**XML**

**(XSD/XSLT)**

**-Podsetnik -**

**1. Osnove XML-a**

**1.1. Uvod**

* XML skracenica oznacava E**X**tensible **M**arkup **L**anguage.
* XML je napravljen radi skladistenja i transporta podataka, stim sato ne treba zaboraviti da on ne predstavlja bazu podataka i ne moze samostalno da prenosi podatke. Za prenos podatka se koriste posebni protokoli kao sto su HTTP, FTP i sl.
* XML je dizajniran da bude citljiv od strane ljudi i masina.
* XML predstavlja softverski i hardverski nezavistan jezik za cuvanje/prenos podataka.
* XML nema nikakvu konkretnu funkciju on samo obezbedjuje cuvanje podatka u adekvatnom formatu, on se pre svega fokusira na sistematican pregled sacuvanih podatka.

**Primer:**

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

<poruke>

<prepiska>

<od>Milos</od>

<do>Jovan</do>

<naslov>Podsetnik - nastava</naslov>

<sadrzaj>Dodatni casovi iz predmeta Internet

tehnologije ce biti odrzani u subodu u 10 casova.

</sadrzaj>

</prepiska>

</poruke>

**XML ne koristi pred-definisane tagove kao HTML**

Tagove unutar XML-a definise sam korisnik i oni treba da na sto precizniji nacin opisu strukturu podatka koji se cuvaju u samom fajlu. Sami podaci se cuvaju unutar pocetnog i zavrsnog taga koji cuva taj konkretan podataka, npr. ako je potrebo sacuvati podataka o nekim korisnicima u sklopu liste korisnika gde ce se cuvati njihova imena, prezimena i broj telefona, prvo se stvara sam element koji predstavlja listu:

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

<listaKorisnika>

</listaKorisnika>

Sada se u okviru liste korisnika kreiraju tagovi koji predstavljaju same korisnike u lisi korisnika:

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

<listaKorisnika>

<korisnik>

</korisnik>

<korisnik>

</korisnik>

<korisnik>

</korisnik>

</listaKorisnika>

Za svakog korisnika se definisu posebno ime, prezime i broj telefona.

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

<listaKorisnika>

<korisnik>

<ime>Jovan</ime>

<prezime>Jovanovic</prezime>

<brojTelefona>065128934</brojTelefona>

</korisnik>

<korisnik>

<ime>Petar</ime>

<prezime>Petrovic</prezime>

<brojTelefona>06420928552</brojTelefona>

</korisnik>

<korisnik>

<ime>Marko</ime>

<prezime>Markovic</prezime>

<brojTelefona>062347345</brojTelefona>

</korisnik>

</listaKorisnika>

XML odvaja same podatke od nacina na koji se oni predstavljaju u programu, sam XML kod ne sadrzi nikakve instrukcije o tome na koji nacin da se podaci prikazu. Na ovaj nacin svaki program moze da iskoristi podatke sacuvane u XML falju i prikaze ih na poseban nacin.

**Primer**:

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

<meni>

<hrana>

<ime>Specijalitet za dvoje</ime>

<cena>1000</cena>

<opis>Mesano meso sa povrcem i krompirima</opis>

</hrana>

<hrana>

<ime>Pica Capricosa</ime>

<cena>700</cena>

<opis>Pica sa sunkom, sirom, pecurkama i maslinama</opis>

</hrana>

<hrana>

<ime>Mesano meso, mala porcija</ime>

<cena>600</cena>

<opis>Svinjso meso, junetina, piletima</opis>

</hrana>

<pice>

<ime>Sok od narandze</ime>

<cena>120</cena>

</pice>

<pice>

<ime>Crno vino</ime>

<cena>140</cena>

</pice>

</meni>

**1.2. XML struktura**

XML kod ima strukturu tzv. "drveta", odnosno, postoji osnovni, koreni, tag u sklopu kojed se vrsi grananje na druge tagove. Ovo struktura sustinski ima relaciju 1 ne vise ili Roditelj-Deca, jedan tag koji predtstavlja neki element je roditelj tagovima koji su definisani u sklopu njega, odnosno oni su njegova deca. Takodje tagovi koji predstavljaju decu takodje imaju svoje pod tagove kojima su oni roditelji, itd.

S obzirom da se tagovi definisu od strane samog korisnika i da imaju strukturu grananja

kao i to da se podaci cuvaju u tekstualnom obliku, sintaksa samog jezika je prilicno jasna.

**Sintaksa XML**

Pravila za sitaksu XML-a su veoma jednostavna i logicna.

* XML fajl mora da sadrzi jedan glavni ili koreni element/tag koji je roditelj svim ostalim elementima/tagovima koji su definisani u okviru njega.
* Svaki XML falj moze da sadrzi deklaraciju koja ako postoji mora da bude prva linja koda. Nista ne sme da se napise pre deklaracije ako ona postoji. Deklaracija ima sledeci oblik.

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

Atrbut encoding definise tip seme za kodovanje, UTF-8 predstavlja defoltnu kodnu semu za XML.

* Svaki pojedinacni element mora da sadrzi pocetni i zavrsni tag, primer:

<ime>Petar</ime>

<prezime>Petrovic</prezime>

Doci ce do greske ako se napise:

<ime>Petar

<prezime>Petrovic</prezime>

* Same definicije tagova su osetljive na velicinu slova, tzv. "case sensitive".

Ispravno je: <Ime>Marko</Ime>

Ispravno je: <ime>Marko</ime>

Nije ispravno: <Ime>Marko</ime> ili <ime>Marko</Ime>

**Posebni entiteti**

S obzirom da znakovi kao sto su <,>,"" i sl. imaju posebno znacenje nihovo direktno koriscenje u unutar tagova kao deo podatka koji se cuva nije dozvoljeno. Iz ovog razloga mogu se koristiti posebni entiteti koji zamenjuju njihvovo znacenje:

|  |  |
| --- | --- |
| Entitet | Znak |
| &lt; | < |
| &gt; | > |
| &amp; | & |
| &apos; | ' |
| &quot; | " |

**Komentari**

Sintaksa za komentare je sledeca: <!-- Ovo je tekst komentara -->

**Nova linija**

Nova linija se upisuje preko naredbe: LF.

XML dokument koji se pridrzava ovih pravila se zove "Pravilno formatiran" ili eng. "Well Formed" XML dokument.

**1.3. XML elemetni**

XML element predstavlja sve sto se nalazi izmedju pocetnog i zavrsanog taga kao i same tagove.

Svaki XML element moze da sadrzi:

* tekst
* atribute
* druge elemente (njegova deca odnosno pod elementi)
* ili njihovu kombinaciju

Ime elementa mora da prati sledeca pravila:

* Imena su osetljiva na velicinu slova.
* Ime mora poceti sa slovom ili donjom crtom.
* Ime ne moze da pocne sa slovima xml (xml, XML, Xml i sl.)
* Ime na moze da sadrzi praznine.

Stil ispisa imena

* Sa donjom crtom: <neko\_ime>
* Sa pocetnim velikim slovima: <NekoIme>
* Camel stil: <nekoImeIme>

Camel stil je popularan u JavaScript-u kao i u ostalim programskim jezicima.

**XML jezik je prosiriv**

Ako se definise sledeci XML element:

<osoba>

<ime>Marko</ime>

<prezime>Markovic</prezime>

<telefon>06243879845</telefon>

</osoba>

Prikaz ovih podatka u nekom programu moze da se izvrsi na sledeci nacin:

Ime: Marko

Prezime: Markovic

Kontakt tel: 06243879845

S obzirom da je svaki element unutar XML-a jedinstveno definisan preko svog taga moguce je definsati dodatne tagove koji ce da cuvaju jos podatka o nekoj osobi ali to nece uticati na progam koj prikazuje samo ranije definisane elemente, primer:

<osoba>

<ime>Marko</ime>

<prezime>Markovic</prezime>

<grad>Bograd</grad>

<ulicaBr>Glavna 56</ulicaBr>

<telefon>06243879845</telefon>

<email>miroslavmov@gmail.com</email>

</osoba>

Svaki program ce i dalje moci da jedinstveno nadje podatke o imenu, prezimenu i broju telefona. Ovo je jedna od glavnih odlika XML-a, dokument se moze prosirivati a da se pri tome nista ne menja u nekoj aplikaciji.

**1.4. XML atributi**

Svaki XML element moze da sadrzi atribute koji ga dodatno opisuju. Atributi kao i tagovi se definisu od strane programera i njihove vrednosti moraju biti unutar znakova "" kao i u slucaju HTML-a, opsti oblik:

ime\_atributa = "vrednost"

**Primer:**

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

<listaOsoba>

<osoba pol="muski">

<ime>Marko</ime>

<prezime>Markovic</prezime>

<telefon>0642498283</telefon>

<email>marko@gmail.com</email>

</osoba>

<osoba pol="zenski">

<ime>Jovana</ime>

<prezime>Jovanovic</prezime>

<telefon>0652234089</telefon>

<email>@gmail.com</email>

</osoba>

<osoba pol="muski">

<ime>Petar</ime>

<prezime>Petrovic</prezime>

<telefon>062435435</telefon>

<email>petar@gmail.com</email>

</osoba>

</listaOsoba>

Ako vrednost atributa i sama treba da sadrzi navodnike onda se moze koristiti kombinacija obicnih i duplih navodnika, primer:

<ime\_taga ime\_atributa='vrednost "vrednost" vrednost'>

**Upotreba elemenata naspram atributa**

Elementi cuvaju podake dok atributi detaljnije opisuju te podatke, jako je bitno zapamtiti da atributi na mogu da sadrze vise vrenosti kao elementi, nemaju strukturu grananja i nisu lako prosirivi. Iz ovog razloga treba izbegavati nepotrebne atribute vec sve predstaviti kao elemente. Ispod je prikazan los primer upotrebe atributa da se dodatno opise neka osoba.

<osoba pol="muski" datumRodjenja="1992-25-8">

<ime>Petar</ime>

<prezime>Petrovic</prezime>

<telefon tipTelef="mobilni">062435435</telefon>

<email>petar@gmail.com</email>

</osoba>

Daleko efikasniji nacin je:

<osoba>

<ime>Petar</ime>

<prezime>Petrovic</prezime>

<datumRodjenja>1992-25-8</datumRodjenja>

<pol>muski</pol>

<mobTel>062435435</mobTel>

<fikTel></fikTel>

<email>petar@gmail.com</email>

</osoba>

Atribute je najkorisnije upotrebiti kao meta podatke odnosno podatke treba cuvati kao elemente a kao atrubute treba cuvati ono sto pomaze da se ti podaci sto efukasnije prikazu i da im se brze pristupi. U primeru ispod atributi su iskorisceni da se identifikuju razlicite osobe na osnovu njihovog id-ja.

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

<listaOsoba>

<osoba id="01">

<ime>Marko</ime>

<prezime>Markovic</prezime>

<telefon>0642498283</telefon>

<email>marko@gmail.com</email>

</osoba>

<osoba id="02">

<ime>Jovana</ime>

<prezime>Jovanovic</prezime>

<telefon>0652234089</telefon>

<email>@gmail.com</email>

</osoba>

<osoba id="03">

<ime>Petar</ime>

<prezime>Petrovic</prezime>

<telefon>062435435</telefon>

<email>petar@gmail.com</email>

</osoba>

</listaOsoba>

**1.5. Prikaz XML dokumenta**

XML dokumenti se mogu prikazati u sklopu skoro svih Internet Browser programa. S obzirom da XML po definiciji ne sadri nikakve instrukcije o tome kako da se prikazu sami podaci on ce biti prikazan kao granjajuca struktura elemenata. U vecini slucajeva svaki element ce imati znak - ili + ispred sebe kako bi se sakrio ili prikazao sadrzaj tog elementa.

Posto tagove XML dokumenta smislja programer, internet pretrazivaci ne znaju kako da procesiraju te tagove i zbog toga samo prikazuju sadrzaj onakav kakav jeste.

**Koriscenje CSS za stilizovanje XML dokumeta**

Za dodatno stilizovanje XML dokumenta radi nesto lepseg prikaza korisniku moze da se iskoristi CSS. Uvoz CSS fajla se obavlja preko naredbe:

<?xml-stylesheet type="text/css" href="putanja\_do\_css\_fajla"?>

Unutar CSS izgled/stil elemenata XML dokumenta se definise preko naziva njihovih tagova s obzirom da su oni jedinstveni.

**Primer:**

Sadrzaj XML dokumenta:

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

<?xml-stylesheet type="text/css" href="**stil\_primer\_4.css**"?>

<listaKorisnika>

<naslov>Lista korisnika</naslov>

<korisnik>

<ime>Jovan</ime>

<godiste>1986</godiste>

<email>jovan@gmail.com</email>

<datumReg>20-06-2015</datumReg>

</korisnik>

<korisnik>

<ime>Marko</ime>

<godiste>1995</godiste>

<email>marko@gmail.com</email>

<datumReg>12-03-2014</datumReg>

</korisnik>

<korisnik>

<ime>Marija</ime>

<godiste>1988</godiste>

<email>marija@gmail.com</email>

<datumReg>14-05-2015</datumReg>

</korisnik>

<korisnik>

<ime>Stefan</ime>

<godiste>1992</godiste>

<email>stefan@gmail.com</email>

<datumReg>10-08-2012</datumReg>

</korisnik>

<korisnik>

<ime>Vladimir</ime>

<godiste>1985</godiste>

<email>vladimir@gmail.com</email>

<datumReg>06-10-2010</datumReg>

</korisnik>

<korisnik>

<ime>Milica</ime>

<godiste>1990</godiste>

<email>milica@gmail.com</email>

<datumReg>12-06-2014</datumReg>

</korisnik>

<korisnik>

<ime>Zeljko</ime>

<godiste>1983</godiste>

<email>zeljko@gmail.com</email>

<datumReg>22-01-2012</datumReg>

</korisnik>

</listaKorisnika>

Sadrzaj CSS fajla **stil\_primer\_4.css**:

listaKorisnika {border:3px solid crimson;

border-radius:8px;

background-color:white;

overflow:auto;

width:600px;

padding:10px;

margin:10px;}

/\* svi elementi koji treba posebno da se definisu trebaju da

imaju definisano display:block; \*/

naslov {display:block;

font-size:22px;

font-weight:bold;

text-align:center;

text-decoration:underline;

}

korisnik {border-radius:6px;

background-color:aqua;

display:block;

width:180px;

float:left;

padding:5px;

margin:5px;}

ime {display:block;}

godiste {display:block;}

email {display:block;}

datumReg {display:block;}

Rezultat:



**NAPOMENA:** Formatiranje i stilizovanje XML dokumenta sa CSS-om nije preporuceno, bolje je koristiti JavaScript ili XSLT.

**1.6. XML dokumenti**

XML dokument sa pravilnom sintaksom se naziva "Dobro formatiran" (eng. "Well Formated"). Dokument koji je i "validan" mora da se pridrzava specificirane definicije dokumenta ili DTD, eng. Document Type Definition. Za ovu svrhu se takodje moze koristiti i XML Schemas.

Ovi metodi za definisanje sadrzaja samog XML dokumenta su korisni kada podatke koristi grupa ljudi stoga je i potrebno da se nadje dogovor o univerzalnom nacinu cuvanja podatka u XML-u, o njegovoj strukturi, imenima tagova/elemenata itd. Uvozenja DTD fajla u XML se postize preko:

<!DOCTYPE ime\_taga System "ime\_DTD-dokumenta">

**Primer**, XML:

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

<!DOCTYPE listaKorisnika SYSTEM "**def\_primer\_5.dtd**">

<listaKorisnika>

<korisnik>

<ime>Jovan</ime>

<godiste>1986</godiste>

<email>jovan@gmail.com</email>

<datumReg>20-06-2015</datumReg>

</korisnik>

<korisnik>

<ime>Marija</ime>

<godiste>1988</godiste>

<email>marija@gmail.com</email>

<datumReg>14-05-2015</datumReg>

</korisnik>

</listaKorisnika>

Sadrzaj DTD dokumenta **def\_primer\_5.dtd**:

<!DOCTYPE listaKorisnika

[

<!ELEMENT korisnik (ime,godiste,email,datumReg)>

<!ELEMENT ime (#PCDATA)>

<!ELEMENT godiste (#PCDATA)>

<!ELEMENT email (#PCDATA)>

<!ELEMENT datumReg (#PCDATA)>

]>

DTD dokument iznad se tumaci kao:

* !DOCTYPE listaKorisnika - definise koreni element XML dokumenta
* !ELEMENT korisnik (ime,godiste,email,datumReg) - definise da ovaj element moza da sadrzi samo elemente: ime, godiste, email i datumReg.
* !ELEMENT ime / godiste / email / datumReg (#PCDATA) - definise da je podatkla tipa PCDATA (tekst).

**1.7. XML Schema - osnove**

XML Schema opisuje strukturu samog XML dokumenta kao i DTD. Shema je "direktna" XML alternativa DTD-u, odnosno, ona se definise u samom XML dokumentu.

Ako je definisan element:

<korisnik>

<ime>Jovan</ime>

<godiste>1986</godiste>

<email>jovan@gmail.com</email>

<datumReg>20-06-2015</datumReg>

</korisnik>

Format Shemea koda koji definise njegov sadrzaj je:

<xs:element name="korisnik">

<xs:complexType>

<xs:sequence>

<xs:element name="ime" type="string">

<xs:element name="godiste" type="string">

<xs:element name="email" type="string">

<xs:element name="datumReg" type="string">

</xs:sequence>

</xs:complexType>

</xs:element>

Ovaj kod se tumaci kao:

* <xs:element name="korisnik"> - definise se element "korisnik".
* <xs:complexType> - element "korisnik" je kompleksnog tipa, sadrzi druge elemente.
* <xs:sequence> - element "korisnik" ima sledeci niz pod elemenata.
* <xs:element name="ime" type="string"> - definise element "ime" tipa string, ovo vazi i za ostale pod elemente.

Prednosti XML Schemas

* XML Schemas se pise u XML jeziku.
* Kao i sam XML ove definicije sadrzaja su prosirive.
* Podrzava definiciju tipa podataka koji se cuva u elementu.
* Imaju slicnu sintaksu kao i sam XML.

**2. XML Schema**

**2.1. Osnove**

XML Schema opisuje stukturu XML dokumenta. U svakom trenutku na stotine standardizovanih XML formata je u dnevnoj upotrebi i format mnogih od njih je definisan preko XML Schemas.

Svrha XML Schemas je da definise ispravne/legalne elemnte koji mogu da budu deo nekog XML dokumenta, slicno kao i DTD, XML Schema:

* Definise elemente koji se mogu pojaviti u XML dokumentu.
* Definise atribute koji se mogu pojaviti u XML dokumentu.
* Definise koji elementi su pod-elementi (deca) drugih elemenata (roditelji).
* Definise redosled pod-elemenata.
* Definise defoltne i fiksne vrednosti za elemente i atribute.

Prednosti upotrebe:

* XML Schema je XML alternativa DTD-u.
* Daleko je efikasnija od DTD-a.

**Napomena:** XML Schemas se poneka nazivaju i XSD ili **X**ML **S**chema **D**efinition.

**XML Schemas podrzava tipove podatka.**

Jedna od najacih strana XML Schemas je to sto podrzava definisanje tipa podatka koji se cuva u sklopu nekog XML elementa. Sa podrskom za tipove podatka:

* Lakse je opisati dozvoljeni sadrzaj dokumenta.
* Lakse je proveriri ("validirati") ispravnost podataka.
* Lakse je raditi sa podacima iz baze podataka.
* Lakse je definisati ogranicenja za vrednosti podataka.
* Lakse je definisati format podataka.
* Laksa je konverzija podataka.

**XML Schemas koristi XML sintaksu**

Jedna od velikih prednosti XML Schemas je to sto se pisu u XML jeziku tako da nije potrebno uciti novi jezik.

**XML Schemas osiguravaju tacan prenos podatka**

Pri slanju podataka od posaljioca ka primaocu potrebno je da obe strane imaju ista ocekivanja po pitanju formata podataka. Ako se salje, npr. datum 04-09-2015, ovaj datum he moze protumaciti kao 4 septembar 2015 ili 9 april 2015. Medjutim XML dokumat sa sa elementom:

<datum type="date">2015-09-04<datum>

osigurava da ce sve strane znati da se radi o 4 septembru 2015 jer tip "date" zahteva format: GGGG-MM-DD.

**XML Schemas je prosiriv**

XML Schemas je prosiriv jer ja napisan u XML-u. Ovo omogucava da se isti kod iznova koristi, stvaranje posebnih tipova podatkaka na osnovu opstih, koriscenje vise XML Schemas u istom XML dokumentu.

**Potreba za Validnim sadrzajem i podacima**

Iako XML dokument moze da bute dobro formatiran i tacan po pitanju sintakse on mora i da bude validan po pitanju sadrzaja ili ce doci do problema, Mora se znati koji elementi su dozvoljeni, **koji elementi su kompleksni a koji su jednostavni itd.**

**2.2. Upotreba**

XML Schema ili XSD fajlovi se cuvaju sa **.xsd** ekstenzijom.

Jednostavan XML fajl:

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

<poruka>

<posaljilac>Jovan</posaljilac>

<primalac>Marko</primalac>

<naslov>Izvestaj</naslov>

<sadrzaj>Ukpan promet za 12 sep 2015 je 34.600,00 din.</sadrzaj>

<vremeSlanja>2015-09-12</vremeSlanja>

</poruka>

XSD fajl "**poruka.xsd**" koji definise elemente gore definisanog XML fajla:

<?xml version="1.0"?>

<xs:schemas>

<xs:element name="poruka">

<xs:complexType>

<xs:sequence>

<xs:element name="posaljilac" tupe="xs:string">

<xs:element name="primalac" tupe="xs:string">

<xs:element name="naslov" tupe="xs:string">

<xs:element name="sadrzaj" tupe="xs:string">

<xs:element name="vremeSlanja" tupe="date">

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schemas>

**Struktura**

Svaki XSD fajl pocinje sa zaglavljem/definicijom, zatim se pise **<schema>** element. <schema> element je koreni elemnt svakog XSD fajla, primer:

<?xml version="1.0"?>

<xs:schema>

. . .

. . .

</xs:schema>

<schema> element moze da sadrzi neke atribute, primer:

<?xml version="1.0"?>

<xs:schema

xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNameSpace="http://www.w3schools.com"

xmlns="http://www.w3schools.com"

elementFormDefault="qualified"

>

. . .

. . .

</xs:schema>

U primeru iznad definsana su cetiri atibuta: **xmlns:xs**, **targetNameSpace**, **xmlns** i **elementFormDefault**.

Atribut **xmlns:xs**

xmlns:xs="http://www.w3.org/2001/XMLSchema"

naznacava da su elementi i tipovi podatka preuzeti sa "http://www.w3.org/2001/XMLSchema". Takodje specificira da ti elementi i tipovi podataka treba da imaju prefiks xs.

Atribut **targetNamespace**

targetNamespace="http://www.w3schools.com"

naznacava da su elementi koji su definisani u ovom fajlu sa "http://www.w3schools.com".

Atribut **xmlns**

xmlns="http://www.w3schools.com"

naznacava da je defoltno ime "http://www.w3schools.com".

Atribut **elementFormDefault**

elementFormDefault="qualified"

naznacava da svaki element koji se koristi u nekom XML dokumentu mora biti odrejen sa namespace-om.

**Referenciranje XSD fajla u XML-u**

Ovaj XML fajl referencira na XSD fajl poruka.xsd:

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

<poruka

**xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**

**xsi:noNamespaceSchemaLocation="poruka.xsd"**

>

<posaljilac>Jovan</posaljilac>

<primalac>Marko</primalac>

<naslov>Izvestaj</naslov>

<sadrzaj>Ukpan promet za 12 sep 2015 je 34.600,00 din.</sadrzaj>

<vremeSlanja>2015-09-12</vremeSlanja>

</poruka>

Atribut:

* xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>" - govori XML-u da ovaj dokument treba da se validira naspram nakog XSD-a.
* xsi:noNamespaceSchemaLocation="poruka.xsd" - specificira gde se XSD fajl nalazi, u ovom slcaju u isto folderu kao i XML fajl.

**2.3. Jednostavni tipovi**

Kao sto je objasnjeno XSD (XML Schemas) definise elemente za neki XML fajl.

**2.3.1. Jednostavni elementi**

Jednostavni element je element koji sadrzi smo tekst i ne moze da sadrzi nikakve druge pod elemente ili atribute. Iako jednostavan element sadrzi samo tekst taj tekst moze biti razlicitih tipova, taj tip moze biti jedan od tipova iz XSD definicije kao sto su: boolean, string, date itd. ili neki poseban tip definisan od strane programera. Moguce je uneti ogranicenja na vrednosti tipova podataka da bi se ogranicio njihov sadrzaj, takodje je moguce zahtevati da podaci budu uneti u odredjenom formatu/semi.

**Definicija**

Definisaje jednostavnog elementa, opsti oblik:

<xs:element name="ime\_elementa" type="tip\_podatka">

Neki od defoltnih tipova podatka su:

* xs:string
* xs:decimal
* xs:integer
* xs:boolean
* xs:date
* xs:time

Primer:

XML kod:

<ime>Marko</ime>

<brojGod>29</brojGod>

<datumRodj>1986-05-22</datumRodj>

<telefon>062/2323525</telefon>

XSD za XML podatke iznad:

<xs:element name="ime" type="xs:string">

<xs:element name="brojGod" type="xs:integer">

<xs:element name="datumRodj" type="xs:date">

<xs:element name="telefon" type="xs;string">

**Defoltne i fiksne vrednosti za jednostavne elemente**

Jednostavni elementi mogu da imaju definisanu defoltnu ILI fiksnu vrednost. Defoltna vrednost se dodeljuje elementu ako to nije uradjeno na pocetku.

Definisanje defoltne vrednosti, opsti oblik:

<xs:element name="ime\_elementa" type="tip\_podatka" default="def\_vrednost"/>

Definisanje fiksne vrednosti, opsti oblik:

<xs:element name="ime\_elementa" type="tip\_podatka" fixed="fiks\_vrednost"/>

**2.3.2. Jednostavni atributi**

Svi atributi su po definiciji jednostavni tipovi.

Iako jednostavni elementi ne mogu da imaju atribute, ako element ima atribut on je kompleksnog tipa, medjutim sam atribut je uvek jednostavnog tipa.

Definicija atributa, opsti oblik:

<xs:attribute name="ime\_atributa" type="tip\_podatka"/>

Kao i kod elemenata defoltni tipovi podatka su: xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time.

**Primer:**

XML element sa atributom: <ime jezik="srp">Marko</ime>

XSD definicija atributa: <xs:attribute name="jezik" type="xs:string"/>

**Defoltne i fiksne vrednosti za jednostavne elemente**

Atributi mogu da imaju definisanu defoltnu ILI fiksnu vrednost.

Definisanje defoltne vrednosti, opsti oblik:

<xs:attribute name="ime\_atributa" type="tip\_podatka" default="def\_vrednost"/>

Definisanje fiksne vrednosti, opsti oblik:

<xs:attribute name="ime\_ atributa" type="tip\_podatka" fixed="fiks\_vrednost"/>

**Opcioni i obavezni atributi**

Atributi su opcioni po defoltu, da bi se definisalo da je atribut obavezan, koristiti use atribut u XSD-u, opsti oblik:

<xs:attribute name="ime\_atributa" type="tip\_podatka" use="required"/>

**Ogranicenje sadrzaja**

Kada neki element ili atribut ima definisan tip podatka, njemu su automatski nametnuta odredjena ogranicenja po pitanju vrednosti. Ako je neki XML element tipa "xs:date", a u taj element je uneta string vrednost, npr, "drugi septembar 2015", doci ce do geske jer se ocekuje format tipa 2015-09-02 (GGGG-MM-DD).

**2.4. XSD ogranicenja (eng. Facets)**

Ogranicenja se koriste da bi se definisale prihvatljive vrednosti za elemente i atribute.

**2.4.1. Ogranicenja za tip integer**

U primeru ispod, element "starost" tipa integer moze da ima vrednosti izmedju 0 i 120.

<xs:element name="starost">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:minInclusive value="0"/>

<xs:maxInclusive value="120"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

U primeru ispod, element "godiste" tipa integer moze da ima cetiri cifre stim sto prva cifra moze da ima samo vrednosti 1 ili 2.

<xs:element name="godiste">

<xs:simpleType>

<xs:restriction base="integer">

<xs:pattern value="[1-2][0-9][0-9][0-9]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

**2.4.2. Ogranicenja za tip string**

**Ogranicenje na set vrednosti za tip string**

U primeru ispod, element "proizvodjac" moza da ima vrednosti: Addidas, Nike, Reebok, Assics, Kappa.

<xs:element name="proizvodjac">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="Addidas"/>

<xs:enumeration value="Nike"/>

<xs:enumeration value="Reebok"/>

<xs:enumeration value="Assics"/>

<xs:enumeration value="Kappa"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

**Ogranicenje na opseg vrednosti za string**

U primeru ispod, element "slovo" moza da ima samo jedno malo slovo od a do z, [a-z].

<xs:element name="slovo">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="[a-z]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

U primeru ispod, element "inicijali" tipa string moza da sadrzi samo tri velika slova od a do z, [A-Z][A-Z][A-Z].

<xs:element name="inicijali">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="[A-Z][A-Z][A-Z]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

U primeru ispod, element "inicijali" tipa string moze da ima tri velika ili mala slova od a do z, [a-zA-Z][a-zA-Z][a-zA-Z].

<xs:element name="inicijali">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

U primeru ispod, element "izborSlova" tipa string moze da ima jedno malo slovo od ponudjenih: a, b ili c, [abc].

<xs:element name="izborSlova">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="[abc]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

U primeru ispod, element "slovo" tipa string kao vrednost moze da ima nula ili vise malih slova od a do z, ([a-z])\*.

<xs:element name="nizSlova">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="([a-z])\*"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

* Mala i velika slova, neograniceno: value="([a-zA-Z])\*"
* Velika i mala slova i brojevi, neograniceno: value="([a-zA-Z0-9])\*"
* Mala slova i brojevi, neograniceno: value="([a-z0-9])\*"
* Velika slova i brojevi, neograniceno: value="([A-Z0-9])\*"

U primeru ispod, element "pol" tipa string moza da ima vrednosti ili "muski" ili "zenski", muski|zenski.

<xs:element name="pol">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="muski|zenski"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

**Ogranicenja na duzinu tipa string**

Ogranicenja koja definisu duzinu elementa tipa string:

<xs:length value="broj\_slova"/> - definisan je fiksni broj slova za element.

<xs:minlength value="broj\_slova"/> - definisan je minimalni broj slova.

<xs:maxlength value="broj\_slova"/> - definisan je maksimalni broj slova.

<xs:element name="ime">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxlength value="30"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

U primeru ispod, element "lozinka" tipa string moza da ima minimum sest karaktera.

<xs:element name="lozinka">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:minLength value="6"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

**Ogranicenja na beline/prazan prostor**

Da bi se specificiralo kako da se postupi sa belinama u tekstu koristi se "whiteSpace" ogranicenje. Moguce vrednosti su: preserve, replace ili collapse.

* <xs:whiteSpace value="preserve"/> - sve unete beline u tekstu ce biti sacuvane.
* <xs:whiteSpace value="replace"/> - sve beline poput taba, visestrukih space-eva i sl ce biti zamenjeni sa jednim space-om.
* <xs:whiteSpace value="collapse"/> - sve beline ce biti neutralisane, spojene reci.

<xs:element name="adresa">

<xs:simpleType>

<xs:restriction base="string">

<xs:whiteSpace value="replace"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

**2.5. Kompleksni tipovi**

Kompleksni element sadrzi druge elemente i/ili atribute. Postoje cetiri razlicita tipa kompleksnih elemenata i svaki od njih moze sadrzati atribute:

* Prazni elementi.
* Elementi koji samo sadrze druge elemente.
* Elementi koji sadrze samo tekst.
* Elementi koji sadrze i druge elemente i tekst.

**2.5.1. Kompleksni element koji samo sadrze druge elemente**

Definisanje kompleksnog elementa "radnik" u XML-u:

<radnik>

<ime>Marko</ime>

<prezime>Markovic</prezime>

<staz>8</staz>

</radnik>

Definisanje elementa "radnik" u XSD-u:

<xs:element name="radnik">

<xs:complexType>

<xs:sequence>

<xs:element name="ime" type="xs:string"/>

<xs:element name="prezime" type="xs:string"/>

<xs:element name="staz" type="xs:positiveInteger"/>

</xs:sequence>

</xs:complexType>

</xs:element>

Tag </xs:sequence> obavezuje da se elementi ime i prezime moraju pojaviti u redosledu definisanom u elementu "radnik".

**2.5.2. Kombinovani kompleksni elementi**

Kombinovani kompleksni element sadrzi atributre, druge elemente i tekst.

Primer: XML element "poruka" sadrzi i dodatne elemente i tekst

<poruka>

Porstovani, <ime>Jovan</ime>

Vasa porudzbina <idPorudzbine>832</idPorudzbine>

evidentirana je<datum>2015-10-14</datum>

</poruka>

Sadrzaj XSD fajla koji definise element poruka, da bi bilo omoguceno da se teks pojavi izmedju definicija elemenata u sklopu taga complexType mora biti definisan atribut mixed="true":

<xs:element name="poruka">

<xs:complexType mixed="true">

<xs:sequence>

<xs:element name="ime" type="xs:string"/>

<xs:element name="idPorudzbine" type="positiveInteger"/>

<xs:element name="datum" type="xs:date"/>

</xs:sequence>

</xs:complexType>

</xs:element>

**2.5.3. Indikatori**

Indikatori omogucavaju da se kontrolise na koji nacin se elementi koji su definisani u XSD fajlu mogu pojavljivatu i XML-u. Postoji ukupno sedam indikatora grupisanih u tri grupe:

Indikator redosleda:

* All
* Choice
* Sequence

Indikator pojave:

* maxOccurs
* minOccurs

Indikator grupe:

* Group name
* attributeGroup name

**Indikator redosleda**

Indikatori redosleda se koriste da se definise redosled pojavljivanja elemenata definisanih u XSD-u. Postoje tri indikatora redosleda:

* <all> - specificira da se pod-elementi nekog elementa mogu pojavitit u bilo kom redosledu i da se svaki pod-element moza pojaviti samo jedanput.
* <choice> - specificira da se moze pojaviti bilo koji od definisanih pod-elemenata.
* <sequence> - specificira da se pod-elementi mog pojaviti iskljucivo redosledom kojim su definisani u XSD-u.

Primer: <all> indikator

<xs:element name="korisnik">

<xs:complexType>

<xs:all>

<xs:element name="ime" type="xs:string"/>

<xs:element name="prezime" type="xs:string"/>

</xs:all>

</xs:complexType>

</xs:element>

Primer: <chioce> indikator

<xs:element name="osoba">

<xs:complexType>

<xs:choice>

<xs:element name="zaposleni" type="xs:string"/>

<xs:element name="korisnik" type="xs:string"/>

</xs:choice>

</xs:complexType>

</xs:element>

**Indikator pojave**

Indikatori pojave se koriste da bi definisalo koliko puta neki pod-element moza da se pojavi u sklopu njegovog nad-elementa u XML-u. Postoje dva indikatora pojave:

<maxOccurs> - specificira maksimalni broj puta koliko neki element moze da se pojavi.

<minOccurs> - specificira minimalni broj puta koliko neki element moze da se pojavi.

Primer:

XSD fajl "**korisnik.xsd**":

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

<xs:schema

xmlns:xs="http://www.w3.org/2001/XMLSchema"

elementFormDefault="qualified"

>

<xs:element name="korisnik">

<xs:complexType>

<xs:choice>

<xs:element name="ime" type="xs:string"/>

<xs:element name="adresa" type="xs:string"

maxOccurs="5" minOccurs="1"/>

<xs:element name="telefon" type="xs:string"

maxOccurs="5" minOccurs="1"/>

</xs:choice>

</xs:complexType>

</xs:element>

</xs:schema>

XML fajl:

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

<listaKorisnika

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="**korisnik.xsd**"

>

<korisnik>

<ime>Marko</ime>

<adresa>Bgd Glavna 23</adresa>

<adresa>Bgd Beogradska 85</adresa>

<telefon>011878239</telefon>

<telefon>011492738</telefon>

</korisnik>

<korisnik>

<ime>Jovan</ime>

<adresa>Bgd Dunavska 51</adresa>

<telefon>011342384</telefon>

</korisnik>

</listaKorisnika>

**Napomena:** Da bi se element pojavio neogranicen broj puta postaviti maxOccurs="unbounded".

**Indikator grupe**

Indikatori grupe se koriste da bi se definisali povezani elementi. Grupa je definisana preko <group> taga, primer:

<xs:group name="ime\_grupe">

. . .

</xs:group>

Unutar grupe se definise jedan od indikatora redosleda all, choice ili sequence, zatim same elemente.

Kreiranje grupe "**osobaGrupa**":

<xs:group name="osobaGrupa">

<xs:sequence>

<xs:element name="ime" type="xs:string"/>

<xs:element name="prezime" type="xs:string"/>

<xs:element name="jmbg" type="xs:string"/>

<xs:element name="datumRodjenja" type="xs:date"/>

<xs:sequence>

<xs:group>

Referenciranje grupe "osobaGrupa" u nekoj drugoj definiciji unutar XSD-a, referenciranje se postize preko atributa ref, primer, ref="ime\_grupe":

<xs:element name="osoba" type="kontaktPodaci">

<xs:complexType name="kontaktPodaci">

<xs:sequnce>

**<xs:group ref="osobaGrupa"/>**

<xs:element name="zemlja" type="xs:string">

<xs:element name="grad" type="xs:string">

</xs:sequence>

</xs:complexType>

</xs:element>

**2.6. Tipovi podatka**

**2.6.1. String podaci**

XML: <ime>Milos</ime>

XSD: <xs:element name="ime" type="xs:string"/>

**2.6.2. Datumi i belezenje vremena**

1. **Date tip podatka**

Format: GGGG-MM-DD

**Napomena:** Sve komponente navedenog formata su neophodne.

XML: <datumPrijave>2015-10-20</datumPrijave>

XSD: <xs:element name="datumPrijave" type="xs:date"/>

1. **Time tip podatka**

Format: cc:mm:ss (cc - cas, mm - minut, ss - sekunde)

**Napomena:** Sve komponente navedenog formata su neophodne.

XML: <vremePojavljivanja>10:30:00</vremePojavljivanja>

XSD: <xs:element name="vremePojavljivanja" type="xs:time"/>

1. **DateTime tip podatka**

Format: GGGG-MM-DDTcc:mm:ss

**Napomena:** Sve komponente navedenog formata su neophodne.

XML: <pocetniDatum>2015-10-20T09:45:00</pocetniDatum>

XSD: <xs:element name="pocetniDatum" tupe="xs:dateTime"/>

1. **Duration tip podatka**

Format: PnYnMnDTnHnMnS

P - period

nY - broj godina, nM - broj Meseci, nD - broj dana

T - specificira da se pocinje sa vremenom, nephodno ako ce se definisati vreme

nH - broj casova, nM - broj minuta, nS - broj sekundi

XML: <preostaloVreme>P0Y2M8DT10H30M</preostaloVreme>

Znacenje: 2 meseca, 8 dana, 10 sati i 30 minuta

XSD: <xs:element name="preostaloVreme" type="xs:duration"/>

Negativno trajanje, nesto sto se vec desilo:

XML: <vremeOdPojave>-P0Y0M5DT6H19M</vremeOdPojave>

XSD: <xs:element name="vremeOdPojave" type="sx:duration">

**2.6.3. Brojevni podaci**

1. **Decimal tip podatka (realni broj)**

XML:

<porosek>4.20</prosek> ili

<porosek>+4.20</prosek> ili

<porosek>-4.20</prosek>

XSD: <xs:element name="porosek" type="xs:decimal">

**Napomena:** Maksimalni broj decimalnih cifara je 18.

1. **Integer tip podatka (ceo broj)**

XML:

<rezultat>15</rezultat> ili

<rezultat>+15</rezultat> ili

<rezultat>-15</rezultat>

XSD: <xs:element name="rezultat" type="xs:integer">

**2.7. Pimer**

**Globalni i Lokalni elementi XSD-a**

Globalni elementi su elementi koji su definisani direktno nakon "schema" elementa, Lokalni elementi su elementi ugnjezdjeni u druge elemente.

**XML dokument**

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

<porudzbina

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="**porudzbina.xsd**"

>

<infoPorudzbine>

<idPorudzbine>815</idPorudzbine>

<datumEvident>2015-16-09T21:37:09</datumEvident>

</infoPorudzbine>

<kupac>

<ime>Petar</ime>

<prezime>Petrovic</prezime>

<telefon>011323874</telefon>

<telefon>0642964862</telefon>

</kupac>

<adresa>

<drzava>Srbija</drzava>

<grad>Beograd</grad>

<ulicaBr>Dunavska 73</ulicaBr>

</adresa>

<stavka>

<naziv>Assics GEL S</naziv>

<kolicina>1</kolicina>

<cena>8200.00</cena>

<komentar>Na snizenju</komentar>

</stavka>

<stavka>

<naziv>Assics tenerka</naziv>

<kolicina>1</kolicina>

<cena>5600.00</cena>

</stavka>

<stavka>

<naziv>Gerbi carape</naziv>

<kolicina>3</kolicina>

<cena>250.00</cena>

</stavka>

</porudzbina>

XML dokument iznad cine koreni element "porudzbina". Element "porudzbina" sadrzi cetiri razlicita elementa: "infoPorudzbine", "kupac", "adresa", "stavka". Element "stavka" moze da se pojavi vise puta.

Linija xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" govori XML parseru da ovaj dokument reba validirati na osnovu neke seme.

Linija xsi:noNamespaceSchemaLocation="**porudzbina.xsd**" specificira gde se ta sema nalazi i njeno ime, u ovom slucaju XSD fajl "porudzbina.xsd" sa nalazi u istom folderu kao i XML.

**Stvaranje XSD fajla** "**porudzbina.xsd**"

Prvo se kreira definicija/zaglavlje zatim osnovni <schema> element.

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

<xs:schema xmlms:xs="http://www.w3.org/2001/XMLSchema">

. . .

</xs:schema>

1. Definisanje elementa **"porudzbina"**, pod-elementi ce biti posebno objasnjeni:

<xs:element name="porudzbina">

<xs:complexType>

<xs:sequence>

. . .

</xs:sequence>

</xs:complexType>

</xs:element>

1. Definisanje elementa **"infoPorudzbine"**:

<xs:element name="infoPorudzbina">

<xs:complexType>

<xs:sequence>

<xs:element name="idPorudzbine" type="xs:positiveInteger"/>

<xs:element name="datumEvident" type="xs:dateTime"/>

</xs:sequence>

</xs:complexType>

</xs:element>

1. Definisanje elementa **"kupac"**:

Element "telefon" moze da se pojavi maksimum pet puta, maxOccurs="5".

<xs:element name="kupac">

<xs:complexType>

<xs:sequence>

<xs:element name="ime" type="xs:string"/>

<xs:element name="prezime" type="xs:string"/>

<xs:element name="telefon" type="xs:string" maxOccurs="5"/>

</xs:sequence>

</xs:complexType>

</xs:element>

1. Definisanje elementa **"adresa"**:

<xs:element name="adresa">

<xs:complexType>

<xs:sequence>

<xs:element name="drzava" type="xs:string"/>

<xs:element name="grad" type="xs:string"/>

<xs:element name="ulicaBr" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

1. Definisanje elementa **"stavka"**:

Element "komentar" je opcioni, minOccurs="0".

<xs:element name="stavka" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element name="naziv" type="xs:string"/>

<xs:element name="kolicina" type="xs:positiveInteger"/>

<xs:element name="cena" type="xs:decimal"/>

<xs:element name="komentar" type="xs:string"

minOccurs="0"/>

</xs:sequence>

</xs:complexType>

</xs:element>

**Napomena:** Ako se koriste, deklaracije atributa uvek treba da idu na kraju.

Kompletni sadrzaj fajla **"porudzbina.xsd"**:

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

<xs:schema xmlms:xs="http://www.w3.org/2001/XMLSchema">

<!-- porudzbina -->

<xs:element name="porudzbina">

<xs:complexType>

<xs:sequence>

<!-- infoPorudzbina -->

<xs:element name="infoPorudzbina">

<xs:complexType>

<xs:sequence>

<xs:element name="idPorudzbine"

type="xs:positiveInteger"/>

<xs:element name="datumEvident" type="xs:dateTime"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- kupac -->

<xs:element name="kupac">

<xs:complexType>

<xs:sequence>

<xs:element name="ime" type="xs:string"/>

<xs:element name="prezime" type="xs:string"/>

<xs:element name="telefon" type="xs:string"

maxOccurs="5"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- adresa -->

<xs:element name="adresa">

<xs:complexType>

<xs:sequence>

<xs:element name="drzava" type="xs:string"/>

<xs:element name="grad" type="xs:string"/>

<xs:element name="ulicaBr" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- stavka -->

<xs:element name="stavka" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element name="naziv" type="xs:string"/>

<xs:element name="kolicina"

type="xs:positiveInteger"/>

<xs:element name="cena" type="xs:decimal"/>

<xs:element name="komentar" type="xs:string"

minOccurs="0"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Metod koji je koriscen pri dizajniranju XSD fajla iznad je relativno prost i prati prirodnu strukturu grananja samih elemenata. Medjutim ovaj metod ima nedostataka pri kreiranju kompleksnih struktura jer je preglednost jako losa.

Sledeci metod koristi mogucnost referenciranja sadrzaja u okviru XSD fajla putem atriduta **ref**, opsti oblik: ref="ime\_elementa". Prvo ce se definisati svi prosti elementi i atributi ako postoje a zatim i kompleksni elementi koji ce da sadrze reference na potrebne proste elemente.

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

<xs:schema xmlms:xs="http://www.w3.org/2001/XMLSchema">

<!-- definisanje prostih elemenata -->

<xs:element name="idPorudzbine" type="xs:positiveInteger"/>

<xs:element name="datumEvident" type="xs:dateTime"/>

<xs:element name="ime" type="xs:string"/>

<xs:element name="prezime" type="xs:string"/>

<xs:element name="telefon" type="xs:string" maxOccurs="5"/>

<xs:element name="drzava" type="xs:string"/>

<xs:element name="grad" type="xs:string"/>

<xs:element name="ulicaBr" type="xs:string"/>

<xs:element name="naziv" type="xs:string"/>

<xs:element name="kolicina" type="xs:positiveInteger"/>

<xs:element name="cena" type="xs:decimal"/>

<xs:element name="komentar" type="xs:string" minOccurs="0"/>

<!--definisanje prostih elemenata -->

<!-- infoPorudzbina -->

<xs:element name="infoPorudzbina">

<xs:complexType>

<xs:sequence>

<xs:element ref="idPorudzbine"/>

<xs:element ref="datumEvident"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- kupac -->

<xs:element name="kupac">

<xs:complexType>

<xs:sequence>

<xs:element ref="ime"/>

<xs:element ref="prezime"/>

<xs:element ref="telefon"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- adresa -->

<xs:element name="adresa">

<xs:complexType>

<xs:sequence>

<xs:element ref="drzava"/>

<xs:element ref="grad"/>

<xs:element ref="ulicaBr"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- stavka -->

<xs:element name="stavka">

<xs:complexType>

<xs:sequence>

<xs:element ref="naziv"/>

<xs:element ref="kolicina"/>

<xs:element ref="cena"/>

<xs:element ref="komentar"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- porudzbina -->

<xs:element name="porudzbina">

<xs:complexType>

<xs:sequence>

<xs:element ref="infoPorudzbina"/>

<xs:element ref="kupac"/>

<xs:element ref="adresa"/>

<xs:element ref="stavka" maxOccurs="unbounded"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Koriscenje CSS fajla radi poboljsanja prikaza XML podataka:

porudzbina {border:2px solid black;

border-radius:8px;

width:350px;

padding:5px;

margin:10px;}

infoPorudzbine {border:1px solid black;

border-radius:8px;

display:block;

padding:5px;

margin-bottom:5px;}

idPorudzbine {display:block; font-weight:bold;}

datumEvident {display:block;}

kupac {border:1px solid black;

border-radius:8px;

display:block;

font-weight:bold;

padding:5px;

margin-bottom:5px;}

telefon {display:block;}

adresa {border:1px solid black;

border-radius:8px;

display:block;

font-weight:bold;

padding:5px;

margin-bottom:5px;}

stavka {border:1px solid black;

border-radius:8px;

display:block;

padding:5px;

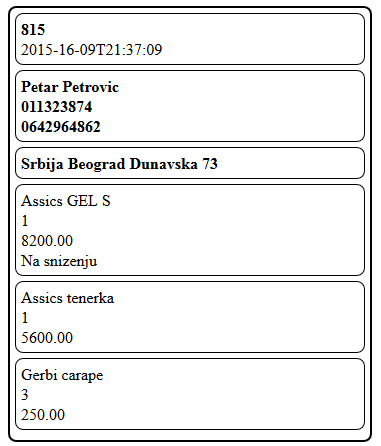
margin-bottom:5px;}

naziv {display:block;}

kolicina {display:block;}

cena {display:block;}

komentar {display:block;}



**3. XSLT**

**3.1. Uvod u XSLT (XLS)**

**XSL** (E**X**tensible **S**tylesheet **L**anguage) i predstavlja specijalan jezik za stilizovanje odnosno formatiranje izgleda XML podataka. XSL je razvijen jer je postojala potreba za jezikom za formatiranje i stilizovanje u sklopu samog XML. Upotreba CSS-a omogucava samo ogranicene rezultate i prvenstveno je namenjena za HTML

Bitni sadrzaji XLS-a su:

* **XSLT** - jezik za transformaciju XML dokumenata.
* **XPath** - jezik za navigaciju kroz XML dokumente.
* **XQuery** - jezik za kreiranje upita nad XML dokumentima.

**XSLT** (E**X**tensible **S**tylesheet **L**anguage **T**ransformations) i predstavlja glavni deo XSL-a. XSLT sluzi za konvertovanje XML dokumenata u XHTML dokumente ili neki drugi XML dokument.

Konvertovanjem XML dokumenata u XHTML format internet pretrazivaci, broser-i, mogu da tumace sadrzaj XML dokumenta. Svi poznati internet pretrazivaci podrzavaju XSLT.

**3.2. Konvertovanje XML-a u XHTML**

**Definisanje XML dokumenta "primer\_8.xml"**:

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

<katalog>

<proizvod>

<idProizvoda>1</idProizvoda>

<naziv>Assics GEL 5</naziv>

<cena>8600.00</cena>

<proizvodjac>Assics</proizvodjac>

<sezona>2015</sezona>

<grupa>muskarci</grupa>

<kategorija>patike</kategorija>

</proizvod>

<proizvod>

<idProizvoda>2</idProizvoda>

<naziv>Assics RUN S</naziv>

<cena>10200.00</cena>

<proizvodjac>Assics</proizvodjac>

<sezona>2015</sezona>

<grupa>muskarci</grupa>

<kategorija>patike</tip>

</proizvod>

<proizvod>

<idProizvoda>3</idProizvoda>

<naziv>Women Sports</naziv>

<cena>7200.00</cena>

<proizvodjac>Nike</proizvodjac>

<sezona>2015</sezona>

<grupa>zene</grupa>

<kategorija>trenerka</kategorija>

</proizvod>

<proizvod>

<idProizvoda>4</idProizvoda>

<naziv>XSport run 21</naziv>

<cena>7500.00</cena>

<proizvodjac>Addidas</proizvodjac>

<sezona>2015</sezona>

<grupa>deca</grupa>

<kategorija>patike</kategorija>

</proizvod>

</katalog>

**Definisanje XSLT dokumenta "primer\_8\_XSLT.xsl":**

Kao i kod obicnog XML dokument pocinje se sa definicijom:

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

Zatim se navodi koreni (glavni) element koji moze da bude, ili <xs:stylesheet> ili <xs:transform>, sasvim je sve jedno da li se koristi <xs:stylesheet> ili <xs:transform>, primer:

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

>

. . .

</xs:stylesheet>

Da bi se dobio pristup XSLT elementima, atributima i sl. mora se definisati namesapce na pocetku dokumenta. Ako se koristi xmlns:xsl="http://www.w3.org/1999/XSL/Transform" mora se ubaciti i atribut version="1.0".

**Povezivanje XML dokumenta sa odgovarajucim XSLT fajlom**, opsti oblik:

<?xml-stylesheet type="text/xsl" href="putanja\_do\_xsl\_fajla"?>

Unos linka u XML dokument "primer\_8.xml":

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

<?xml-stylesheet type="text/xsl" href="**primer\_8\_XSLT.xsl**"?>

<katalog>

<proizvod>

<idProizvoda>1</idProizvoda>

<naziv>Assics GEL 5</naziv>

.

.

.

**XSLT dokument "primer\_8\_XSLT.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>

<style>

.div {border:2px solid red;

border-radius:7px;

width:600px;

padding:5px;

margin:10px;}

.tabelaProizvodi {border: 2px solid darkgray;

border-collapse:collapse;

width:100%;}

.tabelaProizvodi th, tr, td {border: 1px solid darkgray;}

.tabelaProizvodi th {background-color:linen;}

</style>

<body>

<div class="div">

<table class="tabelaProizvodi">

<caption><b>Katalog proizvoda</b></caption>

<tr>

<th style="width:100px;">ID Proizvoda</th>

<th>Naziv</th>

<th style="width:80px;">Cena</th>

<th>Proizvodjac</th>

<th>Kategorija</th>

</tr>

<xsl:for-each select="katalog/proizvod">

<tr>

<td style="text-align:center;"><xsl:value-of

select="idProizvoda"/></td>

<td><xsl:value-of select="naziv"/></td>

<td style="text-align:center;"><xsl:value-of

select="cena"/></td>

<td><xsl:value-of select="proizvodjac"/></td>

<td><xsl:value-of select="kategorija"/></td>

</tr>

</xsl:for-each>

</table>

</div>

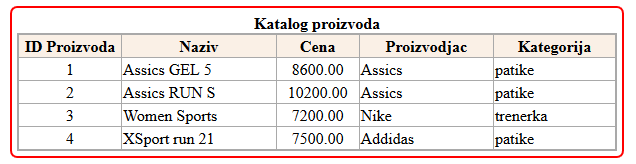
</body>

</html>

</xsl:template>

</xsl:stylesheet>

Rezultat:



Ispis podatka sacuvanih u XML dokumentu "primer\_8.xml".

**3.2.1. Template element, <xsl:template>**

XSL dokument se sastoji od jednog ili vise pravila koja se zovu teplates-i. Templejti sadrzi pravila koja se pimenjuju kada se nadje trazeni sadrzaj.

Element **<xs:template>** se koristi da bi se kreirali templatejti. Atribut **match** se koristi da bi se povezao konkretni templejt sa XML dokumentom. Vrednost match atributa je XPath izraz, za ceo dokumnt postavlja se **match="/"**. Unutar samog templejt elementa ce se definisati HTML kod za ispis na ekranu.

XML dokument, **"primer\_9.xml"**:

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

<?xml-stylesheet type="text/xsl" href="**primer\_9\_XSLT.xsl**"?>

<listaOsoba>

<osoba>

<ime>Jovan</ime>

<prezime>Jovanovic</prezime>

<godiste>1986</godiste>

</osoba>

<osoba>

<ime>Marko</ime>

<prezime>Markovic</prezime>

<godiste>1984</godiste>

</osoba>

<osoba>

<ime>Petar</ime>

<prezime>Petrovic</prezime>

<godiste>1992</godiste>

</osoba>

<osoba>

<ime>Stefan</ime>

<prezime>Stefanovic</prezime>

<godiste>1995</godiste>

</osoba>

<osoba>

<ime>Ivan</ime>

<prezime>Ivanovic</prezime>

<godiste>1989</godiste>

</osoba>

<osoba>

<ime>Obi Van</ime>

<prezime>Kenobi</prezime>

<godiste>1990</godiste>

</osoba>

<osoba>

<ime>Mace</ime>

<prezime>Vindu</prezime>

<godiste>1985</godiste>

</osoba>

</listaOsoba>

XSLT dokument **"primer\_9\_XSLT.xml"**:

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

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<!-- pocetak templejta -->

<xsl:template match="/">

<html>

<head>

<style>

.tableaOsobe {border:3px ridge lightgreen;

border-collapse:collapse;

margin:10px;}

.tableaOsobe th {background-color:aqua;

padding:3px;}

.tableaOsobe tr, th, td {border:1px solid gray;}

.tableaOsobe td {padding:3px;}

</style>

</head>

<body>

<table class="tableaOsobe">

<caption><b>Lista Osoba</b></caption>

<tr>

<th>Ime</th>

<th>Prezime</th>

<th>Godiste</th>

</tr>

<!-- dinamicki ispis sadrzaja -->

<tr>

<td>. . .</td>

<td>. . .</td>

<td>. . .</td>

</tr>

</table>

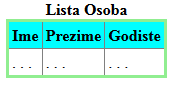
</body>

</html>

</xsl:template>

</xsl:stylesheet>

Rezultat:



Trenutni sadrzaj xsl dokumenta samo postavlja potrebni templejt, za ispis sadrzaja is XML dokumnta potrebni su value-of element za pristup sadrzaju XML elementa i for-each petlja.

**3.2.2. Value-of element, <xsl:value-of>**

Element **<xsl:value-of>** se koristi za pristup sadrzaju specificiranih elemenata XML dokumenta kako bi se se on ispisao u sklopu transforamacije, opsti oblik:

<xsl:value-of select="ime\_element/ime\_pod-elementa/..."/>

Primer ispisa prvod imena i prezimena iz XML dokumenta "primer\_9.xml":

<tr>

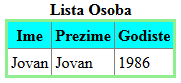
<td><xsl:value-of select="listaOsoba/osoba/ime"/></td>

<td><xsl:value-of select="listaOsoba/osoba/prezime"/></td>

<td><xsl:value-of select="listaOsoba/osoba/godiste"/></td>

</tr>

Rezultat:



Dodatka value-of elementa je omogucio da se ispisu samo podaci iz prvog elementa XML dokumnta, da bi se ispisali svi podaci mora se dodati for-each petlja.

**3.2.3. For-each element/pelja, <xsl:for-each>**

Primer ispisa podatka svih osoba iz liste, obratiti paznju na **select** atribut for-each elementa.

<xsl:for-each **select="listaOsoba/osoba"**>

<tr>

<td><xsl:value-of select="ime"/></td>

<td><xsl:value-of select="prezime"/></td>

<td><xsl:value-of select="godiste"/></td>

</tr>

</xsl:for-each>

Rezultat:



**Filtriranje sadrzaja preko for-each elementa**, opsti oblik:

select="ime\_elementa/ime\_pod-elementa/...[ime\_elementa operator ; 'vrenost']"

Dozvoljeni operatori za definisanje uslova filtriranja:

* **=**  - jednako
* **!=** - nije jednako
* **&lt;** - manje od
* **&gt;** - vece od

Primer: Obratiti paznju na select atribut i sintaksu postavljenog uslova da je pod-element **godiste** iz elementa **osoba** veci od (&gt) vrednosti 1988.

<xsl:for-each **select="listaOsoba/osoba[godiste&gt;'1988']"**>

<tr>

<td><xsl:value-of select="ime"/></td>

<td><xsl:value-of select="prezime"/></td>

<td><xsl:value-of select="godiste"/></td>

</tr>

</xsl:for-each>

Rezultat:



Dozvoljeni operatori za definisanje uslova filtriranja:

* **=**  - jednako
* **!=** - nije jednako
* **&lt;** - manje od
* **&gt;** - vece od

**3.2.4. Sortiranje sadrzaja, <xsl:sort> element**

Opsti oblik: <xsl:sort select="element"/>

**Napomena:** "element", predstavlja neki od elemenata na osnovu cijih vrednosti ce se izvrsiti sortiranje.

Primer:

<xsl:for-each select="listaOsoba/osoba">

<xsl:sort select="ime"/>

<tr>

<td><xsl:value-of select="ime"/></td>

<td><xsl:value-of select="prezime"/></td>

<td><xsl:value-of select="godiste"/></td>

</tr>

</xsl:for-each>

Rezultat:



**3.2.5. If uslov, <xsl:if> element**

Opsti oblik:

<xsl:if test="ime\_elementa operator; vrednost">

. . . neki ispis u slucaju da je uslov ispunjen . . .

</xsl:if>

Opsti primeri:

<xsl:if test="broj &lt; 15"> - vrednost elementa "broj" veca manja od 15.

<xsl:if test="cena =; 6500"> - vrednost elemnenta "cena" jednaka 6500.

<xsl:if test="idProizvoda &gt; 183"> - vrednost, "idProizvoda", veca od 183.

Primer: Obratiti paznju da element <xsl:sort>, ako se koristi, ide pre <xsl:if>.

<xsl:for-each select="listaOsoba/osoba">

<xsl:sort select="ime"/>

<xsl:if test="godiste &gt; 1986">

<tr>

<td><xsl:value-of select="ime"/></td>

<td><xsl:value-of select="prezime"/></td>

<td><xsl:value-of select="godiste"/></td>

</tr>

</xsl:if>

</xsl:for-each>

Rezultat:



Primer koriscenja kombinacije operatora, vece od: **&gt;=**.

<xsl:for-each select="listaOsoba/osoba">

<xsl:sort select="ime"/>

<xsl:if test="godiste **&gt;=** 1986">

<tr>

<td><xsl:value-of select="ime"/></td>

<td><xsl:value-of select="prezime"/></td>

<td><xsl:value-of select="godiste"/></td>

</tr>

</xsl:if>

</xsl:for-each>

Rezultat:

