XML

Autori:
Goran Savić
Milan Segedinac

1. XML

XML (Extensible Markup Language) je jezik za opis podataka. To je generalan jezik koji omogućuje opis proizvoljnih podataka. Jezik je dizajniran tako da bude čitljiv za čoveka, ali takođe i pogodan za procesiranje od strane računara. U nastavku je dat primer opisa podataka o studentima putem XML jezika.

```
<?xml version="1.0" encoding="UTF-8"?>
<studenti>
      <student id="1" obrisan="da">
            <ime>Marko</ime>
            <prezime>Markovic</prezime>
            <indeks>3229</indeks>
      </student>
      <student id="2">
            <ime>Petar</ime>
            <prezime>Petrovic</prezime>
            <indeks>6372</indeks>
      </student>
</studenti>
```

XML dokument se sastoji od hijerarhijski organizovanih **elemenata**. Svaki element je definisan početnim (otvarajućim) i krajnjim (zatvarajućim) **tagom**. Tagovi se definišu putem znakova < > u slučaju otvarajućeg taga, i znakova </ > u slučaju zatvarajućeg taga. U prikazanom primeru, element *studenti* je jedan od elemenata. Vidimo da počinje otvarajućim tagom <*studenti*>, a završava zatvarajućim tagom </*studenti*>. Ovaj element sadrži dva elementa *student*, od kojih svaki sadrži elemente *ime, prezime* i *indeks.* Elementi na dnu hijerarhije imaju sadržaj. Primer ovakvog elementa je element *ime*, koji ima sadržaj *Petar*.

Osim podelemenata i sadržaja, element može da sadrži i **atribute**, koji dodatno opisuju element. Atributi se navode u obliku parova naziv-vrednost u okviru otvarajućeg taga nakon naziva taga. U prikazanom primeru, element *student* ima atribute *id* i *obrisan*.

XML dokument može da počinje sa XML deklaracijom. To je prvi element u prikazanom primeru. Deklaracija opisuje sam dokument, npr. skup karaktera korišćen unutar dokumenta.

Komentari se u XML fajlu navode u obliku

```
<!— Tekst komentara -->
```

Pomenimo još i da postoji specijalni tip elementa, tzv. prazan element. Ovakav element ne sadrži podelemente niti sadržaj, pa se navodi specijalnim tagom koji istovremeno predstavlja i otvarajući i zatvarajući tag. Sintaksa za definisanje ovakvog taga je:

```
<naziv-taga />
```

Obzirom da XML elementi formiraju hijerarhiju, obično se koristi sledeća terminologija. Direktni podelementi nekog elementa se nazivaju deca (eng. *children*). Na primer, deca elementa *studenti* su dva elementa *student*. Direktno nadređeni element se naziva roditelj (eng. *parent*). Tako je elementima *student* roditelj element *studenti*. Elementi koji imaju istog roditelja nazivaju se srodnici (eng. *siblings*). U prethodnom primeru, elementi *ime, prezime* i *indeks* su srodnici. Takođe, dva elementa *student* su srodnici. Svi

podelementi nekog elementa, bez obzira na kojem nivou hijerarhije se nalaze, se nazivaju potomci (eng. *descendants*). Za element *studenti*, potomci su dva elementa *student*, ali i po dva elementa *ime*, *prezime* i *indeks*. Slično, postoji i pojam preci (eng. *ancestors*) za sve nadređene elemente, bez obzira na hijerarhijski nivo na kojem se nalaze. Tako su preci elementa *ime*, elementi *student* i *studenti*.

2. XML šema

Pomenuli smo da je XML generalno dizajniran tako da je moguće opisati proizvoljne podatke koristeći XML notaciju. Iz tog razloga, XML sam po sebi ne sadrži ograničenja u pogledu naziva elemenata i atributa, tipova vrednosti niti organizacije elemenata koje XML dokument može da sadrži. Videli smo u prethodnom primeru da smo za opis studenata uveli elemente *studenti, student*, *ime* itd, kao i da smo odlučili da hijerarhijski organizujemo elemente tako da element *studenti* sadrži sekvencu studenata.

Iako u opštem slučaju XML dokument može da sadrži proizvoljne elemente, pri pojedinačnim primenama XML jezika potrebno je specificirati kakvu strukturu XML dokument mora da ima. Na primer, možemo definisati da se za opis podataka o studentima mora koristiti XML dokument u kojem su dozvoljeni samo elementi *studenti*, *student, ime, prezime* i *indeks*, kao i to da organizacija elemenata mora biti takva da se element *student* mora definisati kao podelement elementa *studenti*. Specifikacija strukture XML dokumenta se naziva **šema XML dokumenta.**

Postoje različiti načini da se definiše struktura XML dokumenta. Najstariji jezik za opis XML šeme je DTD (Document Type Definition). Noviji jezik koji služi istoj svrsi se zove XML Schema.

Prema ovom jeziku, šema se definiše u XSD (XML Schema Definition) fajlu. Dat je primer XML šeme koja specificira kako podaci o studentima moraju biti definisani.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="studenti">
            <xs:complexType>
                  <xs:sequence minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="student" />
                  </xs:sequence>
            </xs:complexType>
      </xs:element>

      <xs:element name="student">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="ime" type="xs:string"/>
               <xs:element name="prezime" type="xs:string" />
               <xs:element name="indeks" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="id" use="required" type="xs:integer" />
            <xs:attribute name="obrisan">
               <xs:simpleType>
                  <xs:restriction base="xs:string">
                     <xs:enumeration value="da"/>
                     <xs:enumeration value="ne"/>
                  </xs:restriction>
               </xs:simpleType>
            </xs:attribute>
         </xs:complexType>
      </xs:element>
</xs:schema>
```

Na prikazanom primeru opisaćemo ključne elemente u opisu šeme. Kao što vidimo, i šema se zadaje u vidu XML dokumenta. Korenski element je *schema*. Atribut *xmlns* ovog elementa definiše prostor imena (eng. *namespace*) koji šema koristi. Prostori imena se koriste da se izbegnu konflikti u imenovanjima XML elemenata. Naime, pošto je XML proširiv jezik, moguće je uvoditi proizvoljne elemente. U slučaju istih imena uvedenih od strane različitih entiteta, važno je da bude jednoznačno određeno na koji se element misli. Iz tog razloga se elementi grupišu u prostore imena i pri navođenju naziva elementa se naglašava da se misli na taj naziv iz određenog prostora imena. Navođenje prostora imena se vrši putem prefiksa ispred imena elementa. Za svaki prostor imena se definiše prefiks koji će biti korišćen. U konkretnom primeru, šema je specificirana putem elemenata definisanih u prostoru imena *http://www.w3.org/2001/XMLSchema* i kada se misli na elemente iz ovog prostora imena u nastavku dokumenta se koristi prefiks *xs*.

Prvi *element* tag u dokumentu specificira da šema predviđa postojanje elementa pod nazivom *studenti*. Element može da sadrži jednostavnu vrednost, kao što su string ili broj, ili može da predstavlja novi kompleksni tip koji skladišti podelemente. U konkretnom slučaju, element *studenti* će sadržati sekvencu elemenata *student*. Na ovom mestu u XML dokumentu se samo referencira element *student*, koji je kasnije specificiran. Kada je reč o definisanju sekvence, vidimo da je moguće definisati i minimalni, odnosno maksimalni broj podelemenata (*minOccurs* i *maxOccurs*).

Element *student* sadrži sekvencu podelemenata *ime*, *prezime* i *indeks*. Ovi elementi direktno sadrže string vrednosti. Pored toga, element student ima i dva atributa *id* i *obrisan*. Atribut *id* je obavezan i vrednosti u ovom atributu su brojevi. Vrednost u atributu

obrisan mora biti string, s tim da je dodatno definisano ograničenje da su dozvoljene samo dve string vrednosti: *da* i *ne*.

Ako smo specificirali strukturu XML dokumenta, moguće je izvršiti proveru da li je određeni dokument napisan u skladu sa šemom. Ovaj postupak se zove **validacija** XML dokumenta, a dokument koji je u skladu sa šemom je **validan** XML dokument. XML dokument prikazan na početku ovog materijala je validan, jer je u skladu sa prikazanom šemom za opis studenata.

Validnost dokumenta treba razlikovati od toga da li je XML dokument ispravno formiran (eng. *well-formed*). Dokument je ispravno formiran ako je u skladu sa XML sintaksom, što znači da svaki otvarajući tag ima odgovarajući zatvarajući tag, da su elementi hijerarhijski organizovani i slično. Dakle, validnost se bavi i sadržajem dokumenta, dok na to da li je dokument ispravno formiran utiče samo to da li su ispoštovana sintaksna pravila XML jezika.

3. XPath

Podatke definisane u XML dokumentu moguće je pronaći i preuzeti putem upitnog jezika pod nazivom **XPath**. XPath omogućuje definisanje izraza koji kao rezultat vraćaju skup čvorova XML dokumenta koji zadovoljavaju kriterijum zadat izrazom. U tabeli su prikazani osnovni XPath izrazi koji preuzimaju elemente ranije prikazanog XML dokumenta o studentima.

Izraz	Primer	Opis

/	`/studenti/student/ime`	Preuzima decu trenutnog elementa
		Rezultat izraza iz primera:
		`<ime>Marko</ime>`
		`<ime>Petar</ime>`

//	`/studenti//prezime`	Preuzima potomke trenutnog elementa
		Rezultat izraza iz primera:
		`<prezime>Markovic</prezime>`
		`<prezime>Petrovic</prezime>`

..	`/studenti//prezime/..`	Preuzima roditelja selektovanog elementa
		Rezultat izraza iz primera:
		`<student id="1" obrisan="da">`
		`  <ime>Marko</ime>`
		`  <prezime>Markovic</prezime>`
		`  <indeks>3229</indeks>`
		`</student>`
		`<student id="2">`
		`  <ime>Petar</ime>`
		`  <prezime>Petrovic</prezime>`
		`  <indeks>6372</indeks>`
		`</student>`

@	`//@obrisan`	Preuzima atribute elemenata *Rezultat izraza iz primera:* `obrisan=da`

*	//studenti/student/*	Preuzimanje bilo kojeg elementa *Rezultat izraza iz primera:* <ime>Marko</ime> <prezime>Markovic</prezime> <indeks>3229</indeks> <ime>Petar</ime> <prezime>Petrovic</prezime> <indeks>6372</indeks>
@*	//studenti/student/@*	Preuzimanje bilo kojeg atributa *Rezultat izraza iz primera:* id=1 obrisan=da id=2

Za precizniji izbor elemenata koji se preuzimaju, mogu se koristiti XPath predikati. Predikati se navode u okviru znakova [ ] i opisani su kroz primere u narednoj tabeli.

Primer	Opis
/studenti/student[1]	Preuzima element na zadatoj poziciji *Rezultat izraza iz primera:* <student id="2"> <ime>Petar</ime> <prezime>Petrovic</prezime> <indeks>6372</indeks> </student>
/studenti/student[@obrisan]	Preuzima elemente koji imaju zadati atribut *Rezultat izraza iz primera*: <student id="1" obrisan="da"> <ime>Marko</ime> <prezime>Markovic</prezime> <indeks>3229</indeks> </student>
/studenti/student[@id=2]	Preuzima elemente koji imaju zadatu vrednost zadatog atributa *Rezultat izraza iz primera:* <student id="2"> <ime>Petar</ime> <prezime>Petrovic</prezime> <indeks>6372</indeks> </student>
/studenti/student[indeks=3229]	Preuzima elemente čiji podelement ima određenu vrednost *Rezultat izraza iz primera:* <student id="1" obrisan="da"> <ime>Marko</ime> <prezime>Markovic</prezime> <indeks>3229</indeks> </student>

Predikat u poslednja dva operatora koristio je operator = za poređenje vrednosti. Generalno, u XPath jeziku je moguće koristiti različite operatore za operacije nad rezultatima izraza. Često korišćen operator je | koji vrši spajanje rezultat više izraza. Na primer, izraz

```
//student/ime | //student/prezime
```

vraća rezultat

```
<ime>Marko</ime>
<prezime>Markovic</prezime>
<ime>Petar</ime>
<prezime>Petrovic</prezime>
```

Pored navedenih, moguće je koristiti i

- aritmetičke operatore: + (sabiranje), - (oduzimanje), * (množenje), div (deljenje), mod (ostatak pri deljenju)
- relacione operatore: != (nejednakost), > (veće), < (manje), >= (veće jednako), <= (manje jednako)
- logičke operatore: or (disjunkcija), and (konjukcija)

Takođe, u XPath izrazima je moguće koristiti i ugrađene funkcije, kao što su:

- boolean() – pretvara rezultat izraza u logičku vrednost
- concat() – spaja stringove
- contains() – da li string sadrži određeni string
- last() – vraća poslednji element iz liste elemenata
- number() – pretvara rezultat u broj
- position() – vraća poziciju elementa u listi elemenata
- starts-with() – vraća informaciju da li string počinje određenim stringom
- string-length() – vraća dužinu stringa
- substring() – vraća podstring određenog stringa
- sum() – vraća sumu vrednosti (prethodno konvertuje vrednosti u broj)

U tabeli su dati primeri nekih izraza koji koriste funkcije

Primer	Opis

`/studenti/student[last()]`	Preuzima poslednji element *student* u okviru elementa *studenti*
	Rezultat izraza iz primera:
	`<student id="2">`
	`  <ime>Petar</ime>`
	`  <prezime>Petrovic</prezime>`
	`  <indeks>6372</indeks>`
	`</student>`

`/studenti/student[position()>1]`	Preuzima sve elemente *student* koji se nalaze u okviru elementa *studenti* osim prvog
	Rezultat izraza iz primera:
	`<student id="2">`
	`  <ime>Petar</ime>`
	`  <prezime>Petrovic</prezime>`
	`  <indeks>6372</indeks>`
	`</student>`

`substring(//student[@id=2]/ime, 2,3)`	Svim elementima *ime* preuzima sadržaj 3 karaktera počevši od drugog karaktera *Rezultat izraza iz primera:* eta

Za kraj pregleda XPath sintakse, objasnićemo još i pojam **ose** (eng. *axis*). Osa definiše skup elemenata u odnosu na trenutni element. U narednoj tabeli dat je opis osa XPath jezika.

Osa	Opis
`ancestor`	Preuzima pretke tekućeg elementa
`ancestor-or-self`	Preuzima sve pretke tekućeg elementa i sam element
`attribute`	Preuzima atribute tekućeg elementa
`child`	Preuzima decu tekućeg elementa
`descendant`	Preuzima potomke tekućeg elementa
`descendant-or-self`	Preuzima potomke tekućeg elementa i sam tekući element
`following`	Preuzima sve elemente koji se u dokumentu nalaze nakon zatvarajućeg taga trenutnog elementa
`following-sibling`	Preuzima sve srodnike tekućeg elementa koji su u dokumentu navedeni nakon njega
`parent`	Preuzima roditelja tekućeg čvora
`preceding`	Preuzima sve elemente koji se u dokumentu nalaze pre otvarajućeg taga trenutnog elementa (isključujući pretke i atribute)
`preceding-sibling`	Preuzima sve srodnike tekućeg elementa koji su u dokumentu navedeni pre njega
`self`	Preuzima tekući čvor

Deo ranije prikazane sintakse predstavlja samo skraćenu sintaksu za funkcionalnosti koje ose omogućuju. Na primer, *child* je podrazumevana osa, pa su izrazi *//studenti/student* i *//studenti/child::student* ekvivalentni. Slično, roditelj se može preuzeti izrazom */studenti//prezime/..* ili navođenjem ose putem izraza */studenti//prezime/parent::**.

Primeri korišćenja osa dati su u sledećoj tabeli.

Izraz	Opis
`//studenti/descendant::indeks`	Preuzima sve *indeks* elemente koji su potomci elementa *studenti* *Rezultat izraza iz primera:* <indeks>3229</indeks> <indeks>6372</indeks>
`(//ime)[1]/following::prezime`	Preuzima sve elemente *prezime* koji slede nakon prvog elementa *ime* *Rezultat izraza iz primera:* <prezime>Markovic</prezime> <prezime>Petrovic</prezime>
`(//student/attribute::id)[1]`	Preuzima prvi iz liste atributa *id* elemenata *student* *Rezultat izraza iz primera:* id=1