je imaju;
- Atributi nisu lako proširivi.

Sa druge strane, jedan element ne može imati više atributa istog imena što može predstavljati problem ukoliko je potrebno navesti više različitih vrednosti istoimenog atributa unutar jednog elementa. Recimo, potrebno je opisati osobu koja ima više zanimanja ili brojeva telefona i slično. Prema tome, na

autoru je da sam izabere kada će koristiti elemente, a kada atribute. U primerima 18, 19 i 20, elementi nose iste informacije zadate na različite načine.

Primer 18: Datum je zadat kao vrednost atributa

```
<obavestenje datum="14.7.2019.">
        <od>Autor</od>
        <za>Čitalac</za>
        <naslov>Obaveštenje</naslov>
        <poruka>
          Počinje kurs. Srećno!
        </poruka>
 </obavestenje>
```

Primer 19: Datum zadat kroz elemente „dan“, „mesec“ i „godina“

```
  <obavestenje>
      <dan>14</dan>
      <mesec>7</mesec>
      <godina>2019</godina>
      <od>Autor</od>
      <za>Čitalac</za>
      <naslov>Obaveštenje</naslov>
      <poruka>
        Počinje kurs. Srećno!
      </poruka>
  </obavestenje>
```

Svi elementi se mogu zameniti odgovarajućim atributima, kao u primeru 20, ali to bi trebalo izbegavati.

Primer 20: Svi podaci zadapisani kao atributi korenskog elementa

```
<obavestenje dan="14" mesec="7" godina="2019" od="Autor" za="Čitalac"
naslov="Obaveštenje" poruka="Počinje kurs. Srećno!">
</obavestenje>
```

Korisno je dodatne informacije o podacima navoditi kao atribute, a glavne podatke kao elemente. Na primer, ukoliko je potrebno identifikovati različite elemente unutar istog XML dokumenta, može se navesti atribut „id“. Taj atribut ne bi nosio informacije o osobinama elementa unutar kog se nalazi već bi one bile korisnije softveru koji koristi dati XML dokument.

Primer 21: Primena atributa

```
<komunikacija>
<obavestenje datum="14.7.2019." id = „1“>
        <od>Autor</od>
        <za>Čitalac</za>
        <naslov>Obaveštenje</naslov>
        <poruka>
          Počinje kurs. Srećno!
        </poruka>
 </obavestenje>
 <obavestenje datum="14.7.2019." id = „2“>
        <od>Čitalac</od>
        <za>Autor</za>
        <naslov>Re: Obaveštenje</naslov>
        <poruka>
```

```
          Hvala na obaveštenju!
        </poruka>
 </obavestenje>
</komunikacija>
```

7 Prostori imena

Autori XML dokumenata imaju slobodu da sami zadaju nazive elementima, u skladu sa pojmovima koje opisuju. Interesantan problem stvaraju homonimi – reči koje se isto pišu a imaju različito značenje. Na primer, u srpskom jeziku reč „sto“ može da znači broj „100“ ili deo nameštaja, reč „jezik“ može da označava govorni organ ili sistem glasova i simbola koji se koriste za komunikaciju. U engleskom jeziku reč „table“ može da znači tabela ili sto (komad nameštaja), i slično. Prema tome, može se desiti da se u XML dokumentu nađu potpuno različiti elementi nazvani istim imenom, što otežava razumevanje dokumenta i ljudima i aplikacijama koje ih koriste.

Primer 22: Vremenska nepgoda

```
    <grad>
        <prečnik>3cm</prečnik>
        <trajanje_nepogode>2h 35min</trajanje_nepogode>
        <naneta_šteta>700 evra</naneta_šteta>
    </grad>
```

Primer 23: Naseljeno mesto

```
    <grad>
        <naziv>Beograd</naziv>
        <zemlja>Srbija</zemlja>
        <broj_stanovnika>1.659.440</broj_stanovnika>
    </grad>
```

Može se primetiti da su u primerima 22 i 23 oba elementa nazvana istim imenom ali predstavljaju potpuno različite tipove podataka. U primeru 22, element „grad“ predstavlja vremensku nepogodu, dok u primeru 23 „grad“ predstavlja naseljeno mesto. Kada bi se ova dva XML elementa našla u istom XML dokumentu, došlo bi do konflikta zbog njihovih istih imena. Parser bi na isti način tretirao oba elementa jer ne bi mogao da vidi razliku u vrsti podataka. Da bi se takvi konflikti izbegli, koriste se takozvani prostori imena (eng. namespace) i njima odgovarajući prefiksi. Prostor imena omogućava razlikovanje istoimenih elemenata i atributa, ali i grupisanje svih elemenata i atributa sa zajedničkim značenjem. Definiše se atributom *xmlns* koji se navodi u početnoj etiketi elementa, na sledeći način:

```
xmlns:prefiks = "URI".
```

Prefiks prostora imena zadaje autor kao kratko obeležje koje će se nalaziti ispred naziva svih elemenata i atributa koji pripadaju istom prostoru imena. URI je identifikator prostora imena. Zadat URI ne služi parseru da pronađe potrebne informacije, već je njegova uloga da zada jedinstveno značenje odgovarajućem prostoru imena i svim elementima i atributima koji mu pripadaju. Najčešće se za URI prostora imena koristi link ka veb stranici koja sadrži informacije koje opisuju odgovarajući objekat.

Prema tome, problem istoimenih elemenata koji predstavljaju različite pojmove može da se reši pomoću prostora imena (primer 24).

Primer 24: Primena prostora imena

```
 <zajedno>

  <nepogoda:grad xmlns:nepogoda="http://www.izvor.com/VremenskaNepogoda">
       <nepogoda:prečnik>3cm</nepogoda:prečnik>
       <nepogoda:trajanje_nepogode>2h 35min</nepogoda:trajanje_nepogode>
       <nepogoda:naneta_šteta>700 evra</nepogoda:naneta_šteta>
```

```
  </nepogoda:grad>
  <mesto:grad xmlns:mesto="http://www.izvor.com/NaseljenoMesto">
       <mesto:naziv>Beograd</mesto:naziv>
       <mesto:zemlja>Srbija</mesto:zemlja>
       <mesto:broj_stanovnika>1.659.440</mesto:broj_stanovnika>
  </mesto:grad>

</zajedno>
```

Atributom *xmlns* su prefiksima „nepogoda“ i „grad“ definisani prostori imena u početnim etiketama odgovarajućih elemenata. Svi podelementi koji imaju isti prefiks će pripadati odgovarajućem prostoru imena i imati isto značenje. Postoji i lakši način, prostori imena se za sve prefikse koji se javljaju u istom XML dokumentu mogu definisati i u korenskom elementu.

Primer 25: Svi prostori imena definisani u korenskom elementu

```
<zajedno2 xmlns:nepogoda="http://www.izvor.com/VremenskaNepogod"
          xmlns:mesto="http://www.izvor.com/NaseljenoMesto">
  <nepogoda:grad>
      <nepogoda:prečnik>3cm</nepogoda:prečnik>
      <nepogoda:trajanje_nepogode>2h 35min</nepogoda:trajanje_nepogode>
      <nepogoda:naneta_šteta>700 evra</nepogoda:naneta_šteta>
  </nepogoda:grad>
  <mesto:grad>
      <mesto:naziv>Beograd</mesto:naziv>
      <mesto:zemlja>Srbija</mesto:zemlja>
      <mesto:broj_stanovnika>1.659.440</mesto:broj_stanovnika>
  </mesto:grad>
</zajedno2>
```

Može se definisati i takozvani podrazumevani prostor imena (eng. default namespace).

Sintaksa je:

`xmlns="URI".`

Definicija podrazumevanog prostora imena u početnoj etiketi posmatranog elementa, omogućava da se bez upotrebe prefiksa označi da svi njegovi podelementi pripadaju tom prostoru imena.

7.1 Praktična primena prostora imena

Da bi se na dobar način prikazala korisnost primene prostora imena, potrebno je u makar u najkraćim crtama upoznati se sa jezikom XSLT. XSL je akronim od „eXtensible Stylesheet Language“, a XSLT predstavlja „XSL Transformacije“. Ovaj jezik se koristi za transformisanje XML dokumenta u neki drugi format (npr. u HTML, što je prikazano u primeru 26). Primera radi, pretpostavimo da je potrebno prikazati tabelarno sve podatke iz dokumenta „bioskop.xml“.

Primer 26: XML dokument „bioskop.xml“

```
<?xml version="1.0" encoding="UTF-8"?>
      <bioskop>
        <film zanr="komedija">
          <naziv> Ghostbusters </naziv>
          <reditelj>Ivan Reitman</reditelj>
```

```
        <trajanje>1h 45min</trajanje>
        <cena>300</cena>
      </film>
      <film zanr="horor">
        <naziv> It </naziv>
        <reditelj>Andy Muschietti</reditelj>
        <trajanje>2h 15min</trajanje>
        <cena>350 </cena>
      </film>
      <film zanr="fantastika">
        <naziv> Interstellar </naziv>
        <reditelj> Christopher Nolan</reditelj>
        <trajanje>2h 49min</trajanje>
        <cena>400</cena>
      </film>
    </bioskop>
```

U cilju da repertoar bioskopa bude prikazan u obliku tabele, može se formirati odgovarajući XSLT dokument (bioskop.xsl) kojim će se definisati željeni izgled XML dokumenta o repertoaru bioskopa. Taj dokument mora sadržati prostor imena „http://www.w3.org/1999/XSL/Transform“ koji identifikuje XSLT elemente u HTML dokumentu.

Primer 27: XSLT dokument „bioskop.xsl“

```
<?xml version="1.0" encoding="UTF-8"?>
        <xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
          <html>
          <body>
          <h1>Repertoar bioskopa</h1>
          <table border="1">
            <tr bgcolor="#cc33ff">
              <th>Naziv</th>
              <th>Reditelj</th>
              <th>Trajanje</th>
              <th>Cena</th>
            </tr>
            <xsl:for-each select="bioskop/film">
            <tr>
              <td><xsl:value-of select="naziv"/></td>
              <td><xsl:value-of select="reditelj"/></td>
              <td><xsl:value-of select="trajanje"/></td>
              <td><xsl:value-of select="cena"/></td>
            </tr>
            </xsl:for-each>
          </table>
          </body>
          </html>
        </xsl:template>
        </xsl:stylesheet>
```

Ostalo je još samo da se u XML dokument „bioskop.xml“ unese referenca ka napravljenom XSLT dokumentu „bioskop.xsl“:

`<?xml-stylesheet type="text/xsl" href="bioskop.xsl"?>`.

Nakon toga dokument „bioskop.xml“ ima izgled prikazan u primeru 28.

Primer 28: XML dokument „bioskop.xml“ sa referencom ka XSLT dokumentu

```
<?xml version="1.0" encoding="UTF-8"?>
        <?xml-stylesheet type="text/xsl" href="bioskop.xsl"?>
        <bioskop>
          <film zanr="komedija">
            <naziv> Ghostbusters </naziv>
            <reditelj>Ivan Reitman</reditelj>
            <trajanje>1h 45min</trajanje>
            <cena>300</cena>
          </film>
          <film zanr="horor">
            <naziv> It </naziv>
            <reditelj>Andy Muschietti</reditelj>
            <trajanje>2h 15min</trajanje>
            <cena>350</cena>
          </film>
          <film zanr="fantastika">
            <naziv> Interstellar </naziv>
            <reditelj> Christopher Nolan</reditelj>
            <trajanje>2h 49min</trajanje>
            <cena>400</cena>
          </film>
        </bioskop>
```

Konačno, ukoliko se ovaj XML dokument otvori pomoću pretraživača veba koji podržava XSLT, on će biti prikazan kao na slici 6.

Repertoar bioskopa

Naziv	Reditelj	Trajanje	Cena
Ghostbusters	Ivan Reitman	1h 45min	300
It	Andy Muschietti	2h 15min	350
Interstellar	Christopher Nolan	2h 49min	400

Slika 6: Prikaz dokumenta „bioskop.xml“ formiran pomoću XSLT

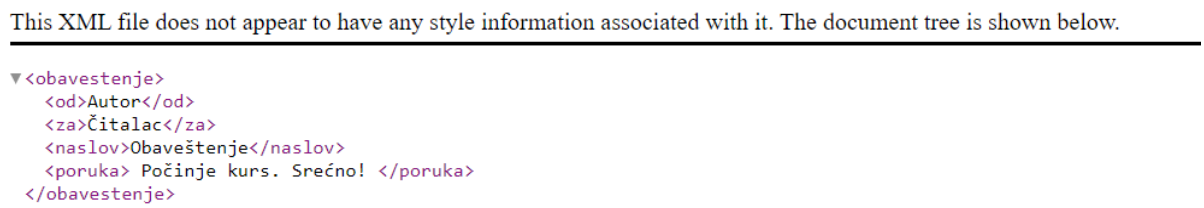
Tema rada nije XSLT, prema tome, nema potrebe dublje zalaziti u detalje ovog jezika. Više o XSLT tehnologiji se može pročitati u [4].

8 Prikaz XML dokumenta

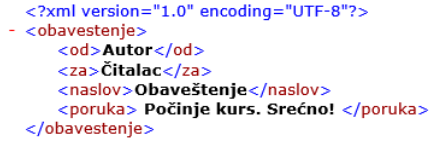
Većina pretraživača veba prikazuje XML dokumente vrlo slično njihovom izvornom kodu. Zapravo, prikazuju ih kao obojeni kod uz koji se mogu naći znaci plus, minus ili strelice za proširivanje i sužavanje vidljivosti sadržaja elemenata. Ovo se dešava zato što XML dokumenti ne sadrže informacije o prikazu. S obzirom na to da su etikete u XML dokumentima formirane od strane autora, pretraživači ne razumeju njihovo tačno značenje. Prema tome, bez informacija o prikazu, oni ih mogu prikazati samo u izvornom obliku.

Zbog sličnosti XML i HTML standarda, dešava se da korisnici očekuju da prikaz XML dokumenata liči na prikaz HTML dokumenata. Međutim, XML se suštinski razlikuje od HTML standarda. Dok je HTML namenjen prikazu podataka, XML ima fokus na samim podacima, odnosno njihovom skladištenju, označavanju i opisivanju, a ne na tome kako će podaci izgledati ukoliko se pokrenu pomoću pretraživača veba ili različitih aplikacija. U prethodnoj lekciji pomenut je jezik XSLT kojim se XML dokument može transformisati u neki drugi format pa time dobiti drugačiju formu prikaza ukoliko je to potrebno.

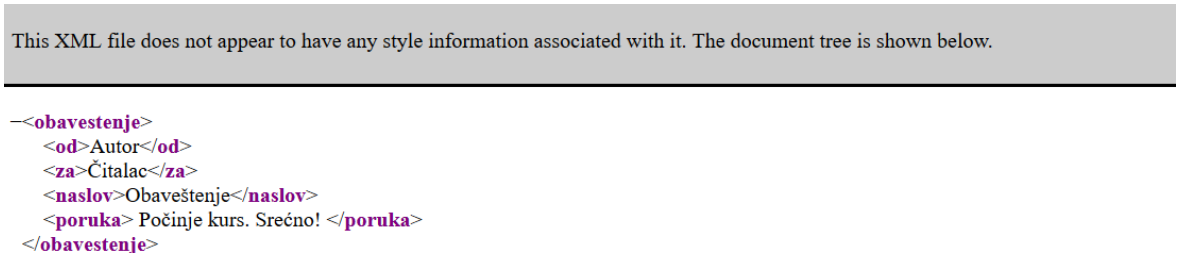
Prikaz XML dokumenta koji daju neki internet pretraživači predstavljeni su slikama 7, 8 i 9.



Slika 7: Prikaz XML dokumenta u pretraživaču Google Chrome



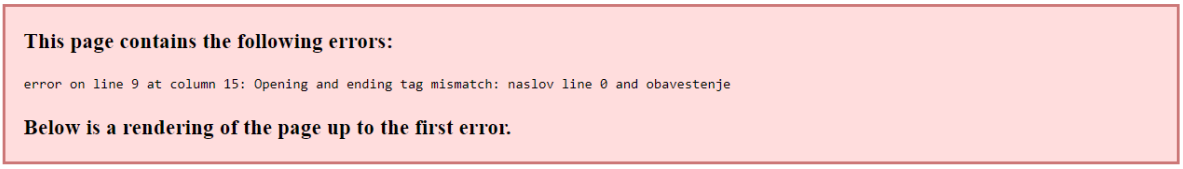
Slika 8: Prikaz XML dokumenta u pretraživaču Internet Explorer



Slika 9: Prikaz XML dokumenta u pretraživaču Mozilla Firefox

8.1 Prikaz XML dokumenta sa greškom

Ukoliko XML dokument ima sintaksnu grešku, različiti pretraživači mogu da prikažu taj dokument na različite načine. Neki će prijaviti grešku, neki će ga prikazati, a neki će ga neispravno prikazati. Slika 10 sadrži prikaz XML dokumenta sa greškom kakav daje pretraživač Google Chrome.



Autor Čitalac Obaveštenje Počinje kurs. Srećno!

Slika 10: XML dokument sa greškom (Google Chrome)

U XML dokumentu čiji je prikaz ilustrovan slikom 10, javlja se greška zato što je završna etiketa elementa „naslov“ pogrešno zapisana. Zapravo, zapisana je isto kao početna, što prikazuje primer 29.

Primer 29: Pogrešno zapisana završna etiketa elementa „naslov“

```
<?xml version="1.0" encoding="UTF-8"?>
    <obavestenje>
      <od>Autor</od>
      <za>Čitalac</za>
      <naslov>Obaveštenje<naslov>
      <poruka>
        Počinje kurs. Srećno!
      </poruka>
    </obavestenje>
```

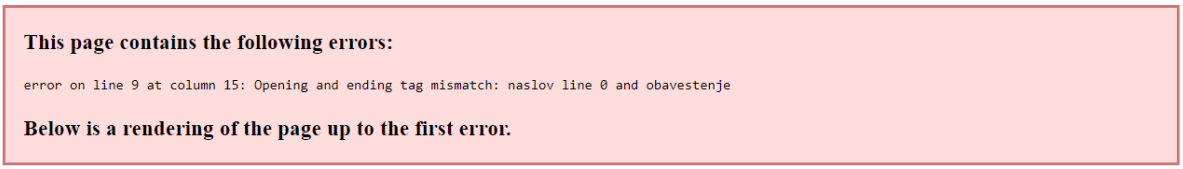
9 Validacija XML dokumenta

Svaki XML dokument mora biti dobro formiran. Za XML dokument kažemo da je dobro formiran ako ima ispravnu sintaksu, odnosno ako su ispoštovana brojna sintaksna pravila, od kojih se posebno mogu istaći sledeća:

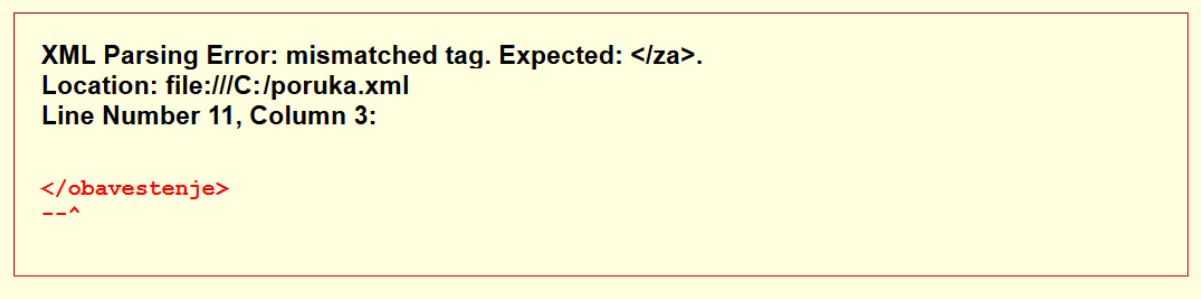
- XML dokument mora imati jedinstven korenski element;
- Svaka početna etiketa mora imati odgovarajuću završnu etiketu;
- XML etikete su osetljive na veličinu slova (eng. case sensitive);
- XML elementi moraju biti dobro ugnježđeni;
- Vrednosti XML atributa moraju biti navedene pod jednostrukim ili dvostrukim navodnicima.
- Jedan XML element ne sme imati više atributa nazvanih istim imenom;
- Komentari i instrukcije za obradu se ne smeju naći unutar etiketa;
- Znakovi „<“ ili „&“ se ne smeju naći unutar tekstualnog sadržaja elemenata ili unutar vrednosti atributa, osim ako su korićene reference entiteta.

Za razliku od HTML standarda, gde pretraživači mogu da otvore HTML dokument sa greškom, u XML datotekama prostora za greške nema. Svaka aplikacija koja koristi XML dokument će biti zaustavljena ukoliko u njemu postoji greška. Zapravo, XML parser prilikom čitanja dokumenta mora da prijavi sintaksnu grešku bez obzira na to koliko se ona često javlja i da li je „mala“ ili „velika“. Ono što parser ne sme da uradi jeste da ne sme da ispravlja greške koje otkrije. Dakle, ne sme da doda završnu etiketu koja izostaje, ne sme da ispravi veličinu slova ukoliko nije ispoštovana osetljivost na mala i velika slova, ne sme da zatvori otvorene navodnike, ne sme da briše komentare sa mesta na kojima oni nisu dozvoljeni i slično. Prema tome, uvek treba proveriti da li je „finalni“ XML dokument dobro formiran. To se može proveriti na više načina.

Najjednostavniji način jeste da se XML dokument otvori pomoću pretraživača veba koji prepoznaje XML dokumente i ima svoj XML parser. Ovakvi pretraživači mogu da prijave sintaksne greške i ukažu na deo XML dokumenta u kome se javila greška. Neki od takvih pretraživača su Mozilla Firefox i Google Chrome.



Slika 11: Detalji greške (Google Chrome)



Slika 12: : Detalji greške (Mozilla Firefox)

Na slici 11 prikazana je prijava greške u pretraživaču Google Chrome, dok je na slici 12 prikazana prijava greške u pretraživaču Mozilla Firefox. Svaki od ovih pretraživača na svoj način prikazuje detalje o grešci koju je njihov parser primetio: šta je tačno greška, na kom mestu u dokumentu je nastala, lokaciju dokumenta, šta je očekivano rešenje i slično. Uz sve ovo, pretraživač može prikazati ceo dokument ili samo deo na kome je primećena greška ili ga ne prikazati uopšte. Zahvaljujući svemu tome, autor dokumenta može brzo da pronađe deo dokumenta u kome je nastao problem i da ga lako ispravi.

Drugi način da se proveri da li je XML dokument dobro formiran jeste koristeći direktno XML parsere. Neki od njih su SAX, expat, TclExpat, XML Microsoft's Validating XML Processor, Python and XML Processing Preliminary XML Parser, XML Testbed, Java built-in parser.

U ručno pisanim XML dokumentima se često nalaze greške, zato je dobra praksa proveriti da li su dobro formirani. Parseri će prijavljivati greške sve dok se i poslenja ne ispravi, ali se zato proveren i ispravljen XML dokument može slobodno dalje koristiti.

Validan XML dokument nije isto što i dobro formiran XML dokument. Validan XML dokument mora biti dobro formiran, ali pored toga mora odgovarati definiciji tipa dokumenta. Postoje dve vrste definicije tipa dokumenata koje se primenjuju u XML standardu:

- DTD - originalno definisanje tipa dokumenata;
- XML Schema - alternativa standardu DTD.

Definicija tipa dokumenata definiše sintaksu elemenata i atributa jednog XML dokumenta.

10 Definicija tipa dokumenata - XML DTD

Definicija tipa dokumenata, XML DTD (eng. Document Type Definition) je tehnologija za definisanje strukture XML dokumenta. On definiše gradivne blokove i redosled elemenata u XML dokumentu. DTD razlikuje sledeće gradivne blokove jednog XML dokumenta:

- Elementi - glavni gradivni blokovi XML dokumenta;
- Etikete;
- Atributi;
- Entiteti;
- PCDATA (Parsed Character DATA) – tekstualni sadržaj između početne i završne etikete koji će biti analiziran parserom i koju je pre svega moguće parsirati;
- CDATA (Character DATA) – tekstualni sadržaj koji parser neće analizirati (najčešće se odnosi na atribute).

Zapravo, XML DTD standardizuje strukturu XML dokumenta listom elemenata i atributa koji su dozvoljeni u njemu. Takođe, specifikuje i redosled i način na koji su elementi ugnježđeni kao i kog su tipa sadržani podaci. Na taj način, XML DTD omogućava standardizovane informacije o informacijama koje nosi određeni XML dokument i time je osiguran dalji prenos i razmena podataka, bez mogućnosti da se pojave podaci koji nisu odgovarajući prema formatu, strukturi ili sadržaju. Koristan je kada su u pitanju XML dokumenti koji nose veliku količinu podataka. U manjim XML dokumentima lako se mogu uočiti i ispraviti neispravni podaci, dok u velikim XML dokumentima provera ispravnosti nije tako jednostavna i potrebno je uvesti logički red i na taj način izvršiti validaciju podataka. Upravo to omogućava XML DTD. Već je bilo reči o sintaksnim pravilima koja moraju biti ispoštovana da bi XML dokument bio dobro formiran, a ukoliko su ispoštovana i pravila koja zadaje DTD, XML dokument je validan.

10.1 Implementacija DTD dokumenta

XML DTD ima potpuno drugačiju sintaksu od XML standarda. On se može nalaziti u samom XML dokumentu ili u spoljašnjem DTD dokumentu i sadrži deklaracije svih gradivnih blokova XML dokumenta na koji se odnosi. Na primeru XML dokumenta „obavestenje.xml“ mogu se predstaviti oba slučaja.

Primer 30: XML dokument "obavestenje.xml" sadrži referencu ka spoljašnjem DTD dokumentu

```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE obavestenje SYSTEM "obavestenje.dtd">
    <obavestenje rbroj = "001">
      <od>Autor</od>
      <za>Čitalac</za>
      <naslov>Obaveštenje</naslov>
      <poruka>
        Počinje kurs. Srećno!
      </poruka>
    </obavestenje>
```

U primeru 30, *DOCTYPE* definicija sadrži referencu ka spoljašnjem DTD dokumentu "obavestenje.dtd", a izgled DTD dokumenta prikazuje primer 31.

Primer 31: DTD dokument "obavestenje.dtd"

```
<!DOCTYPE obavestenje
     [
     <!ELEMENT obavestenje (od, za, naslov, poruka)>
     <!ELEMENT od (#PCDATA)>
     <!ELEMENT za (#PCDATA)>
     <!ELEMENT naslov (#PCDATA)>
     <!ELEMENT poruka (#PCDATA)>
     <!ATTLIST obavestenje rbroj CDATA "001">
     ]>
```

Tabela 1 opisuje značenje deklaracija elemenata iz DTD dokumenta prikazanog u primeru 31.

Tabela 1: Deklaracije elemenata iz primera 31

Deklaracija	Značenje
!DOCTYPE obavestenje	Korenski element odgovarajućeg XML dokumenta je element *obavestenje*.
!ELEMENT obavestenje (od, za, naslov, poruka)	Element *obavestenje* sadrži četiri pod elementa (od, za, naslov i poruka).
!ELEMENT od (#PCDATA)	Element *od* je tipa "#PCDATA".
!ELEMENT za (#PCDATA)	Element *za* je tipa "#PCDATA".
!ELEMENT naslov (#PCDATA)	Element *naslov* je tipa "#PCDATA".
!ELEMENT poruka (#PCDATA)	Element *poruka* je tipa "#PCDATA".
!ATTLIST obavestenje rbroj CDATA "001"	Element *obavestenje* ima atribut *rbroj* koji je tipa "#CDATA" i ima vrednost *001*.

Kao što je već rečeno, DTD može biti implementiran i unutar XML dokumenta. U tom slučaju DTD je deklarisan unutar *DOCTYPE* definicije, što je prikazano u primeru 32.

Primer 32: DTD implementiran unutar XML dokumenta

```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE obavestenje [
     <!ELEMENT obavestenje (od, za, naslov, poruka)>
     <!ELEMENT od (#PCDATA)>
     <!ELEMENT za (#PCDATA)>
     <!ELEMENT naslov (#PCDATA)>
     <!ELEMENT poruka (#PCDATA)>
     <!ATTLIST obavestenje rbroj CDATA "001">
     ]>
    <obavestenje rbroj = "001">
      <od>Autor</od>
      <za>Čitalac</za>
      <naslov>Obaveštenje</naslov>
      <poruka>
        Počinje kurs. Srećno!
      </poruka>
    </obavestenje>
```

Ukoliko se XML dokument iz primera 32 otvori pomoću pretraživača veba, on neće prikazati DTD. Da bi se DTD pročitao, mora se otići u izvorni kod stranice („view source“).

10.2 **Deklarisanje elemenata**

XML elementi se deklarišu pomoću DTD deklaracije koja sadrži obeležje *!ELEMENT*, ime elementa i tip podataka sadržaja tog elementa ili kategoriju kjoj pripada taj element (primer: prazan element). Deklaracija elementa ima sledeći oblik:

<!ELEMENT ime_elementa (tip sadržaja)>

ili

<!ELEMENT ime_elementa kategorija>.

Ukoliko autor želi da definiše tip sadržaja nekog elementa, iskoristiće prvu od dve prikazane deklaracije.

Primer 33: Definisanje tipa sadržaja elementa

```
<!ELEMENT poruka (#PCDATA)>
```

Element „poruka“ iz primera 33 ima tekstualni sadržaj koji se može parsirati.

Elementi koji imaju podelemente se deklarišu navođenjem imena podelemenata (elemenata potomaka) unutar zagrada:

<!ELEMENT osnovni_element (potomak1, potomak2, potomak3, potomak4)>

Da bi deklaracija elementa sa više podelemenata bila potpuna, svaki od podelemenata se nakon ovoga mora deklarisati, redom. Svaki podelement može imati još svojih podelemenata. Takođe, redosled navođenja podelemenata se mora ispoštovati i u XML dokumentu za koji je pisan DTD.

Primer 34: Potpuna deklaracija elementa sa više podelemenata

```
  <!ELEMENT obavestenje (od, za, naslov, poruka)>
    <!ELEMENT od (#PCDATA)>
    <!ELEMENT za (#PCDATA)>
    <!ELEMENT naslov (#PCDATA)>
    <!ELEMENT poruka (#PCDATA)>
    <!ATTLIST obavestenje rbroj CDATA "001">
```

Primer 35: Deklarisanje podelemenata koji se tačno jednom pojavljuju

```
<!ELEMENT osnovni_element (potomak)>
```

U primeru 35 element „potomak“ se unutar elementa „osnovni_element“ pojavljuje tačno jednom. Svako pojavljivanje češće od toga će prouzrokovati grešku prilikom validacije.

Primer 36: Deklarisanje podelemenata koji se makar jednom ponavljaju

```
<!ELEMENT osnovni_element (potomak+)>
```

Znak „+“ specifikuje da se element „potomak“ unutar elementa „osnovni_element“ mora pojaviti jednom ili više puta.

Primer 37: Deklarisanje podelementa koji se ne mora pojaviti ili se pojavljuje više puta

```
<!ELEMENT osnovni_element (potomak*)>
```

Znak „*“ specifikuje da se element „potomak“ unutar elementa „osnovni_element“ može pojaviti više puta ili se ne mora pojaviti.

Primer 38: Deklarisanje podelementa koji se nijednom ili jednom pojavljuje

```
<!ELEMENT osnovni_element (potomak?)>
```

Znak „?“ specifikuje da se element „potomak“ unutar elementa „osnovni_element“ ne mora pojaviti ili se može pojaviti najviše jednom.

Primer 39: Deklarisanje „ili-ili“ podelemenata:

```
<!ELEMENT osnovni_element (potomak1,potomak2,potomak3,potomak4|potomak5)>
```

Deklaracija prikazana u primeru 39 označava da se unutar elementa „osnovni_element“ moraju pojaviti elementi „potomak1“, „potomak2“ i „potomak3“, a kao četvrti potomak „potomak4“ ili „potomak5”.

Primer 40: Deklarisanje elemenata mešovitog sadržaja

```
<!ELEMENT osnovni_element (#PCDATA|potomak1|potomak2|potomak3)*>
```

Deklaracija prikazana u primeru 40 označava da element „osnovni_element“ može sadržati tekstualni sadržaj koji se može parsirati ili bilo koji broj podelemenata „potomak1“, „potomak2“, „potomak3“.

Primer 41: Deklarisanje praznog elementa

```
<!ELEMENT ime_elementa EMPTY>
```

Primer 41 pokazuje da se prazni elementi deklarišu korišćenjem naziva kategorije *EMPTY*.

Primer 42: Deklarisanje elementa čiji je sadržaj proizvoljnog tipa

```
<!ELEMENT ime_elementa ANY>
```

Primer 42 pokazuje da se elementi čiji sadržaj može biti proizvoljnog tipa deklarišu korišćenjem naziva kategorije *ANY*.

10.3 **Deklarisanje atributa**

XML atributi se deklarišu pomoću DTD deklaracije koja sadrži obeležje *!ATTLIST*, ime elementa koji sadrži taj atribut, ime atributa, tip atributa i podrazumevanu vrednost atributa. Deklaracija atributa ima oblik:

<!ATTLIST ime_elementa ime_atributa tip_atributa "podrazumevana_vrednost_atributa">.

Primer 43: Deklaracija atributa

```
<!ATTLIST obavestenje rbroj CDATA "001">
```

Tipovi atributa prikazani su u tabeli 2.

Tabela 2: Tipovi atributa

Tip	Opis
CDATA	Atribut čija je vrednost je karakterni podatak.
(vrednost1|vrednost2|…)	Atribut čija vrednoost može biti neka od navedenih.

Tip	Opis
ID	Atribut čija je vrednost jedinstveni identifikator.
IDREF	Atribut čija je vrednost referenca ka identifikatoru nekog elementa.
IDREFS	Atribut čija je vrednost lista identifikatora drugih elemenata.
NMTOKEN	Atribut čija je vrednost validno XML ime.
NMTOKENS	Atribut čija je vrednost lista validnih imena.
ENTITY	Atribut čija je vrednost entitet.
ENTITIES	Atribut čija je vrednost lista entiteta.
NOTATION	Atribut čija je vrednost ime notacije.
xml:	Atribut čija je vrednost predefinisana XML vrednost.

Različite vrste vrednosti atributa prikazane su u tabeli 3.

Tabela 3: Vrednosti atributa

Vrednost	Opis
vrednost	Podrazumevana vrednost atributa.
#REQUIRED	Atribut se mora navesti.
#IMPLIED	Atribut se ne mora navesti.
#FIXED *vrednost*	Vrednost atributa je fiksna.

U slučaju navođenja podrazumevane vrednosti atributa DTD deklaracijom, ukoliko se prilikom formiranja odgovarajućeg XML dokumenta ne navede taj atribut, on će se ipak pojaviti i imati podrazumevanu vrednost.

Primer 44: Podrazumevana vrednost atributa eksplicitno promenjena prilikom formiranja XML dokumenta

DTD:

```
<!ATTLIST obavestenje rbroj CDATA "000">
```

XML:

```
...
<obavestenje rbroj = "001" />
...
```

U slučaju da unutar etikete „obavestenje“ nije naveden atribut „rbroj“ u XML dokumentu u primeru 44, prilikom pokretanja tog dokumenta aplikacijom koja ga koristi, atribut „rbroj“ bi bio prikazan i imao bi podrazumevanu vrednost 000.

Ukoliko je potrebno da autor XML dokumenta obavezno navede određeni atribut, ali ne postoji podrazumevana vrednost koju bi on imao, koristi se ključna reč *#REQUIRED*.

Primer 45: Deklarisanje atributa pomoću obeležja #REQUIRED

```
<!ATTLIST obavestenje rbroj CDATA #REQUIRED >
```

Ispravan XML na koji se odnosi navedeni DTD:

```
<obavestenje rbroj = "1231" />
```

Neispravan XML na koji se odnosi navedeni DTD:

```
<obavestenje />
```

Ukoliko autor XML dokumenta može a ne mora da navede određeni atribut, i ne postoji podrazumevana vrednost koju bi on imao, koristi se ključna reč *#IMPLIED*.

Primer 46: Deklarisanje atributa pomoću obeležja #IMPLIED

```
<!ATTLIST obavestenje rbroj CDATA #IMPLIED >
```

Ispravan XML na koji se odnosi navedeni DTD:

```
<obavestenje rbroj = "111-2222" />
```

Ispravan je i ovaj XML na koji se odnosi navedeni DTD:

```
<obavestenje />
```

Ukoliko je potrebno da vrednost određenog atributa bude fiksirana, u deklaraciji se nakon ključne reči *#FIXED* navodi fiksirana vrednost. Ako se vrednost takvog atributa promeni u XML dokumentu, parser će prijaviti grešku.

Primer 47: Definisanje atributa korišćenjem obeležja #FIXED

```
<!ATTLIST obavestenje rbroj CDATA #FIXED "1" >
```

Ispravan XML na koji se odnosi navedeni DTD:

```
<obavestenje rbroj = "1" />
```

Neispravan XML na koji se odnosi navedeni DTD:

```
<obavestenje rbroj = "2" />
```

Ako atribut treba da ima jednu od nekoliko dozvoljenih vrednosti, u DTD deklaraciji se kao tip atributa u zagradi navode sve dozvoljene vrednosti odvojene uspravnim crtama, a nakon toga se pod navodnicima zadaje i podrazumevana vrednost tog atributa.

Primer 48: Deklarisanje atributa koji može imati jednu od nekoliko dozvoljenih vrednosti

```
<!ATTLIST osoba pol (muški|ženski) "muški" >
```

Ispravan XML na koji se odnosi navedeni DTD:

```
<osoba pol = "muški" />
```

Ispravan XML na koji se odnosi navedeni DTD:

```
< osoba pol = "ženski" />
```

10.4 **Deklarisanje entiteta**

XML DTD se može koristiti i za deklarisanje referenci entiteta, odnosno deklarisanje specijalnih karaktera i stringova koji se koriste u dokumentu. Reference entiteta se mogu deklarisati interno ili eksterno, u zavisnosti od toga razlikuje se sintaksa deklaracija.

Sintaksa interne deklaracije referenci entiteta:

<!ENTITY ime_entiteta "vrednost_entiteta" >.

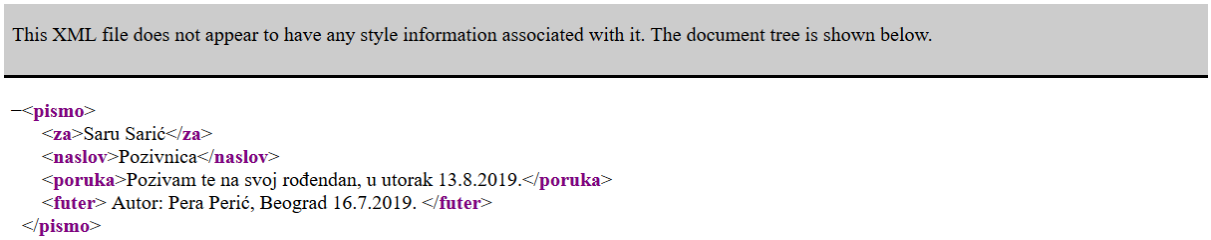
Sintaksa eksterne deklaracije referenci entiteta:

<!ENTITY ime_entiteta SYSTEM "URI/URL" >.

Primer 49: Deklarisanje referenci entiteta

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pismo [
<!ENTITY autor "Autor: Pera Perić">
<!ENTITY mesto "Beograd">
<!ENTITY datum "16.7.2019.">
]>
<pismo>
  <za>Saru Sarić</za>
  <naslov>Pozivnica</naslov>
  <poruka>Pozivam te na svoj rođendan, u utorak 13.8.2019.</poruka>
  <futer> &autor;, &mesto; &datum;</futer>
</pismo>
```

Zanimljivo je to da će XML dokument iz primera 49, ukoliko se pokrene pomoću pretraživača koji podržava XML, biti prikazan bez DTD deklaracija, kao na slici 13.



Slika 13: Prikaz XML dokumenta iz primera 49

XML DTD je dobro koristiti kada se pišu veliki XML dokumenti ili kada grupa ljudi radi na zajedničkom projektu. Zahvaljujući njemu, mogu se standardizovati struktura XML dokumenta i tipovi podataka koji će se koristiti, a samim tim i proveravati validnost podataka koji se razmenjuju tokom rada. XML DTD nije obavezan u XML dokumentu, prema tome, u slučaju pisanja manjih XML dokumenata nema potrebe kreirati i odgovarajući DTD zato što se oni jednostavno mogu proveriti.

11 XML šema

XML Schema (XML šema) je alternativni način za specifikovanje strukture XML dokumenta. Ona se često naziva i XML Schema Definition (XSD), prema tome, XML Schema dokumenti imaju ekstenziju „ *.xsd* “. Kao i XML DTD, ona definiše gradivne blokove, redosled pojavljivanja elemenata i atributa, tip sadržanih podataka i podrazumevane i fiksirane vrednosti elemenata i atributa. Takođe, XML dokument je validan ukoliko je dobro formiran i poštuje sintaksu koju zadaje odgovarajuća XML Schema. U prednosti je u odnosu na XML DTD iz nekoliko razloga. Kao prvo, XML Schema ima XML sintaksu, stoga se ne mora učiti novi jezik za pisanje XML Schema dokumenata. Oni se mogu pisati i uređivati korišćenjem istog editora kao i prilikom formiranja običnog XML dokumenta i parsirati XML parserom. Kako ima XML sintaksu, to je XML Schema proširiva. Sa druge strane, XML Schema podržava različite tipove podataka pa je njome lakše opisati strukturu sadržaja odgovarajućeg XML dokumenta, izvršiti validaciju podataka, konvertovati podatke iz jednog tipa u drugi i slično. U XML Schemi ima dosta definisanih tipova podataka. Najčešće su to:

- xs:string,
- xs:decimal,
- xs:integer,
- xs:byte,
- xs:boolean,
- xs:date,
- xs:dateTime.

XML Schemom se definišu elementi i atributi koji će se naći u XML dokumentu na koji se ona odnosi. XML Schema dokumenti počinju XML deklaracijom. Korenski element svake XML Scheme je xs:schema i on sadrži deklaracije svih elemenata i atributa koji se pojavljuju u XML dokumentu na koji se šema odnosi. Ovaj element može sadržati atribute, a obavezni atribut jeste xmlns:xs kojim se definiše prostor imena kome pripadaju svi elementi šeme, pri čemu se svi elementi šeme zapisuju sa prefiksom xs:. Taj prostor imena je uvek http://www.w3.org/2001/XMLSchema. Elementi mogu biti prostog tipa (simpleType) ili složenog tipa (complexType), dok atributi ne mogu biti složenog tipa. Više o korišćenju XML Scheme može se pročitati u [5].

11.1 Deklarisanje elemenata

11.1.1 XML elementi prostog tipa

XML elementi prostog tipa imaju samo tekstualni sadržaj i ne mogu sadržati druge elemente i atribute. Međutim, tekstualni sadržaj tih elemenata može biti različitog tipa: ili tipa koji je već definisan u XML Schemi (string, decimal, integer, boolean, date...) ili tipa koji sam autor XML dokumenta može definisati.

Deklaracija XML elementa prostog tipa ostvaruje se pomoću elementa *xs:element* koji ima dva atributa *name* i *type*:

<xs:element name="ime_elementa" type="tip_podataka"/>.

Primer 50: XML elementi i odgovarajuće XML Schema deklaracije

XML elementima:

```
<ime>Pera</ime>
<prezime>Perić</prezime>
<datum_rođenja>1994-05-11</datum_rođenja>
```

```
<redni_broj>5</redni_broj>
```

Odgovaraju deklaracije:

```
<xs:element name="ime" type="xs:string"/>
<xs:element name="prezime" type="xs:string"/>
<xs:element name="datum_rođenja" type="xs:date"/>
<xs:element name="redni_broj" type="xs:integer"/>
```

Elementi mogu imati podrazumevanu (default) ili fiksiranu (fixed) vrednost.
Podrazumevana vrednost je ona koja se dodeljuje elementu ukoliko nijedna druga nije eksplicitno zadata prilikom formiranja XML dokumenta.

<xs:element name="ime" type="tip_podatka" default="podrazumevana_vrednost"/>.

Fiksirana vrednost elementa se u XML dokumentu ne može menjati:

<xs:element name="ime" type=" tip_podatka " fixed="fiksirana_vrednost"/>.

11.1.2 XML elementi složenog tipa

XML elementi složenog tipa mogu sadržati druge elemente i atribute i oni se XML Schemom deklarišu pomoću elementa *xs:complexType*.

Primer 51: Deklaracija elementa složenog tipa

```
<xs:element name="ime_složenog_elementa">
      <xs:complexType>
       <xs:sequence>
         <xs:element name="prvi_podelement" type="tip_elementa"/>
         <xs:element name="drugi_podelement " type="tip_elementa"/>
         <xs:element name="treci_podelement " type=" tip_elementa "/>
         ...
       </xs:sequence>
      </xs:complexType>
</xs:element>
```

Elementima *xs:complexType* i *xs:sequence* definiše se da je dati element složenog tipa i da sledi niz elemenata koje on sadrži. U odgovarajućem XML dokumentu se podelementi deklarisanog složenog elementa moraju pojavljivati u redosledu koji je zadat šemom.

11.2 **Deklarisanje atributa**

Vrednosti atributa svakako ne mogu sadržati elemente ili druge atribute, prema tome atributi se uvek deklarišu kao prosti tipovi. Deklaracija atributa ostvaruje se pomoću elementa *xs:attribute*:

<xs:attribute name="ime_atributa" type="tip_atributa">.

Tip vrednosti atributa može biti neki od već definisanih u XML Schemi (string, decimal, integer, boolean, date...) ili može biti definisan od strane autora XML dokumenta.

Primer 52: XML Schema deklaracija atributa

XML atribut „pol“ elementa „osoba“:

```
<osoba pol="muški">Pera</osoba>
```

U XML Schemi ima deklaraciju:

```
<xs:attribute name="pol" type="xs:string"/>
```

Vrednosti atributa, kao i vrednosti elemenata, mogu biti podrazumevane ili fiksirane, što se definiše analogno načinu definisanja vrednosti elemenata. Takođe, podrazumevana vrednost se dodeljuje atributu ukoliko nijedna druga nije posebno zadata prilikom formiranja XML dokumenta, a fiksirana se, u XML dokumentu, ne može menjati.
Definicijom tipa podataka koje sadrže elementi ili atributi, ograničava se tip njihovog sadržaja. Recimo, ukoliko je XML Schemom definisan tip nekog elementa *xs:integer* a u odgovarajućem XML dokumentu je njegov sadržaj tekstualni „Dobro došli“, taj element neće biti validan i XML parser će tražiti korekciju ovog elementa.

11.3 **Implementacija XML Scheme**

Kao što XML dokument može imati referencu ka spoljašnjem DTD dokumentu, kojim se specifikuje njegova struktura, tako može imati i referencu ka spoljašnjem XML Schema dokumentu. To se može sagledati na već pomenutom primeru XML dokumenta „obavestenje.xml“.

Primer 53: XML dokument „obavestenje.xml“

```
<?xml version="1.0" encoding="UTF-8"?>
<obavestenje>
      <od>Autor</od>
      <za>Čitalac</za>
      <naslov>Obaveštenje</naslov>
      <poruka>
        Počinje kurs. Srećno!
      </poruka>
</obavestenje>
```

XML Schema „obavestenje.xsd“ prikazana je u primeru 54.

Primer 54: XML Schema „obavestenje.xsd“

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>
    <xs:element name="obavestenje">
      <xs:complexType>
       <xs:sequence>
         <xs:element name="od" type="xs:string"/>
         <xs:element name="za" type="xs:string"/>
         <xs:element name="naslov" type="xs:string"/>
         <xs:element name="poruka" type="xs:string"/>
       </xs:sequence>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

Referenca ka XML Schemi iz primera 54 se u XML dokument unosi kao atribut korenskog elementa na sledeći način (u primeru 55, korenski element je „obavestenje“):

Primer 55: Referenca ka XML Schema dokumentu ukoliko deklarisani gradivni blokovi ne pripadaju nijednom prostoru imena

```
<?xml version="1.0"?>
      <obavestenje xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:noNamespaceSchemaLocation="obavestenje.xsd">
             <od>Autor</od>
             <za>Čitalac</za>
             <naslov>Obaveštenje</naslov>
             <poruka>
               Počinje kurs. Srećno!
             </poruka>
      </obavestenje>
```

Dakle, u korenskom elementu se definiše prostor imena za prefiks xsi (XML Schema Instance), a zatim se zadaje adresa odgovarajuće šeme pomoću atributa *xsi:noNamespaceSchemaLocation* čija je vrednost lokacija odgovarajućeg XML Schema dokumenta.

Međutim, ukoliko gradivni blokovi deklarisani XML Schemom pripadaju nekom XML prostoru imena, unošenje reference u određeni XML dokument ka takvoj šemi postaje složenije. Zapravo, tada se, u XML Schemi, korenskom elementu *xs:schema* pored atributa *xmlns:xs* dodaje i atribut *targetNamespace* koji označava prostor imena kome pripadaju svi elementi i atributi deklarisani ovom šemom. Referenca u XML dokumentu ka ovakvoj šemi se formira pomoću već pomenutog atributa *xmlns:xsi* i atributa *xsi:schemaLocation* koji ima dve vrednosti od kojih je prva prostor imena dokumenta šeme a druga lokacija šeme koja se koristi u tom prostoru imena. Da bi svi gradivni blokovi posmatranog XML dokumenta zaista pripadali određenom prostoru imena, potrebno je deklarisati ili podrazumevani prostor imena (sintaksa *xmlns="URIprostoraImenaŠeme"*) ili prefiks koji će ukazivati na taj prostor imena (sintaksa *xmlns:prefiks="URIprostoraImenaŠeme"*).

Primer 56: XML Schema XML dokumenta čiji elementi pripadaju nekom prostoru imena

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                   targetNamespace="URIprostoraImenaŠeme">
    <xs:element name="obavestenje">
      <xs:complexType>
       <xs:sequence>
         <xs:element name="od" type="xs:string"/>
         <xs:element name="za" type="xs:string"/>
         <xs:element name="naslov" type="xs:string"/>
         <xs:element name="poruka" type="xs:string"/>
       </xs:sequence>
     </xs:complexType>
   </xs:element>
</xs:schema>
```

XML dokument sa referencom ka odgovarajućem XML Schema dokumentu prikazan je u primeru 57.

Primer 57: Referenca ka XML Schema dokumentu ukoliko deklarisani gradivni blokovi pripadaju nekom prostoru imena

```
<?xml version="1.0"?>
      <obavestenje xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation=" URIprostoraImenaŠeme  obavestenje.xsd"
             xmlns="URIprostoraImenaŠeme">
             <od>Autor</od>
```

```
        <za>Čitalac</za>
        <naslov>Obaveštenje</naslov>
        <poruka>
          Počinje kurs. Srećno!
        </poruka>
    </obavestenje>
```

Da bi se proverila validnost XML dokumenta u odnosu na zadatu šemu, potreban je XML parser koji podržava šeme. Jedan od takvih je Xerces parser koji je kreirala organizacija Apache XML Project. Više o tome može se pročitati u [3].

12 Upotreba i struktura XPath izraza

Kako XML dokumenti služe sa čuvanje i razmenu podataka, često se javlja potreba da se pristupi pojedinačnim podacima ili određenim grupama podataka unutar XML dokumenta. Da bi se to postiglo, razvijene su posebne tehnologije za identifikovanje različitih delova XML dokumenta i izdvajanje željenih elemenata i atributa. Jedna od njih je tehnologija XPath (preporučena od strane W3C) koja je namenjena pristupu različitim delovima XML dokumenta.

12.1 Čvorovi XML dokumenta

XPath omogućava pristup raznim delovima XML dokumenta tako što prepoznaje strukturu stabla XML dokumenta i njegove grane i čvorove. Xpath razlikuje sedam vrsti čvorova, i to su:

- Korenski čvor;
- Čvorovi elemenata;
- Čvorovi atributa;
- Čvorovi teksta;
- Čvorovi komentara;
- Čvorovi instrukcija za obradu;
- Čvorovi prostora imena.

Osim toga, XPath prepoznaje i veze među njima, odnosno međusobnu povezanost elemenata, ili elemenata i atributa i slično. Sadrži biblioteku od 27 standardizovanih funkcija, među kojima su funkcije za rad sa čvorovima, znakovnim nizovima, logičkim vrednostima, numeričke funkcije, funkcije za upoređivanje podataka tipa *date* i *time* i još mnogo njih. Ta biblioteka se može proširiti i funkcijama koje korisnik sam kreira. XPath je dizajniran da radi sa logičkim strukturama jednog XML dokumenta.

Važno je napomenuti da XPath ne može locirati XML deklaraciju, definiciju tipa dokumenata (DOCTYPE), niti delove DTD dokumenata.

Treba naglasiti da korenski čvor nije isto što i korenski element. On sadrži korenski element XML dokumenta i sve komentare i instrukcije za obradu koje se mogu naći ispred njegove početne ili iza njegove završne etikete. Ostale vrste čvorova se jednostavno mogu uočiti u XML dokumentu.

Primer 58: XML dokument "bioskop.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
     <bioskop>
       <film zanr="horor">
         <naziv> It </naziv>
         <reditelj>Andy Muschietti</reditelj>
         <trajanje>2h 15min</trajanje>
         <cena>350</cena>
       </film>
       <film zanr="komedija">
         <naziv> Ghostbusters </naziv>
         <reditelj>Ivan Reitman</reditelj>
         <trajanje>1h 45min</trajanje>
         <cena>300</cena>
       </film>
       <film zanr="fantastika">
```

```
        <naziv> Interstellar </naziv>
        <reditelj> Christopher Nolan</reditelj>
        <trajanje>2h 49min</trajanje>
        <cena>400</cena>
      </film>
    </bioskop>
```

U primeru 58, u kome je prikazan jednostavan XML dokument „bioskop.xml“ mogu se primetiti različite vrste čvorova. Čvor „bioskop“ predstavlja korenski čvor. U slučaju da je ovaj dokument sadržao komentare i instrukcije za obradu pre i posle korenskog elementa, i oni bi bili uključeni u korenski čvor. Mogu se primetiti i čvorovi elemenata kao što su „film“, „reditelj“, „trajanje“, čvorovi teksta kao što su „Andy Muschietti“, „2h 49min“ i čvorovi atributa od kojih je jedan „zanr="fantastika"“.

Specifično je to što XPath ne posmatra atribute *xmlns* i *xmlns:prefiks* kao atributske čvorove. Pored toga, kao čvor prostora imena XPath prepoznaje sve elemente i atribute koji pripadaju tom prostoru imena, a ne samo element u kome je prostor imena deklarisan. Čvorovi prostora imena se ne obrađuju eksplicitno tako često.

Čvorovi koji nemaju potomke ni pretke nazivaju se atomskim vrednostima. Neke atomske vrednosti iz primera dokumenta „bioskop.xml“ su: „Interstellar“, „2h 49min“, „"fantastika"“.

Veze između čvorova analogne su vezama između elemenata XML dokumenta. Zapravo, postoje čvorovi roditelji i čvorovi deca i svaki element i atribut ima jednog svog roditelja. Čvorovi elemenata mogu i ne moraju imati jedno ili više dece. Svi čvorovi koji imaju zajedničkog roditelja nazivaju se braćom i sestrama čvorovima. Čvorovi preci jednog čvora su njegov roditelj, roditelj roditelja i tako dalje, a čvorovi potomci jednog čvora su njegovo dete, dete deteta i tako dalje.

12.2 **XPath operatori**

Često XPath izrazi u sebi sadrže operatore. XPath podržava operatore osnovnih računskih operacija kao što su operator sabiranja (+), oduzimanja (-), množenja (*), celobrojnog deljenja (div) i određivanja ostatka pri deljenju (mod). Osim navedenih, XPath podržava i operatore poređenja: jednako (=), različito (!=), manje (<), manje ili jednako (<=), veće (>), veće ili jednako (>=), kao i logičke operatore „*or*“ i „*and*“. Specifičan operator koji se često koristi u XPath izrazima je operator „|“ kojim se istovremeno ispituju dva skupa čvorova različite vrste.

12.3 **XPath izrazi**

Glavni alat koji XPath pruža jesu XPath izrazi. Zapravo, XPath omogućava pisanje izraza koji upućuju na željene čvorove određenog XML dokumenta. Recimo, na atribut „zanr“ elementa „film“, ili na drugi element „film“ koji se javlja u datom XML dokumentu, ili pak na željenog potomka određenog elementa, skup određenih elemenata i slično. Mogu se koristiti u jezicima Java, C, C++, PHP, JavaScript, Python, XSLT i mnogim drugim. XPath izrazi kao rezultat vraćaju skup čvorova, znakovne nizove, brojeve ili logičke vrednosti. Prema tome, postoje različite vrste XPath izraza:

- Lokacione putanje (ili putanje lokacija);
- Logički izrazi;
- Znakovni nizovi;
- Brojevini izrazi.

12.3.1 Lokacione putanje

Lokacione putanje su XPath izrazi koji se najviše koriste i oni služe za pronalaženje nekog skupa čvorova u dokumentu. Taj skup (lista) čvorova može biti prazan ili sadržati jedan ili više čvorova. Elementi tog skupa mogu biti čvorovi iste vrste ili kombinacija različitih vrsta čvorova. Lokacione putanje mogu biti *apsolutne* ili *relativne*. U oba slučaja sadrže jedan ili više koraka (XPath izraza) razdvojenih znakom „/“. Ukoliko putanja počinje znakom „/“, ona je *apsolutna*, a ukoliko taj znak izostaje sa početka putanje, u pitanju je *relativna* lokaciona putanja. Tekući čvor (ili skup čvorova) u odnosu na koji se pronalaze ostali čvorovi naziva se konteksni čvor (ili konteksni skup čvorova).

Primer 59: Apsolutna lokaciona putanja

```
/korak1/korak2/korak3/...
```

Primer 60: Relativna lokaciona putanja

```
korak1/korak2/korak3/...
```

Korak lokacione putanje se sastoji od:

- Ose;
- Testa čvorova (node-test);
- Predikta (može a ne mora postojati u koraku).

Konteksni čvor je uvek u nekom odnosu sa ostalim čvorovima dokumenta. XPath ose razvrstavaju čvorove dokumenta u različite skupove u zavisnosti od toga u kakvom su odnosu sa konteksnim čvorom. Odnosno, preciznije određuju smer pretrage čvorova u odnosu na konteksni čvor. U tabeli 4 navedene su XPath ose.

Tabela 4: Ose

XPath osa	Značenje
parent	Roditelj konteksnog čvora.
ancestor	Preci (roditelj, roditelj roditelja, itd.) konteksnog čvora.
ancestor-or-self	Svi preci (roditelj, roditelj roditelja, itd.) konteksnog čvora i on sam.
attribute	Atributi konteksnog čvora.
child	Deca konteksnog čvora.
descendant	Potomci (deca, deca dece, itd.) konteksnog čvora.
descendant-or-self	Svi potomci (decu, decu dece, itd.) konteksnog čvora i on sam.
following	Svi čvorovi koje dokument sadrži nakon završne etikete konteksnog čvora.
following-sibling	Svi čvorovi elemenata koji su međusobno braća i sestre, nakon konteksnog čvora.
namespace	Svi čvorovi koji pripadaju istom prostoru imena kao i konteksni čvor.
preceding	Svi čvorovi koji se u dokumentu nalaze ispred konteksnog čvora, osim čvorova predaka konteksnog čvora, čvorova atributa i čvorova prostora imena.

XPath osa	Značenje
preceding-sibling	Svi čvorovi elemenata koji su međusobno braća i sestre i nalaze se ispred konteksnog čvora.
self	Konteksni čvor.

Testovi čvorova su izrazi kojima se dalje sužava skup čvorova koji su odabrani zadatom osom. Postoji sedam vrsti testova čvorova i to su:

- ime_čvora - Na osi elemenata ovaj test „prolaze“ svi elementi koji su tako imenovani, na osi atributa – svi atributi koji su tako imenovani, na osi prostora imena – svi prostori imena sa tim prefiksom;
- prefiks:* - Ovaj test „prolaze“ svi čvorovi elemenata i čvorovi atributa čiji su URI identifikatori prostora imena jednaki URI identifikatoru prostora imena koji je definisan za ovaj prefiks;
- comment() – Odgovara čvorovima komentara;
- text() – Odgovara čvorovima teksta;
- processing-instruction() – Odgovara čvorovima instrukcija za obradu;
- node() – Odgovara svim vrstama čvorova.

Pomoću predikata dalje se sužava skup čvorova koji su izabrani testom čvora. Njima se pronalaze specifični čvorovi sa specifičnim vrednostima. Navode se unutar uglastih zagrada odmah nakon testa čvorova. To su najčešće logički izrazi kojima se testiraju svi čvorovi iz tekuće liste čvorova u odgovarajućem koraku putanje. Ukoliko je rezultat testa logička vrednost *true,* testiran čvor se zadržava u listi, a ukoliko je rezultat *false*, testiran čvor se briše iz liste.

Potpuna sintaksa koraka lokacionih putanja je sledeća:

```
ime_ose::test_čvora[predikat].
```

Tabela 5: Lokacione putanje

XPath izraz	Značenje
child::ime_čvora	Pronalazi sve elemente zadatog imena koji su deca konteksnog čvora.
parent:: node()	Pronalazi roditelja konteksnog čvora.
attribute::ime_atributa	Pronalazi sve atribute konteksnog čvora koji imaju zadato ime.
child::*	Pronalazi sve elemente koji su deca konteksnog čvora.
attribute::*	Pronalazi sve atribute konteksnog čvora.
child::text()	Pronalazi sve čvorove teksta sve dece konteksnog čvora.
child::node()	Pronalazi sve čvorove koji su deca konteksnog čvora.
descendant::ime_čvora	Pronalazi sve potomke zadatog imena, konteksnog čvora.
descendant-or-self::ime_čvora	Pronalazi sve potomke zadatog imena, konteksnog čvora i njega samog ukoliko i njegovo ime odgovara zadatom.
ancestor::ime_čvora	Pronalazi sve pretke zadatog imena, konteksnog čvora.
ancestor-or-self:: ime_čvora	Pronalazi sve pretke zadatog imena, konteksnog čvora i njega samog ukoliko i njegovo ime odgovara zadatom.

Međutim, veća je upotreba skraćenih zapisa koraka lokacija. Prema tome, koraci lokacionih putanja koji se najčešće koriste su prikazani u tabeli 6.

Tabela 6: Kraći zapisi koraka lokacija

XPath izraz	Značenje
ime_čvora	Pronalazi sve čvorove zadatog imena. Skraćeno od *child::ime_čvora*.
/	Pronalazi korenski čvor dokumenta.
//	Pronalazi potomke konteksnog čvora, uključujući i njega, bez obzira na to gde se oni u XML dokumentu nalaze. Skraćeno od: */descendant-or-self:: node()/*.
.	Ukazuje na konteksni čvor. Najčešće se koristi ako je potrebno izdvojiti vrednost tekućeg elementa. Skraćeno od: *self::node()*.
..	Ukazuje na roditelja konteksnog čvora. Skraćeno od: *parent:: node()*.
@	Pronalazi atribut. Skraćeno od: *attribute::ime_atributa.*

Primer 61: Lokaciona putanja 1

```
/bioskop/film/naziv
```

Lokaciona putanja iz primera 61 pronalazi sve „naziv“ elemente korenskog elementa „bioskop“ u dokumentu „bioskop.xml“.
Rezultat ove putanje je:

```
<naziv> Ghostbusters </naziv>
<naziv> It </naziv>
<naziv> Interstellar </naziv>
```

Primer 62: Lokaciona putanja 2

```
/bioskop/film/naziv/text()
```

Lokaciona putanja iz primera 62 pronalazi samo tekstualni sadržaj elemenata iz primera 58.
Rezultat ove putanje je:

```
It
Ghostbusters
Interstellar
```

Tabela 7: Korišćenje predikata

XPath izraz	Značenje
/bioskop/film[1]	Pronalazi se prvi *film* element koji je dete elementa *bioskop*.
/bioskop/film[last()]	Pronalazi se poslednji *film* element koji je dete elementa *bioskop*.
/bioskop/film[last()-1]	Pronalazi se pretposlednji *film* element koji je dete elementa *bioskop*.
/bioskop/film[position()<3]	Pronalaze se prva dva *film* elementa koja su deca elementa *bioskop*.
//film[@zanr]	Pronalaze se svi *film* elementi koji imaju atribut *zanr*.
//film[@zanr='fantastika']	Pronalaze se svi *film* elementi čija je vrednost atributa *zanr* "fantastika".
//cena[.=400]	Pronalaze se svi *cena* elementi čija je vrednost 400.
/bioskop/film[cena>300]	Pronalaze se svi *film* elementi elementa bioskop koji imaju vrednost elementa *cena* veću od 300.
/bioskop/film[cena>300]/naziv	Pronalaze se svi *naziv* elementi elemenata *film* (deca elementa bioskop) koji imaju vrednost elementa cena veću od 300.

12.3.2 Logički izrazi

Logički izrazi se koriste prilikom poređenja podataka, najčešće brojeva. Kao rezultat logičkih izraza, dobijaju se logičke vrednost *true* (istinito) i *false* (neistinito). Za utvrđivanje jednakosti koriste se operatori = i !=, za poređenje vrednosti <, <=, >, >=, a koriste se i operatori *and* i *or* koji omogućavaju konjunkciju i disjunkciju logičkih izraza. Najčešće se javljaju u prediktima lokacionih putanja. U logičkim izrazima može se javiti i funkcija *not()* koja vraća suprotnu logičku vrednost od logičke vrednosti svog argumenta.
Primer 63 prikazuje XPath izraz koji pronalazi sve „film“ elemente čija je cena 350 ili 400.

Primer 63: Logički izraz sa operatorom „or“

```
//film[cena=350 or cena=400]
```

12.3.3 Znakovni nizovi

Znakovni nizovi su uređene niske Unicode znakova. Na primer, to su: „Andy Muschietti“, „λόγος“, „Perišić“ i slično. Vrednosti znakovnih nizova zadaju se pod dvostrukim ili jednostrukim navodnicima, u zavisnosti od samog njihovog sadržaja. Ukoliko znakovni nizovi u nekom delu svog sadržaja imaju dvostruke navodnike, navešće se u jednostrukim navodnicima i obrnuto. Vrednosti znakovnih nizova mogu sadržati beline kao što su razmaci, tabulatori, znakovi za prelazak u novi red i slično.

12.3.4 Brojevni izrazi

Brojevni izrazi služe za predstavljanje brojeva ili za računanje sa brojevima pronađenim u datom dokumentu. U XPath sintaksi, brojevi su dvostruke preciznosti, po standardu IEEE 754 (odgovara tipu *double* u Javi) koji zauzimaju osam bajtova memorije. Ovaj format brojeva sadrži i oznake za pozitivnu i negativnu beskonačnost kao i oznaku za vrednost koja nije broj (NaN). Brojevni izrazi koriste XPath aritmetičke operatore +,-,*,*div* i *mod*. S obzirom na to da se znak „/“ koristi za razdvajanje koraka lokacije, operator *div* se koristi kao operator deljenja.

12.3.5 „Džokerski“ znakovi

Postoje tri džokerska znaka, i to su: *, @* i *node()*. Znak „*“ odgovara svim čvorovima elemenata bez obzira na njihova imena i tipove. Ovaj znak ne odgovara atributima, instrukcijama za obradu, tekstu i komentarima. Može se koristiti sa prefiksom za neki prostor imena i tada odgovara svim elementima tog prostora imena. Znak „@*“ odgovara svim čvorovima atributa. Znak „*node()“* odgovara svim vrstama čvorova.

Tabela 8: Primeri korišćenja džokerskih znakova

XPath izraz	Značenje
/bioskop/*	Pronalazi svu decu čvorove čvora elementa *bioskop*.
//*	Pronalazi sve elemente u dokumentu.
//film[@*]	Pronalazi sve *film* elemente koji imaju barem jedan, bilo kakav atribut.

12.3.6 Pronalaženje više skupova čvorova različite vrste istovremeno

Korišćenjem operatora „|” unutar jednog XPath izraza, može se upotrebiti više lokacionih putanja kako bi se pronašlo više čvorova različite vrste istovremeno.

Primer 64: Korišćeje operatora „|“

```
/bioskop/film[@zanr="fantastika"]/naziv | //cena
```

XPath izraz prikazan u primeru 64 pronalazi sve „naziv“ elemente elementa „film“ čiji atribut „zanr“ ima vrednost „fantastika“ i koji je podelement elementa „bioskop“, i sve cene u dokumentu.

Primer 65: Korišćeje operatora „|“

```
//* | //@*
```

XPath izraz prikazan u primeru 65 pronalazi sve elemente i sve atribute, bez obzira na to gde se oni nalaze u XML dokumentu.

Više o XPath izrazima može se pročitati u [3].

12.4 **XPath funkcije**

XPath ima svoju biblioteku od 27 funkcija. Ta biblioteka se može proširiti ili od strane korisnika ili od strane tehnologija koje koriste XPath, funkcijama koje su dodatno potrebne. Argumenti i vrednosti XPath funkcija mogu biti brojevi, znakovni nizovi, logičke vrednosti ili skup čvorova. Međutim, bez obzira na tip koji neka funkcija očekuje, moguće je zadati joj argument bilo kog tipa, jer će XPath taj tip prevesti u onaj koji funkcija zahteva. Recimo, ukoliko se funkciji koja očekuje znakovnu vrednost kao argument, zada logička vrednost, XPath procesor će tu logičku vrednost pretvoriti u znakovni niz „true“ ili „false“. Jedino što XPath ne može, jeste da promeni broj, znakovni niz ili logičku vrednost u skup čvorova. XPath funkcije mogu primati jedan ili više argumenata ili ne moraju primati argumente.

Primena XPath funkcija u XPath izrazima može se ilustrovati funkcijom *starts-with()* koja prima dva znakovna niza kao argumente, a kao rezultat vraća logičku vrednost. Vrednost *true* se vraća ukoliko prvi znakovni niz počinje drugim znakovnim nizom, a u svim ostalim slučajevima vraća vrednost *false*. Važno je naglasiti da je ova funkcija osetljiva na veličinu slova. Prema tome, ukoliko se pozove sa *starts-with("Simbolična priča", "Simbol")* vratiće logičku vrednost *true*, dok pozvana sa *starts-with("Simbolična priča", "simbol")* vraća vrednost false.

Primer 66 prikazuje XPath izraz koji vraća sve nazive filmova iz dokumenta „bioskop.xml“ koji počinju slovom „I“.

Primer 66: Primena funkcije „starts-with()“

```
//bioskop/film/naziv[starts-with(., ' I')]
```

Rezultat ovog XPath izraza je*:*

```
<naziv> It </naziv>
<naziv> Interstellar </naziv>
```

U primeru 66, bilo je potrebno uzeti u obzir da tekstualni sadržaj naziva filmova u posmatranom XML dokumentu počinje razmakom (belinom). Stoga je i drugi argument funkcije *starts-with()* počeo razmakom. U slučaju da razmak nije naveden, funkcija *starts-with()* bi vratila vrednost *false*, i dati XPath izraz ne bi pronašao odgovarajuće elemente.

12.4.1 Funkcije skupa čvorova

Funkcije skupa čvorova za argumente primaju skupove čvorova ili vraćaju informacije o čvorovima. Ovoj vrsti funkcija pripadaju funkcije *name(), id(), count(), position(), last(), namespace-uri(), local-name()*.

Funkcija koja kao rezultat vraća ime čvora je funkcija *name()*. Ona za argumente prima skup čvorova, a kao rezultat vraća znakovni niz koji sadrži ime prvog čvora u tom skupu. Ukoliko joj se ne prosledi argument, ona vraća ime konteksnog čvora ili prazan znakovni niz ako konteksni čvor nema ime (recimo, to se dešava u slučaju čvorova komentara).

Argument funkcije *id()* je znakovni niz koji sadrži jedan ili vše identifikatora razdvojenih belinama. Međutim, identifikatori nisu atributi sa imenima ID ili id, već atributi koji su tipa ID deklarisani u DTD bloku ili dokumentu. Dakle, ova funkcija se može koristiti za obradu podataka samo onih XML dokumenata koji imaju DTD. Rezltat funkcije *id()* je skup svih čvorova čiji identifikatori odgovaraju ulaznim.

Funkcije *count()* i *last()* za argumente primaju skup čvorova. Rezultat ovih funkcija je ili broj elemenata tog skupa ili redni broj određenog čvora u njemu. Funkcija *count()* vraća broj svih elemenata u skupu čvorova koji je primila kao argument; funkcija *last()* vraća redni broj poslednjeg čvora u tom skupu. Slična njima je i funkcija *position()* čiji je rezultat broj koji odgovara položaju tekućeg čvora u konteksnom skupu čvorova.

Funkcija *local-name()* kao rezultat vraća lokalno ime konteksnog čvora (ukoliko joj se ne proslede argumenti) ili ime prvog čvora u skupu čvorova koji joj je prosleđen kao argument. U slučaju da ime sadrži prefiks, rezultat je samo deo iza dvotačke. Imena bez prefiksa se u celosti vraćaju kao rezultat ove funkcije.

Funkcija *namespace-uri()* kao rezultat vraća URI identifikator prostora imena konteksnog čvora (ukoliko joj se ne proslede argumenti) ili URI identifikator prostora imena prvog čvora u skupu čvorova koji joj je prosleđen kao argument.

12.4.2 Funkcije znakovnih nizova

Funkcije znakovnih nizova služe za obradu tekstualnog sadržaja elemenata i atributa XML dokumenta. Ovoj vrsti pripadaju funkcije *string(), starts-with(), contains(), substring(), substring-before(), substring-after(), string-length(), normalize-space()*. Funkcija *starts-with()* je prethodno pomenuta i objašnjena u primeru 66.

Funkcija *string()* pretvara argumente različitih tipova u znakovne nizove. Recimo, logičke vrednosti pretvara u znakovne nizove „true“ i „false“; brojeve u znakovne nizove formata koji se najčešće upotrebljavaju (recimo, „1994“ ili „3.14“); skupove čvorova u znakovne nizove koji predstavljaju ime prvog čvora u skupu.

Funkcija *contains()* prima dva argumenta i vraća logičku vrednost. Ukoliko se drugi argument sadrži u prvom, vraća vrednost *true* a u suprotnom vraća vrednost *false*.

Funkcija *substring-before()* za argumente prima dva znakovna niza. Kao rezultat vraća podniz prvog argumenta koji prethodi prvom pojavljivanju znakovnog niza drugog argumenta u njemu. Ukoliko se drugi znakovni niz ne pojavljuje u prvom, kao rezultat se vraća prazan znakovni niz. Ova funkcija je posebno značajna ukoliko treba izdvojiti recimo dan, mesec ili godinu iz datuma formata *MM/DD/GGGG*. Rezultat funkcije *substring-before(''04/20/2020'', ''/'')* jeste *04*. Slična ovoj funkciji je funkcija *substring-after()* koja takođe prima dva znakovna niza za argumente a za rezultat vraća podniz prvog znakovnog niza koji sledi za prvim pojavljivanjem drugog znakovnog niza u njemu. Recimo, rezultat

funkcije *substring-after("04/20/2020", "/")* je *20/2020*. Ako se tačno zna pozicija podniza (koji je potrebno izdvojiti) unutar znakovnog niza iz kog se izdvaja, bolje je upotrebiti funkciju *substring()*. Ona prima tri argumenta: znakovni niz iz kojeg treba izdvojiti podniz, redni broj znaka u nizu kojim počinje podniz i redni broj znakova koje treba kopirati (dužina podniza). Na primer, funkcija *substring('svemirski', 4, 3)* kao rezultat vraća znakovni niz „mir“.

Funkcija *string-length()* kao argument prima znakovni niz, a kao rezultat vraća broj koji predstavlja dužinu tog znakovnog niza. Važno je napomenuti da, osim toga što broji karaktere, ova funkcija broji i beline. Ukoliko argument nije zadat, *string-length()* kao rezultat vraća dužinu znakovne vrednosti konteksnog čvora. Na primer ukoliko posmatramo dokument „bioskop.xml“ u primeru 58, rezultat funkcije *string-length(/bioskop/film[position()=1]/reditelj)* biće *15*, zato što znakovni niz „Andy Muschietti“ ima 15 karaktera. Za uklanjanje belina i normalizovanje sadržaja elemenata koristi se funkcija *normalize-space()*.

12.4.3 Logičke funkcije

Logičke funkcije kao rezultat vraćaju logičke vrednosti, a za argument mogu primiti znakovni niz, broj, logičku vrednost ili skup čvorova. Jednostavne su za korišćenje i nema ih mnogo. Logičke konstante u XPath sintaksi ne postoje, ali kao zamene za njih koriste se funkcije *true()* i *false()*. Rezultat funkcije *true()* je uvek *true*, a rezultat funkcije *false()* je uvek *false*. Funkcija *not()* kao rezultat vraća suprotnu logičku vrednost od logičke vrednosti svog argumenta. Za pretvaranje vrednosti koje nisu logičkog tipa u logički tip koristi se funkcija *boolean()*. Ukoliko ovoj funkciji nije prosleđen argument, ona se odnosi na konteksni čvor. Pomenutom funkcijom se broj konvertuje u vrednost *true* ukoliko je različit od nule i nije *NaN*, u suprotnom, konvertuje se u vrednost *false*. Znakovni nizovi se konvertuju u vrednost *true* ukoliko sadrže bar jedan znak (karakter). U slučaju da im je dužina jednaka nuli, konvertuju se u vrednost *false*. Skup čvorova se konvertuje u vrednost *false* samo ako je prazan. Dovoljno je da taj skup sadrži jedan čvor da bi bio konvertovan u vrednost *true*.

12.4.4 Numeričke funkcije

Numeričkim funkcijama se mogu vršiti osnovne matematičke operacije nad brojevima, konvertovati vrednosti nekog drugog tipa u broj i zaokruživati brojevi.

Za konvertovanje vrednosti koje nisu brojevi u brojeve, koristi se funkcija *number()*. Ovom funkcijom, logička vrednost *true* se pretvara u broj *1*, a *false* u broj *0*. Ukoliko su karakteri znakovnog niza cifre, ova funkcija će ih pretvoriti u broj sa odgovarajućim ciframa. Na primer, znakovni niz „452“ će se konvertovati u broj *452*. U slučaju da znakovni nizovi imaju karaktere neke druge vrste, konvertovaće se u *NaN*. Dakle, znakovni niz „Svemir“ se konvertuje u *NaN*. Skupovi čvorova se prvo pretvaraju u odgovarajuće znakovne nizove, a zatim konvertuju u brojeve prema pomenutom pravilu.

Za zaokruživanje brojeva koriste se funkcije *round(), floor()* i *ceiling()*, koje za argument primaju jedan broj. Rezultat funkcije *round()* je najbliži ceo broj na koji se može zaokružiti argument. Brojevi koji su podjednako udaljeni od dva cela broja zaokružuju se na veći ceo broj. Recimo, broj *14.5* se zaokružuje na *15*, a broj *-14.5* na *-14*. Rezultat funkcije *floor()* je najveći ceo broj manji od argumenta, a rezultat funkcije *ceiling()* je najmanji ceo broj veći od argumenta.

Funkcija *sum()* za argumente prima skup čvorova koje prvo konvertuje u odgovarajuće znakovne nizove, a zatim u brojeve koje sabira i kao rezultat vraća dobijeni zbir.

Više o XPath funkcijama može se pročitati u [3].

13 Upotreba i struktura XQuery izraza

XQuery je tehnologija preporučena od strane W3C koja služi za postavljanje upita nad XML dokumentom. XQuery je izgrađen na osnovu XPath izraza, i koristi iste operatore i funkcije kao i XPath. Međutim, dok XPath može raditi samo sa jednim XML dokumentom, XQuery je dizajniran i za rad sa dobro formiranim delovima XML dokumenta, nizom delova XML dokumenata i nizom XML dokumenata. XQuery omogućava pisanje izraza (upita) koji služe za pronalaženje i izdvajanje elemenata i atributa iz XML dokumenta. Na primer, on može dati odgovor na sledeći zahtev: „Pronađi cenu filma „Interstellar“ u dokumentu „bioskop.xml“ i odredi kolika bi ona bila nakon sniženja od 20%“.

Kao i XPath, XQuery prepoznaje strukturu stabla XML dokumenta i razlikuje sedam vrsti čvorova: korenski čvor, čvorove elemenata, atributa, teksta, komentara, prostora imena i instrukcija za obradu. Osim samih čvorova, XQuery razaznaje i veze između njih: roditelje, decu, pretke i potomke.

XQuery sintaksna pravila:

XQuery je osetljiv na veličinu slova i sva imena elemenata i atributa koje on koristi moraju biti validna XML imena. Znakovni nizovi se, kao i u XML dokumentima, navode pod dvostrukim ili jednostrukim navodnicima. U XQuery izrazima se mogu pojaviti promenljive koje se definišu znakom „$“ za kojim sledi ime promenljive (primer: „$ime“). Sintaksa komentara se razlikuje od sintakse XML komentara. Komentar je ograđen zagradama i dvotačkama, na sledeći način:

```
(: Ovo je jedan komentar :)
```

13.1 XQuery izrazi

Postoji šest tipova XQuery izraza, i to su:

- Izrazi putanja;
- Konstruktori elemenata;
- FLWOR izrazi;
- Uslovni izrazi;
- Kvantifikovani izrazi;
- Izrazi koji koriste operatore i fukcije.

Pomoću ovih izraza mogu se pretraživati i veb dokumenti, izdvajati informacije koje se koriste u veb servisima i slično.

Za ilustrovanje primera XQuery izraza, koristiće se XML dokument „bioskop.xml“.

Primer 67: XML dokument "bioskop.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
      <bioskop>
        <film zanr="horor">
          <naziv> It </naziv>
          <reditelj>Andy Muschietti</reditelj>
          <trajanje>2h 15min</trajanje>
          <cena>350</cena>
        </film>
        <film zanr="komedija">
          <naziv> Ghostbusters </naziv>
          <reditelj>Ivan Reitman</reditelj>
```

```
        <trajanje>1h 45min</trajanje>
        <cena>300</cena>
      </film>
      <film zanr="fantastika">
        <naziv> Interstellar </naziv>
        <reditelj> Christopher Nolan</reditelj>
        <trajanje>2h 49min</trajanje>
        <cena>400</cena>
      </film>
    </bioskop>
```

13.1.1 Izrazi putanja

XQuery izrazi putanja služe kao vodiči kroz elemente i atribute XML dokumenta. Kako XQuery može raditi sa više XML dokumenata istovremeno, postoji funkcija *doc("ime_dokumenta.xml")* kojom se bira i otvara željeni XML dokument.

Sintaksa XQuery izraza putanja je jako slična sintaksi XPath izraza putanja.

Primer 68: XQuery izraz putanje

```
doc("bioskop.xml")/bioskop/film/naziv
```

Primer 68 pokazuje XQuery izraz putanje za pronalaženje svih „naziv“ podelemenata elementa „film“ u dokumentu „bioskop.xml“. Korakom */bioskop* pronalazi korenski element „bioskop“, korakom */film* pronalaze se svi „film“ podelementi elementa „bioskop“, i konačno, korakom */naziv* pronalaze se svi „naziv“ podelementi svih „film“ elemenata.

Rezultat XQuery izraza iz primera 68 je:

```
<naziv> Ghostbusters </naziv>
<naziv> It </naziv>
<naziv> Interstellar </naziv>
```

XQuery izrazi koriste i predikte kojima se dodatno sužava skup informacija koje se izdvajaju iz XML dokumenta. Kao što je poznato iz primera XPath izraza, predikti se navode unutar uglastih zagrada.

Primer 69 prikazuje XQuery izraz koji pronalazi sve filmove koji su podelementi elementa „bioskop“ i čija je cena manja od 350.

Primer 69: Predikat u XQuery izrazu

```
doc("bioskop.xml")/bioskop/film[cena<350]
```

Rezultat ovog izraza je potpuni element „film“ koji odgovara zadatim uslovima:

```
      <film zanr="komedija">
        <naziv> Ghostbusters </naziv>
        <reditelj>Ivan Reitman</reditelj>
        <trajanje>1h 45min</trajanje>
        <cena>300</cena>
      </film>
```

Već je napomenuto da XQuery izrazi koriste iste operatore kao i XPath.

Primer 70 prikazuje XQuery izraz koji iz dokumenta „bioskop.xml“ izdvaja cenu za 5 karata za film „IT“.

Primer 70: Operator u XQuery izrazu

```
doc("bioskop.xml")/bioskop/film[naziv="It"]/cena * 5
```

13.1.2 FLWOR izrazi

FLWOR je akronim za „For, Let, Where, Order by, Return“. Izgovara se isto kao i engleska reč „flower“. Klauza „for“ je iterativna konstrukcija. Ona promenljivoj dodeljuje rezultat zadatog XPath izraza nakon svake iteracije. Klauza „let“ takođe dodeljuje vrednosti promenljivim, ali se dodeljivanje vrši jednom, nema iterativnog postupka. Najbolji način da se prikaže razlika između „for“ i „let“ klauze je kroz primer.

Primer 71: „for“ klauza

```
for $x in (1 to 4)
return <provera>{$x}</provera>
```

Izraz iz primera 71 ima rezultat:

```
<provera>1</provera>
<provera>2</provera>
<provera>3</provera>
<provera>4</provera>
```

Primer 72: „let“ klauza

```
let $x := (1 to 4)
return <provera>{$x}</provera>
```

Izraz iz primera 72 ima rezultat:

```
<provera>1 2 3 4</provera>
```

Klauzom „where“ zadaju se uslovi pomoću kojih se dalje sužava skup čvorova koji je dobijen klauzama „for“ ili „let“. Klauzom „order by“ zadaje se pravilo po kome će dobijeni niz čvorova biti sortiran. Klauzom „return“ formira se rezultat FLWOR izraza. Ona se izvršava jednom za svaki čvor koji je prošao klauzu „where“, sve što se vrati njenim izvršavanjem predstavlja konačan rezultat celog FLWOR izraza. Ne moraju se sve pomenute klauze naći u jednom FLWOR izrazu.

Primer 73: FLWOR izraz

```
      for $x in doc("bioskop.xml")/bioskop/film
      where $x/cena < 400
      order by $x/naziv
      return $x/naziv
```

Primer 73 prikazuje FLWOR izraz koji u dokumentu "bioskop.xml" pronalazi sve elemente čija je cena manja od 400 i sortira ih prema nazivima.

Rezultat ovog FLWOR izraza je:

```
<naziv> Ghostbusters </naziv>
```

```
<naziv> It </naziv>
```

U primeru 73:

- Klauza „for“ selektuje „film“ podelemente elementa „bioskop“ i dodaje ih promenljivoj $x;
- Klauza „where“ selektuje samo one „film“ elemente čija je cena manja od 400 dinara;
- Klauza „oder by“ definiše sortiranje elemenata po nazivu;
- Klauza „return“ specifikuje da ono što treba da se vrati moraju biti nazivi filmova.

13.1.3 Konstruktori elemenata

Postoje XQuery izrazi koji kreiraju nove elemente. Oni se nazivaju konstruktorima elemenata.

Jedan tip konstruktora elemenata su direktni konstruktori. Oni koriste XML oznake i imena elemenata koja su njima zadata su konstantna. Direktni konstruktor prikazan je u primeru 74.

Primer 74: Direktni konstruktor

```
<film zanr="drama">
          <naziv>Gone Girl</naziv>
          <reditelj>David Fincher</reditelj>
          <trajanje>2h 29min</trajanje>
          <cena>350</cena>
</film>
```

U primeru 74, generisan je element „film“ koji ima atribute „zanr“ i sadrži podelemente „naziv“, „reditelj“, „trajanje“ i „cena“.

Unutar direktnih konstruktora, mogu se koristiti vitičaste zagrade „{}“ koje ograđuju druge XQuery izraze. Vitičaste zagrade odvajaju XQuery izraze od ostatka tekstualnog sadržaja. U rezultatu takvih konstruktora, svi ograđeni izrazi biće zamenjeni njihovim rezultatima.

Primer 75 prikazuje konstruktor elementa „film“ koji ima atribut „zanr“ i sadrži podelemente „naziv“, „reditelj“, „trajanje“ i „cena“. Pritom, konstruktor koristi promenljive $zanr, $naziv, $reditelj, $trajanje, i $cena kojima su već dodeljene vrednosti u nekom drugom delu upita, tako da $zanr ima vrednost „triler“, $naziv ima vrednost „<naziv> Gone Girl </naziv>“, $reditelj ima vrednost „<reditelj>David Fincher</reditelj>“, $trajanje ima vrednost „<trajanje>2h 29min</trajanje>“ i $cena ima vrednost „<cena>350</cena>“.

Primer 75: Konstruktor koji sadrži rezultate drugih XQuery izraza

```
let $zanr:="triler"
let $naziv:=<naziv>Gone Girl</naziv>
let $reditelj:=<reditelj>David Fincher</reditelj>
let $trajanje:=<trajanje>2h 29min</trajanje>
let $cena:=<cena>350</cena>

return <film zanr="{$zanr}">
          {$naziv}
          {$reditelj}
          {$trajanje}
          {$cena}
       </film>
```

Rezultat ovog konstruktora biće sledeći element:

```
<film zanr="drama">
          <naziv>Gone Girl</naziv>
          <reditelj>David Fincher</reditelj>
          <trajanje>2h 29min</trajanje>
          <cena>350</cena>
</film>
```

Osim direktnih konstruktora postoje i konstruktori čija se sintaksa razlikuje od XML sintakse. Oni počinju ključnom reči koja identifikuje vrstu čvora koji se generiše: *element* – za elemente, *attribute* – za atribute, *namespace* – za prostore imena, *comment* – za komentare, *processing-instruction* – za instrukcije za obradu, *text* – za tekstualni sadržaj i *document* – za korenski čvor. Nakon vrste čvora navodi se ime koje će taj čvor imati, a zatim se navodi i vrednost čvora unutar vitičastih zagrada.

Primer 76: Konstruktor

```
element film {
   attribute zanr {"drama" },
   element naziv { "Gone Girl"},
   element reditelj {"David Fincher"},
   element trajanje {"2h 29min "},
   element cena {"350"}
}
```

U primeru 76, generisan je već pomenuti element „film“ sa atributom „zanr“ i podelementima „naziv“, „reditelj“, „trajanje“ i „cena“.

Više o konstruktorima elemenata može se pročitati u [6].

13.1.4 Uslovni izrazi

Uslovni izrazi sastoje se od klauza *if*, *then* i *else*. Klauza *if* sadrži XQuery izraz koji predstavlja test čvorova. Taj izraz se obavezno navodi unutar običnih zagrada. Ukoliko je njegova istinitosna vrednost *true*, kao rezultat celog uslovnog izraza vraća se vrednost izraza koji je zadat klauzom *then*. U suprotnom, rezultat celog uslovnog izraza biće vrednost izraza koji je zadat klauzom *else*.

Primer 77: Primena klauze „if“

```
for $f in doc("bioskop.xml")/bioskop/film
return if ($f/@zanr="komedija")
then <komedija>{data($f/naziv)}</komedija>
else <ostalo>{data($f/naziv)}</ostalo>
```

Primer 77 prikazuje primenu klaze *if*. Iz dokumenta „bioskop.xml“ u primeru 67, izdvajaju se svi filmovi čiji je žanr „komedija“ pod istoimenim etiketama, a filmovi bilo kog drugog žanra izdvajaju se pod etiketama „ostalo“.

Rezultat uslovnog izraza iz primera 77 biće:

```
<ostalo>It</ostalo>
<komedija>Ghostbusters</komedija>
<ostalo>Interstellar</ostalo>
```

13.1.5 Kvantifikovani izrazi

Kvantifikovani XQuery izrazi u sebi sadrže klauze *some* ili *every* koji su ekvivalenti matematičkim kvantifikatorima „postoji“ i „svaki“, redom. Klauza *some* se koristi da bi se ispitalo da li postoji bar jedan čvor u posmatranom skupu čvorova koji ispunjava dati uslov, a klauza *every* da bi se ispitalo da li svi čvorovi posmatranog skupa zadovoljavaju dati uslov.

Primer 78: Primena klauze „some“

```
for $f in doc("bioskop.xml")/bioskop/film
where some $c in $f/cena satisfies ($c = 300)
return $f/naziv
```

Rezultat izraza iz primera 78 je naziv onog filma iz dokumenta „bioskop.xml“ čija je cena 300, ukoliko takav film postoji. Dakle, rezultat je:

```
<naziv> Ghostbusters </naziv>
```

Primer 79: Primena klauze „every“

```
for $f in doc("bioskop.xml")/bioskop/film
where every $n in $f/naziv satisfies contains($n, "I")
return $f/cena
```

Rezultat izraza iz primera 79 je cena onog filma iz dokumenta „bioskop.xml“ koji u svim svojim nazivima sadrži slovo „I“ (može se desiti da element „film“ ima više podelemenata „naziv“). Dakle, rezultat su cene filmova „It“ i „Interstelar“:

```
<cena>350</cena>
<cena>400</cena>
```

U slučaju da nijedan naziv nije sadržao slovo „I“, bila bi prijavljena greška.

13.1.6 Izrazi u kojima se koriste operatori

XQuery koristi iste operatore, koji odgovaraju osnovnim računskim operacijama, kao i XPath. Mala razlika nastaje u slučaju operatora za poređenje vrednosti. Zapravo, osim operatora =, !=, <, <=, >, >= koriste se i operatori *eq, ne, lt, le, gt* i *ge*. Razlika između ovih operatora može se videti na sledećim primerima.

Primer 80: Primena operatora „>“

```
doc("bioskop.xml")/bioskop/film/cena > 300
```

XQuery izraz iz primera 80 vraća vrednost *true* ukoliko bilo koji element „cena“ ima vrednost veću od 300.

Primer 81: Primena operatora „lt“

```
doc("bioskop.xml")/bioskop//film[@zanr="komedija"]/cena lt "380"
```

XQuery izraz iz primera 81 ispituje samo atomske vrednosti (ne može ispitati niz vrednosti) i vraća vrednost *true* ukoliko postoji element „cena“ čija je vrednost manja od 380.

Ukoliko bi se više elemenata „cena“ vratilo zadatom putanjom lokacije, bila bi prijavljena greška u primeru 81.

13.1.7 Izrazi u kojima se koriste funkcije

XQuery podržava istu biblioteku funkcija kao i XPath, i takođe dozvoljava da korisnici definišu nove funkcije kojima će proširiti ovu biblioteku. Tako se mogu definisati funkcije koje prikupljaju informacije iz dokumenta, računaju ih, sortiraju, menjaju i obrađuju ih na bilo koji način na koji korisnik želi.

Funkcije se mogu pozivati na bilo kojim mestima na kojima se mogu pozivati i drugi XQuery ili XPath izrazi.

Primer 82: Funkcija se poziva unutar elementa

```
      <naziv>{upper-case($nazivfilma)}</naziv>
```

Primer 83: Funkcija se poziva na mestu predikata u izrazima putanja

```
      doc("bioskop.xml")/bioskop/film[substring(naziv,1,5)='Ghost']
```

Primer 84: Funkcija se poziva u klauzi „let“

```
let $krataknaziv := (substring($nazivfilma,1,5))
```

Ukoliko u biblioteci funkcija koju koristi XQuery nema funkcija koje su korisniku potrebne, on ih može sam definisati. To može uraditi unutar XQuery izraza ili u posebnoj biblioteci. Sintaksa za definisanje funkcija od strane korisnika je prikazana je u primeru 85.

Primer 85: Definicija funkcije

```
declare function local:imeFunkcije($promenljiva as tip_podataka)
as tipPodatkaKojiSeVraća
{
 ...telo funkcije...
};
```

Tipovi podataka koji se dodeljuju promenljivim su uglavnom tipovi podataka definisani XML Schemom.

Primer 86: Definisanje funkcije koja računa cenu filma nakon popusta od 20%

```
declare function local:manjaCena($c as xs:decimal?,$p as xs:decimal?)
as xs:decimal?
{ let $popust := ($c * $p) div 100
return ($c - $popust) };
let $popust := 20
let $film := doc("bioskop.xml")/bioskop//film[@zanr="komedija"]
```

Funkcija se nakon definicije poziva na sledeći način:

```
return <manjaCena>{local:manjaCena($film/cena,$popust)}</manjaCena>
```

Rezultat je:

```
<manjaCena>240</manjaCena>
```

Više o XQuery izrazima može se naći u [7].

Zaključak

XML format omogućava dobro strukturiranje podataka, i čini ih lako dostupnim i pogodnim za dalju obradu. Jednostavnost pristupa informacijama unutar XML dokumenta mnogim aplikacijama olakšava korišćenje, skladištenje, razmenu, pa čak i prikaz informacija. Stoga, ovaj format je postao najpoželjniji u skoro svim veb aplikacijama. Konfiguracijske datoteke novijih aplikacija često imaju upravo XML format. Njegova velika prednost i razlog rasprostranjenog korišćenja jeste nezavisnost od platforme na kojoj se koristi, a potom i laka prenosivost. Takođe, podjednako lako ga mogu čitati i ljudi i mašine.

Dobro je napomenuti da softveri napisani u programskim jezicima C ili Java omogućavaju smeštanje XML podataka u bazu, ali i preuzimanje podataka iz iste. Zbog načina na koji XML strukturira i obeležava informacije etiketama, pretraživanje veb stranica je dodatno olakšano, a rezultati pretrage postaju precizniji. Prema tome, svako ko radi na kreiranju ili unapređivanju veb aplikacija će se u nekom trenutku susresti sa radom sa XML dokumentima.

Elektronske lekcije kreirane za potrebe ovog rada bi trebalo da omoguće jednostavno savladavanje osnova XML standarda. Lekcije su pažljivo napisane i prilagođene svima koji se prvi put susreću sa ovim formatom podataka. Pogodne su za sticanje početnog znanja iz ove oblasti i stvaranje dobre osnove koja se dalje može nadograđivati i unapređivati. Ilustrovane su jednostavnim primerima kako bi se čitaocima približile sve opisane strukture i upitni izrazi. Cilj elektronskih lekcija jeste da na lak i brz način osposobe korisnika da kreira validne XML dokumente za svoje potrebe i konstruiše različite vrste XQuery i XPath izraza koji će mu omogućiti pronalaženje željenih informacija unutar XML dokumenta.

Na internetu je dostupna besplatna i obimna literatura za učenje o XML formatu podataka i upotrebi XPath i XQuery upitnih izraza nad XML dokumentom. Međutim, literatura koja se na taj način može pronaći je najčešće na engleskom jeziku. Iz tog razloga učenje može biti otežano, jer je potrebno dobro poznavanje stranih stručnih izraza. Posebna pogodnost elektronskih lekcija o XML formatu podataka i upotrebi XPath i XQuery izraza za obilazak stabla XML dokumenta jeste ta što su napisane na srpskom jeziku. Lekcije su lako dostupne svima na platformi „eŠkola veba“.

Literatura

[1] Jurić N., Marić M., eŠkola veba, VII Simpozijum Matematika i primene, Matematički fakultet, Beograd, 5. novembar 2016.

[2] Jelena Graovac, Magistarski rad: *XML baze podataka u upravljanju leksičkim resursima*, magistarski rad, Matematički fakultet, Beograd, 2008: http://poincare.matf.bg.ac.rs/~jgraovac/publications/jt.pdf

[3] Elliotte Rusty Harold, W. Scott Means, *XML in a Nutshell (2nd Edition)*, O'Reilly & Associates, Inc, Sebastopol, 2002.

[4] Cvetana Krstev, *Kurs iz XML-a*: http://poincare.matf.bg.ac.rs/~cvetana/kurs-xml/xml-sadr.html

[5] Elektronski kurs XML tehnologija: https://www.link-elearning.com/kurs-XML-tehnologije-i-veb-servisi_384_4

[6] Zvanični sajt W3C (World Wide Web Consortium): https://www.w3.org/TR//xquery-31/

[7] Sajt platforme za interaktivno učenje veb tehnologija: https://www.w3schools.com/