M. Essert: WEB PROGRAMIRANJE
3. predavanje

3. XML

3.1. Standardizacija Interneta

Engleska riječ 'site' predstavlja kraticu za mjesto gdje se pohranjuje adekvatno pripravljena informacija (web stranice) za javni (Internet) ili ograničeni, lokalni (Intranet) pristup. Ako se stranice u fazi testiranja objavljuju na računalu koje nije spojeno na mrežu, ali ima server, onda se takvo mjesto zove *local site*, za razliku od *web site*-a ili remote (udaljenog) site-a.

Protokol za razmjenu dokumenata između računala u mreži razvijen je 80-tih godina u CERN-u (*Conseil Europèene pour la Recherche Nuclèaire*), europskoj organizaciji za nuklearni razvitak. Taj TCP/IP protokol, postat će poznat kao HTTP (*Hypertext Transfer Protocol*) protokol, jer će prenositi dokumente kodirane u HTML (*HyperText Markup Language*) jeziku između računala povezanih u jedinstvenu svjetsku mrežu zvanu Internet. Među zaslužnim pojedinincima za njegov razvoj spominju se Doug Engelbart - 1960: "*NLS - oNLine System*" s preglednikom i e-mailom, Ted Nelson - 1965: uvodi pojam "hypertext"-a za prijenos informacije među računalima, te Berners-Lee - 1980: "*Enquire-Within-Upon-Everything*" s koncepcijom svjetskog WEB-a i univerzalnog adresiranja dokumenata preko URL-a (*Universal Resource Locator)*.

Ideja jezika za označavanje podataka je da se korisni sadržaj uokviri odgovarajućim oznakama. Oznake bi trebale biti jednostavno čitljive i razumljive i čovjeku koji ih gleda u bilo kojem programu za uređivanje teksta, a i računalni programi koji parsiraju taj sadržaj trebali bi moći na jednostavan način izvaditi određene podatke.

3.1.1. Generalized Markup Language (GML)

60ih godina 20. stoljeća IBM je imao velik problem s ogromnom količinom različite tehničke dokumentacije koja se za svaku posebnu namjenu morala prepisivati i nanovo uređivati za što su trošili ogromnu količinu ljudskih resursa. Ideja do koje su došli bila je prvi šire korišteni jezik za označavanje podataka. Korisni sadržaj uokvirio bi se određenim oznakama koje ce ga opisivati. Nakon što se to učini, za određene namjene jednostavno će se povlačiti sadržaji određenog tipa. GML oznake opisivale su određene dijelove dokumenta kao npr. poglavlja, važna poglavlja, manje važna poglavlja, liste, tablice itd. Korištenjem GMLa iz istog sadržaja mogla se dobiti ispisanu različita dokumentacija, tehnička, ali i korisnička.

3.1.2. Standard Generalized Markup Language (SGML)

GML se pokazao kao uspješan proizvod tako da je razvoj u tom smjeru nastavljen. 80ih godina 20. stoljeća American National Standards Institute (ANSI) radio je na razvoju standarda jezika za označavanje podataka. Zahtjevi kojima su autori pokušali zadovoljiti bili su da nastali proizvod bude dovoljno formaliziran da može jamčiti vjernost dokumenta izvorniku, dovoljno strukturiran da se može nositi s kompleksnim dokumentima i dovoljno otvoren da može podržati rukovanje velikim količinama podataka. Nastali jezik nazvan je Standard Generalized Markup Language (SGML) i 1986. objavljen je kao međunarodna norma ISO 8879. Problem SGMLa, kao i mnogih sličnih proizvoda koji su razvijani na takav način, od strane organizacija za standardizaciju je što je bio golem. Autori su pokušali pokriti svaku moguću primjenu jezika i nastali proizvod je bio opširan, složen za korištenje te zbog toga skup u upotrebi. Zbog tih osobina SGML nije bio jako raširen u praksi i korisnici SGMLa bile su uglavnom velike kompanije, državne službe i znanstvene institucije.

3.1.3. HyperText Markup Language (HTML)

HTML je nastao kada je Tim Berneres Lee izabrao jedan mali skup oznaka iz SGML skupa koji je korišten na CERNu i primjenio ih na formatiranje dokumenata. HTML je imao mali skup oznaka koje su opisivale osnovne dijelove dokumenta. Programi koji su tumačili strukturu takvih dokumenata bili su HTML preglednici.

3.1.4. Extensible Markup Language (XML)

Problem HTMLa je što ima mali skup zadanih oznaka. Kada se želi proširiti s novim oznakama mora se mijenjati standard što ga čini nepraktičnim. Osim toga, iako je HTML izvorno zamišljen kao jezik za opisivanje sadržaja, zbog potreba i želja tržišta te razvoja pregledničkih tehnologija (naročito za vremena "pregledničkog rata" između Microsofta i Netscapea proširivan je nestandardnim oznakama koje su prvenstveno služile za formatiranje sadržaja u smislu njegovog prikaza u internet pregledniku. Za opisivanje sadržaja SGML je bolji izbor od HTMLa, ali ima veliki nedostatak što je preglomazan za korištenje i izvršavanje unutar internet preglednika. Zbog toga je trebalo kreirati jezik koji će s jedne strane biti dovoljno malen i jednostavan da se može izvršavati unutar internet preglednika, a s druge strane dovoljno prilagodljiv da se može proširivati korisničkim oznakama. Posla izrade specifikacije takvog jezika prihvatio se početkom 90ih godina 20. stoljeća World Wide Web Consortium. Željeli su razviti jezik koji će objediniti jednostavnost HTMLa i izražajnu snagu SGMLa. Na početku su odredili su 10 ciljeva kojih su se u razvoju trudili pridržavati:

1. XML mora biti izravno primjenjiv preko interneta.
2. XML mora podržavati širok spektar primjena.
3. XML mora biti kompatibilan s SGML-om.
4. Mora biti lako pisati programe koji procesiraju (parsiraju) XML dokumente.
5. Broj slobodnih značajki u XML-u mora biti apsolutno minimalan, u idealnom slučaju jednak nuli.
6. XML dokumenti moraju biti čitljivi ljudima, te u razumnoj mjeri jednostavni
7. Standard mora biti specificiran što prije
8. Dizajn XML-a mora biti formalan i precizan
9. Kreiranje XML dokumenata mora biti jednostavno
10. Sažetost kod označavanja dokumenta XML-om je od minimalnog značaja.

World Wide Web Consortium je 10. veljače 1998. objavio prvu verziju XML preporuke. Danas je XML jezik vrlo raširen i koristi se za različite namjene: odvajanje podataka od prezentacije, razmjenu podataka, pohranu podataka, povećavanje dostupnosti podataka i izradu novih specijaliziranih jezika za označavanje. XML je standardizirani jezik i za njegovu standardizaciju brine se World Wide Web Consortium.

3.2. Elementi i atributi

XML dokument sastoji se od jednog ili više elemenata. Element se sastoji od niti jednog, jednog ili više znakova informacije omeđenog(-ih) s dvije oznake, engleski *tag*-a. Tag se sastoji od imena tag-a omeđenog znakovima manje '<' i više '>'. Ime tag-a mora početi slovom, podvučenom crticom '_' ili dvotočkom ':', nakon čega može slijediti jedan više slova, znamenki i posebnih znakova (povlake '-', podvlake '_', točke '.' ili dvotočke ':'). Napomena: jednostruki navodnici (')ovdje služe samo zato da se znak odvoji od opisa, te ne čine dio znaka. U imenu tag-a ne smije se pojavljivati niz 'xml' u bilo kojem rasporedu velikih ili malih slova.

Početni i završni (dočetni) tag XML elementa imaju (identički) jednako ime, s tim što završni tag ima desnu kosu crtu '/' ispred svog imena. Ako element nema niti jednog znaka informacije, onda se dva tag-a mogu sažeti u jedan, koji ispred svoje zadnje međe '>' ima desnu kosu crtu '/'. Element može imati jedan ili više atributa koji se dopisuju nakon imena prvog tag-a, a služe za dodatni opis informacije

elementa. Atribut ima ime kojem je znakom jednakosti '=' pridružena neka vrijednost omeđena dvostrukim navodnicima ' " '. Unutar tag-a mogu se slobodno koristiti prazna mjesta (razmaci , tabulatori ili novi redovi), pa se obično tag koji ima više atributa zbog bolje preglednosti piše u nekoliko redaka.

Korištenje posebnih znakova i onih koji se koriste kao međe tag-ova unutar informacije koju tag-ovi omeđuju postiže se uporabom znaka '&' na početku i znaka ';' na kraju opisne kratice. Tako na primjer znak '<' u XML informaciji pišemo kao '<' (dakako, bez jednostrukih navodnika), dok za '>' imamo ekvivlanetno '>', za jednostruki navodnik (') pišemo ''', dvostruke navodnike zamjenjujemo sa '&quote;', a znak '&' sa '&'. Posebni znakovi umeću se u XML informaciju preko njihovog heksadecimalnog ekvivalenta kojem prethode znakovi '#x', npr. umjesto znaka © koji označuje 'Copyright' pišemo '&#xA9', gdje je heksadecimalni broj 'A9' ekvivalent ASCII znaka ©.

Primjer 3.1

Primjer XML elementa	Opis
<naslov> Uvod u XML </naslov>	Ispravan XML element, s tagom imena 'naslov' i informacijom 'Uvod u XML'
<nema_informacije></nema_informacije> ili <nema_informacije/>	Ispravan XML element bez informacije
<cijena tečaj="kn"> 102.45 </cijena>	Ispravan XML element s atributom 'tečaj' i pridružene vrijednosti 'kn'
<poruka broj="10" datum="2002-06-30" od="Đuro Sudeta" > Kućice u cvijeću , © by Zipm </poruka>	Ispravan XML element s više atributa formatiranih zbog bolje preglednosti; u informaciji poseban znak ©
<adresa> Ilica 512	Neispravan XML element: nedostaje dočetni tag
<moja boja> Ljubičasta </moja boja>	Neispravan XML element: ime tag-a ne smije imati razmak
<ime> Petar Hektorović </Ime>	Neispravan XML element: XML je osjetljiv na velika i mala slova u imenu tagova
<primjer_xmL-a slova="mala-i-velika" brojke=da> Sintaktički pogrešan primjer 123 <primjer_xmL-a>	Neispravan XML element: u imenu tag-a ne smije biti niz 'xml', vrijednost atributa treba biti unutar dvostrukih navodnika, dočetni tag mora ispred imena imati '/'

XML elementi se mogu gnijezditi (engl. *nested*), što znači da unutar jednog XML elementa može postojati jedan ili više drugih XML elemenata, od kojih svaki može sadržavati nove elemente. Na taj način XML dokument predstavlja stablenu (engl. *tree*) strukturu elemenata. Temeljni element u kojem se nalaze svi ostali zove se korjenski (engl. *root*) element. U takvoj strukturi XML elementi se često zovu i čvorovi (engl. *nodes*), a njihov međusobni odnos izražava se pojmovima iz obiteljskog stabla: roditelj (engl. *parent*), dijete (engl. *children*), unuci (engl. *grandchildren*) i sl. ili se koriste pojmovi prethodnika (engl. *ancestor*) ili sljedbenika (engl. *descendant*).

Prilikom gnježdenja elemenata treba paziti da se tag-ovi elemenata ne smiju preklapati: početni tag zadnje gnježdenog elementa mora se zatvoriti s pripadajućim dočetnim tagom istog elementa prije nego se zatvori dočetni tag elementa prethodnika (LIFO struktura: *last in - first out*).

3.3. Namespace – prostor imena

Budući da se više XML dokumenata može međusobno povezati u jedinstveni XML dokument, bilo je potrebno otkloniti mogućnost podudaranja imena tag-ova koji u različitim dokumentima mogu imati različit smisao, atribute i čvorno mjesto. Stoga je uveden pojam prostora imena (engl. *namespace*), koji se kao prefiks odvojen dvotočkom dodaje početnom imenu tag-a. Tako će se tag-ovi <Fsb:knjiga> i <FER:knjiga> razlikovati - ime 'knjiga' dobit će dva nova prostora. U dočetnom tag-u ne navodi se prostor imena.

XML dokument počinje naredbom <?xml ... ?> koji može imati nekoliko argumenata. XML dokument može imati i komentar koji ima oblik posebnog tag-a: <!-- ovo je komentar --> i koji u svojoj relevantnoj informaciji zbog očevidnog razloga ne smije imati dvije crtice '--' u nizu. Komentar se ne smije gnijezditi unutar tag-ova elemenata, a ne smije doći ni ispred početne XML naredbe.

Primjer 3.2.

Primjer gnježdenih XML elemenata	Opis
<knjiga> <naslov> Uvod u XML </naslov> <izdavač> FSB Zagreb </izdavač> </knjiga>	Ispravan XML element, s tagom imena 'knjiga' u koji su uglavljeni (ugnježdeni) tag-ovi 'naslov' i 'izdavač' s pripadanom informacijom
<?xml version="1.0"> <FSB:knjiga ISBN="1-12345-678-9" > <FSB:naslov> Uvod u XML </naslov> <FSB:Slika src="logo.gif"/> <FSB:izdavač godina="2002"> FSB Zagreb </izdavač> </knjiga>	Ispravan XML element s početnom naredbom i gnježdenim elementima s prostorom imena. Neki tagovi ('knjiga', 'izdavač', 'Slika') imaju atribute, s tim da je jedan ('Slika') prazan (nema eksplicitnu informaciju).
<!-- ovo je preuranjeni komentar --> <?xml version="1.0"> ovo je bold italic tekst 	Neispravno ispreplitanje HTML tagova, koji su također i XML tagovi. Pogrešan je i komentar, koji je prije XML naredbe.

Da se isti prefiks prostora imena ne bi morao prečesto pisati, ponavljati u tag-ovima elemenata, postoji način njegovog zadavanja i to preko atributa 'xmlns:' (kratica od *XML NameSpace*) kojemu se uz prikladno ime pridužuje najčešće URL adresa s kojeg potječe dokument. Na taj način se postiže najsigurnija jednoznačnost elemenata XML dokumenta. Dopušteno je definirati više od jednog prostora imena, pa neka imena tag-ova mogu imati jedan (pretpostavljeni, engl. *default*), a druga drugi prostor imena. Konačno, ako ne želimo da se pretpostavljena vrijednost prostora imena koristi za neki(-e) element(-e), onda atributu 'xmnls' pridružimo prazan niz znakova.

Primjer 3.3.

Primjer prostora imena	Opis
<knjige xmlns='http://zrno.fsb.hr/' xmlns:FSB='http://www.fsb.hr/'> <knjiga> <naslov> Uvod u XML </naslov> <ISBN>1-12345-678-9</ISBN> </knjiga> <FSB:izdavač> FSB Zagreb </FSB:izdavač> </knjige>	Ispravan XML dokument, s dva prostora imena. Pretpostavljeni se odnosi na elemente 'knjiga', 'naslov' i 'ISBN', dok se prostor imena 'FSB' primjenjuje na element 'izdavač'.

<stavka xmlns=' ' xmlns:COMA='http://zrno.fsb.hr/' xmlns:FSB='http://www.fsb.hr/' > <naslov> Uvod u XML </naslov> <Slika src="logo.gif"/> <cijena> 505.- </cijena> </stavka>	Ispravan XML dokument s više prostora imena. Budući da je pretpostavljeni prostor imena prazan, tag-ovi 'naslov', 'slika' i 'cijena' nemaju pretpostavljeni (niti ikakav) prostor imena.

3.4. Provjera ispravnosti XML dokumenta

Da bi se XML dokumenti mogli programski obrađivati potrebno je da budu valjani (engl. *valid*). Ta se valjanost ne odnosi na sintaktičku ispravnost (ispunjavanje zadanih XML pravila) koja je do sada razmatrana (engl. *well-formed*), već na logičku povezanost elemenata, atributa i informacije koju opisuju. Logička valjanost definira se novim pravilima koja su zapisanu u posebnoj datoteci ili su unutar samog dokumenta. Postoje dva pristupa istom problemu: preko DTD (Document Type Definition) pravila ili noviji preko XML shema (XDR - XML Data Reduced ili XSD - XML Schema Definition).

Shema određuje koje elemente smije sadržavati XML dokument te redoslijed i broj tih elemenata. Ispravnost provjerava se u odnosu na određenu Document Type Definition (DTD) ili XML schemu. Dokument sheme navodi se u zaglavlju XML dokumenta.

Povezivanje XML dokumenta s DTD shemom:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE poruka SYSTEM "PorukaPodsjetnik.dtd">
<poruka>
        <zaKoga>Pero</zaKoga>
        <odKoga>Kate</odKoga>
        <naslov>Podsjetnik</naslov>
        <tijelo>Otiđi kupiti kruh</tijelo>
</poruka>
```

Povezivanje XML dokumenta s XML shemom:

```
<?xml version="1.0" encoding="UTF-8"?>
<poruka xmlns="http://hr.wikipedia.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="poruka.xsd">
<poruka>
        <zaKoga>Pero</zaKoga>
        <odKoga>Kate</odKoga>
        <naslov>Podsjetnik</naslov>
        <tijelo>Otiđi kupiti kruh</tijelo>
</poruka>
```

3.5. XML i HTML

XML i HTML sintaksno su slični, iako su razvijeni s različitim namjenama. XML je prvenstveno razvijen za opisivanje podataka. XML ne radi ništa osim što opisuje podatke. Oznake kod XMLa su slobodne i korisnici ih moraju sami smisliti i kreirati. U HTMLu postoji predefinirani skup oznaka koje uglavnom služe za prikazivanje sadržaja u internet pregledniku na odgovarajući način. Sintaksna pravila XMLa vrlo su stroga i ako dokument nije formatiran u skladu s njima, računalni program neće moći pročitati XML dokument. S druge strane HTML dokument koji je sintaksno neispravan većina internet preglednika će uredno pročitati i pokušati protumačiti na najbolji mogući način iz ispravnih informacija. EXtensible HyperText Markup Language (XHTML) je novi standard razvoja HTML jezika u skladu sa strožim pravilima XMLa. Namjena tog jezika je da s vremenom u potpunosti zamjeni stari HTML standard.

3.5.1. Prikaz XMLa na web stranicama

XML je jezik namijenjen opisu podataka i u sebi ne nosi informacije o tome na koji način se određeni podaci trebaju prikazati. To se može napraviti pomoću odgovarajućih atributa, ali u tom slučaju potreban je poseban računalni program koji će na odgovarajući način protumačiti te atribute. Kada se XML dokument gleda u internet pregledniku prikazuje se kao običan tekstualni dokument. Neki internet preglednici prikazuju stablo na kojem se klikom na odgovarajući čvor njemu podređene grane mogu prikazati ili sakriti od prikazivanja. Želimo li prikazati sadržaj u odgovarajućem formatu, to možemo napraviti na nekoliko načina:

- pozvati XML sadržaj unutar HTML dokumenta. U Internet Explorer internet pregledniku pomoću neslužbene <xml></xml> oznake moguće je unutar HTML dokumenta umetnutu sadržaj XML dokumenta (tzv. podatčani otoci (data islands). U drugim internet preglednicima moguće je učitati sadržaj pomoću JavaScript programa.
- u XML dokumentu navesti naziv CSS datoteke koja ga formatira na odgovarajući način. Povezivanje se radi unutar zaglavlja XML dokumenta.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="poruka.css"?>
<poruka>
        <zaKoga>Pero</zaKoga>
        <odKoga>Kate</odKoga>
        <naslov>Podsjetnik</naslov>
        <tijelo>Otiđi kupiti kruh</tijelo>
</poruka>
```

- u XML dokumentu navesti naziv XSLT datoteke koja ga formatira na odgovarajući način

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="poruka.xslt"?>
<poruka>
        <zaKoga>Pero</zaKoga>
        <odKoga>Kate</odKoga>
        <naslov>Podsjetnik</naslov>
        <tijelo>Otiđi kupiti kruh</tijelo>
</poruka>
```

3.6. Korištenja XMLa

3.6.1. Odvajanje podataka od prezentacije

Unutar HTML dokumenta sadržaj se može odvojiti od prezentacije. Na takav način web dizajneri imaju veću slobodu u formatiranju i promjenama izgleda web stranica. Kada se promjeni izgled web stranice, ne mora se mijenjati njezin sadržaj. Sadržaj bi se čuvao unutar XML dokumenta, a HTML dokument služio bi za prezentaciju toga sadržaja.

3.6.2. Razmjena podataka

XML dokument je obična tekstovna datoteka. Njezin sadržaj je vidljiv i pristupačan čovjeku, bez ikakvog posebnog programa kao i svakom računalnom programu koji može čitati tekstovne datoteke. Razmjena između sustava na različitim platformama najčešće ide na način da jedan sustav ispiše svoje podatke u dogovorenom formatu u tekstovnu datoteku, a drugi sustav ih pročita. Format tih datoteka obično je bio pozicijski (svi podaci u jednom retku, a svaki podatak ima određenu duljinu) ili delimiterski (podaci razdvojeni posebnim znakom – razdjeljnikom, delimiterom npr. zarez ili točka-zarez). Problem kod toga je što je za svaku posebnu razmjenu potrebno raditi novi format prema dogovoru strana u razmjeni. osim toga ti formati su neintuitivni. Kada pogledate tekstovnu datoteku u pozicijskom formatu bez dokumentacije toga formata nećete moći niti naslutiti kakvi podaci se u njoj nalaze. XML omogućuje da svaka strana opiše podatke koje ima, odnosno one koje im trebaju odgovarajućim oznakama koje su čitljive i čovjeku i koje dosta dobro opisuju podatak koji se unutar određene oznake nalazi.

3.6.3. Pohrana podataka

XML dokument opisuje sadržaj i kao takav može se koristiti za njegovu pohranu. Najjednostavniji način je izravna pohrana u tekstualne datoteke koje se spremaju na disk. Podaci iz datoteka mogu se čitati izravno, korištenjem običnog programa za uređivanje teksta ili korištenjem nekog od posebnih programa za rad s XML podacima. Nešto napredniji način pohrane je da se XML elementi pohranjuju u relacijsku bazu podataka. Nedostatak tog pristupa je da relacijska struktura baze ne odgovara hijerarhijskom pristupu XML dokumenata. Zbog toga su se počele kreirati posebne XML baze podataka koje su svojim načinom rada posebno prilagođene pohrani i radu s XML podacima.

3.6.4. Povećavanje dostupnosti podataka

XML podatke pohranjuje u običnu tekstovnu datoteku koja je čitljiva gotovo na svakoj platformi. Format je jasno vidljiv i jednostavno je napraviti računalni program koji će na odgovarajući način interpretirati podatke. Postoji nekoliko inicijativa s idejom da se za pohranu podataka umjesto zatvorenih binarnih formata (npr. doc, xls,...) za pohranu podataka koristi XML format. Danas sve više računalnih programa koristi taj format za pohranu podataka.

3.6.5. Izradu novih specijaliziranih jezika za označavanje

- XML je jezik ali i platforma koja služi za stvaranje novih, specijaliziranih jezika za označavanje. Neki od jezika koji su kreirani iz XMLa
- Wireless Markup Language (WML) je jezik koji se koristi u WAP mrežama za pristup internetu s prijenosnih uređaja (mobitela, ručnih računala,...)
- MathML je jezik koji omogućuje jednostavno kodiranje matematičkih izraza s ciljem jednostavne razmjene, prikazivanje i korištenje na web stranicama
- User Interface Markup Language (UIML) je jezik namijenjen modeliranju korisničkih sučelja računalnih programa
- Chemical Markup Language (CML) je jezik namijenjen prikazu kemijskih formula
- HelpML (Help Markup Language) je jezik namijenjen opisu dokumenata pomoći.

3.7. XML proširenja i tehnologije

- **XPath** je jezik koji omogućuje jednostavnije pretraživanje sadržaja u XML dokumentu.
- **XQuery** je upitni jezik za pretraživanje XML dokumenta. On je za XML isto ono što je SQL za relacijske baze podataka
- **XSLT** je jezik koji omogućuje transformacije XML dokumenata iz jednog formata u drugi (npr. iz XMLa u XHTML)
- **XSL-FO** je jezik koji služi za formatiranje izlaznog rezultata XML dokumenta
- **XML Linking Language** je jezik koji opisuje standardni način kreiranja hiperlinkova u XML dokumentima
- **XML Pointer Language** je jezik koji opisuje standardni način na koji hiperlinkovi pokazuju na određeno mjesto unutar XML dokumenta
- **XML Document Object Model (DOM)** je sučelje koje omogućuje računalnim programima pristup i ažuriranje sadržaja i strukture XML dokumenta
- **SOAP** je jednostavan protokol baziran na XMLu koji omogućuje aplikacijama razmjenu podataka u tekstualnom obliku preko HTTP protokola
- **Web Services Description Language (WSDL)** je jezik baziran na XMLu koji omogućuje opis web servisa i sučelja za njihovo korištenje
- **Really Simple Syndication (RSS)** je način za distribuciju sadržaja s jednog web-a na druge pomoću XMLa
- **Wireless Application Protocol (WAP)** je protokol razvijen s namjerom pristupa internetskim sadržajima pomoću prijenosnih uređaja (mobitela, ručnih računala i sl.)
- **XML Signature** definira sintaksu i pravila za kreiranje digitalnog potpisa XML sadržaja
- **XML Encryption** definira sintaksu i pravila za šifriranje XML sadržaja

3.8 Prednosti i nedostaci XMLa

Prednosti

- jednostavno je čitljiv i čovjeku u običnom tekstualnom editoru i računalu
- XML dokument je obična tekstovna datoteka čitljiva na svakoj platformi koja može čitati tekstovne podatke. To ga čini neosjetljivim na tehnološke promjene jer bez obzira na napredak tehnologije, tekstovni podaci će još jako dugo ostati nešto što će svaki računalni sustav trebati moći pročitati
- podržava Unicode i omogućuje prikaz teksta na svim danas poznatim jezicima
- format je samodokumentirajući. Oznake opisuju sadržaj koji se nalazi unutar njih.
- ima stroga sintaksna pravila tako da je jednostavno kontrolirati ispravnost nastalog dokumenta. Računalni programi koji obrađuju dokument zbog toga mogu jednostavno obrađivati XML sadržaj
- međunarodno prihvaćen standard. Mnogi proizvođači programa su ga prihvatili i koriste u svojim proizvodima
- hijerarhijska struktura je pogodna za opisivanje mnogih sadržaja (ali ne i svih!)
- kompatibilan je sa SGMLom koji se koristi od 80ih godina 20. stoljeća, a za SGML postoji dosta računalnih programa koji ga mogu obrađivati, pa kada je XML standard objavljen već je postojala određena baza korisnika koji nisu gotovo ništa morali učiti o novom jeziku niti kreirati nove programe već su ga jednostavno usvojili

Nedostaci

- sintaksa je redundantna i opširna što može zamarati i zbunjivati osobu koja čita XML dokument. računalni program koji obrađuje dokument morati će obraditi veliku količinu podataka što će ga djelomično usporiti
- da bi dokument bio dovoljno dobro "samoopisan" nazivi oznaka moraju biti dovoljno precizni što dovodi do dugih "kobasičastih" naziva (npr. u nekom dokument nije dovoljan opis podatka Ime_i_prezime jer to može biti ime i prezime osobe koja je kreirala dokument, ali i osobe na koju se slog odnosi, osobe kojoj se dokument šalje,... Nije dovoljno osloniti se na "samoopisivanje" sadržaja
- redundancija i velika količina podataka stvaraju velike zahtjeve za propusnosti mreže (iako to danas uglavnom više nije problem)
- programi koji obrađuju XML podatke su dosta složeni jer moraju obrađivati velike količine ugniježđenih podataka u više razina
- nedostatak formalno propisanih formata za podatke može stvarati probleme ako sudionici u razmjeni nisu dobro opisali (npr. da li se decimalni brojevi prikazuju s decimalnom točkom ili zarezom
- pohrana XML podataka u relacijske baze podataka nije prirodan način i to dovodi do smanjenja performansi sustava koji koriste takav način pohrane. S druge strane XML baze podataka koje su razvijene za pohranu XML podataka još su u fazi razvoja
-

3.9. Verzije XMLa

Postoje 2 verzije XMLa. Prva XML 1.0 inicijalno je stvorena 1998. godine. Do danas je imala nekoliko manjih revizija. Širom je prihvaćena i još i danas se preporučuje za korištenje. Verzija XML 1.0 bazira se na filozofiji da je sve što nije dozvoljeno zabranjeno. S razvojem drugih standarda na koje se oslanja (prvenstveno Unicode koji je u međuvremenu s verzije 2.0 došao do verzije 4.0 dolazilo je prvenstveno na velikim sustavima (IBM mainframe) do određenih problema jer se nisu mogli koristiti oni znakovi koju u vrijeme definiranja standarda nisu postojali. Druga verzija XML 1.1 inicijalno je objavljena 2004. godine i ima određena svojstva koja olakšavaju rad programima na velikim računalima. Njezin pristup je da je sve što nije zabranjeno dozvoljeno. Na takav način omogućuje se korištenje svih budućih Unicode znakova koji će se bilo kada u budućnosti definirati. Zbog svoje raširenosti uglavnom se još uvijek koristi verzija XML 1.0 jer je zadovoljavajuća za većinu korisnika. Korisnici na velikim

računalnim serverima koji imaju problema s ograničenjima verzije XML 1.0 uglavnom su prešli na korištenje verzije XML 1.1.

3.10. DTD

Namjera DTD pravila je opisati svaki element koji se koristi u razmatranom XML dokumentu, kao i međusobne odnose tih elemenata. Kako element može imati atribute, potrebno je posebnim pravilima opisati i njihov sadržaj i odnose.

Svaki element opisuje se <!ELEMENT .... > deklaracijom, koja u općem slučaju glasi:

<!ELEMENT *ime_elementa* *pravilo*>

Najjednostavnija i najopćenitija deklaracija elementa pretpostavlja da između početnog i dočetnog njegovog tag-a može biti bilo što (engl. *ANY*) tj. novi (ugnježđeni) tagovi ili znakovni podaci, informacija:

<!ELEMENT *ime_elementa* ANY> .

Ako želimo istaknuti da element ima sao podatke, a ne i nove tag-ove, onda koristimo kraticu PCDATA (od engl. *parsed character data*):

<!ELEMENT *ime_elementa* (#PCDATA)> .

Za element koji nema inforacijskog sadržaja unutar svojih tag-ova koristi se oznaka EMPTY u opisu elementa:

<!ELEMENT *ime_elementa* EMPTY> .

U slučaju gniježđenih elemenata, prvo se definiraju elementi više razine, grupirajući imena ugnježđenih elemenata unutar zagrada i odijeljujući ih znakom zareza ',' po obvezujućem poretku pojavljivanja ili znakom '|' ako se tek pretpostavlja pojavljivanje barem nekog od njih. Ime jednog elementa ili grupe elemenata može se proširiti posebnim znakom, operatorom pojavljivanja, za koje vrijedi:

Operator pojavljivanja	Opis
?	bez pojavljivanja ili jedanput (0 ili 1 put)
+	barem jedanput (1 ili više puta)
*	nijednom ili bilo koliko puta (0, 1 ili više)

Argumenat 'standalone' početne XML naredbe <?xml version="1.0"> govori o mjestu spremljenih DTD pravila, pa će <?xml version="1.0" standalone="yes"> DTD pravila uključivati na početku tog istog XML dokumenta, dok će <?xml version="1.0" standalone="no"> značiti da se DTD pravila nalaze u posebnoj datoteci određenoj preko <!DOCTYPE .... > naredbe:

<!DOCTYPE *ime_root_elementa* SYSTEM "URL">

gdje URL predstavlja adresu dokumenta na izvorišnom stroju u mreži. Ako se DTD objavljuje za opću uporabu, onda umjesto 'SYSTEM' stoji 'PUBLIC' sa svojim jedinstvenim imenom.

Primjer 3.4

Primjer DTD pravila	Opis
<?xml version="1.0" standalone="yes"> <!-- DTD definicije --> <!DOCTYPE knjiga [<!ELEMENT knjiga (naslov, izdavač) > <!ELEMENT naslov (#PCDATA)> <!ELEMENT izdavač (#PCDATA)>]> <!-- XML dokument --> <knjiga> <naslov> Uvod u XML </naslov> <izdavač> FSB Zagreb </izdavač> </knjiga>	Budući da je XML 'standalone' (samostojna) aplikacija, u istoj datoteci spremljena su i DTD pravila i XML dokument. Prvo je opisan root element 'knjiga' koja ima dva elementa od kojih prvo dolazi 'naslov', a potom 'izdavač'. Svaki od ta dva elementa sadrže čiste znakove podataka (PCDATA).
<!DOCTYPE knjiga [<!ELEMENT knjiga (naslov | izdavač) > <!ELEMENT naslov (#PCDATA)> <!ELEMENT izdavač (#PCDATA)>]>	U ovom slučaju XML dokument u svom elementu mogao bi imati ili element 'naslov' ili element 'izdavač', ali nikako oboje.
<!DOCTYPE knjiga [<!ELEMENT knjiga ((naslov, autor, izdavač)* | pregled+) <!ELEMENT naslov (#PCDATA)> <!ELEMENT autor (ime_autora+)> <!ELEMENT ime_autora (#PCDATA)> <!ELEMENT naslov (#PCDATA)> <!ELEMENT izdavač (tvrtka | logo)?> <!ELEMENT tvrtka (#PCDATA)> <!ELEMENT logo EMPTY> <!ELEMENT pregled (#PCDATA)>]>	U ovom slučaju XML dokument bi mogao imati jedan ili više pregleda, ili jedan ili više nizova elemenata ('naslov', 'autor','izdavač'), navedenima u tom (i samo tom) rasporedu ali nikako istodobno (i 'pregled' i grupa elemenata). Element 'autor' može imati jedan ili više elemenata 'ime_autora'. Na sličan način izdavač se može, a ne mora pojaviti i to sa znakovnim opisom (PCDATA) ili elementom 'logo' koji je prazan (što vrijedi npr. za elemente tipa slika).

Atributi se u DTD pravilima definiraju preko ATTLIST-e koja uz ime elementa i ime njegovog atributa definira atributski tip podatka i eventualnu pretpostavljenu vrijednost ili ključnu riječ umjesto nje. Postoji više tipova podataka koji se koriste u opisu atributa nekog XML elementa. Najčešći su obični znakovni podaci (CDATA), kao i nizovi vrijednosti od kojih se jedno izabire (engl. *enumerated types*). Ostali (ENTITY, ID, NMTOKEN, NOTATION) sa svojim varijantama zahtijevaju poseban opis. Pretpostavljene vrijednosti modifikatora u opisu atributa su:

Pretpostavljeni modifikatori	Opis
#REQUIRED	Atributna vrijednost mora se specificirati s elementom.
#IMPLIED	Atributna vrijednost može ostati nespecificirana u XML dokumnetu, ali se u DTD opisu mora navesti.
#FIXED	Atributna vrijednost je fiksna i ne može se mijenjati od strane korisnika

Svaki opis atributa mora sadržavati jedan modifikator, kao opću pretpostavljenu vrijednost. Opći oblik DTD naredbe za opis atributa je:

<!ATTLIST odredišni_element ime_atributa atributni_tip pretpostavljena_vrijednost>

Primjer 3.5.

Atributne deklaracije	Opis
<!ATTLIST cijena tečaj CDATA #REQUIRED "kn">	U elementu 'cijena' postoji atribut 'tečaj' koji se mora, zahtijeva (eng. reguired) da bude naveden, a pretpostavljena vrijednost (ako u XML dokumentu izostane) mu je 'kn'.
<!ATTLIST datum godina CDATA #FIXED "2002">	U elementu 'datum' postoji atribut 'godina' koji se ne mora navesti u XML dokumentu. U tom slučaju XML parser za vrijednost tog atributa uzet će fiksnu vrijednost '2002'.
<!ATTLIST osoba spol (muško | žensko) "žensko">	U elementu 'osoba' postoji atribut 'spol' koji se može poprimiti dvije vrijednosti ('muško' ili 'žensko'), a ako nje specificirano pretpostavlja se 'žensko'.
<!ATTLIST osoba bračno_stanje (sam | sama | oženjen | udata | rastavljen | rastavljena | udovac | udovica) #IMPLIED>	U elementu 'osoba' postoji atribut 'bračno_stanje' koji poprima jednu od nabrojenih vrijednosti.

ID, IDREF i IDREFS atributni tipovi podataka dopuštaju definiranje atributa kao jedinstvenih identifikatora elemenata. Oni su korisni za povezivanje različitih dijelova dokumenta, kao i referenciranje drugih ID-ova.

Primjer 3.6.

ID, IDREF	Opis
<?xml version="1.0" standalone="yes" ?> <!DOCTYPE odjel SYSTEM odjel.dtd> <odjel> <zaposlenik zapid="z007"> Marko Matić </zaposlenik> <zaposlenik zapid="z008"> Hrvoje Horvat </zaposlenik> <zaposlenik zapid="z009" šef="z007"> Darko Dragić </zaposlenik> <zaposlenik zapid="z010" šef="z008"> Štefica Štef </zaposlenik> </odjel>	Ovaj XML dokument ima vanjsku DTD datoteku 'odjel.dtd'. Elementi 'zaposlenik' koriste ID atribut 'zapid', a neki i ID atribut 'šef'. Ovaj element 'odjel' ima četiri elementa 'zaposlenik'.
<!-- odjel.dtd datoteka --> <!ELEMENT odjel (zaposlenik*)> <!ELEMENT zaposlenik (#PCDATA)> <!ATTLIST zaposlenik zapid ID #REQUIRED> <!ATTLIST zaposlenik šef IDREF #IMPLIED>	U DTD datoteci opisani su elementi i atributi prethodnog XML dokumenta. ID tip podatka govori da atribut 'zapid' mora biti jednoznačan identifikator. IDREF tip podatka govori da atribut 'šef' pokazuje na neki ID i ne mora (#IMPLIED) biti uključen.

IDREFS proširuje tip IDREF na više, prazninom odijeljenih, IDREF atributnih tipova podataka. Ovi identifikatori postaju posebno zanimljivi u radu s XLink i XPointer procesorima.

NMTOKEN i NMTOKENS opisuju atribute koji deklariraju XML ime (token). Takav XML token je podskup od #PCDATA jer sadrži pojedinačnu riječ (ne smije biti razmak u nizu znakova), a znakovi koji se smiju koristiti su standardni alfanumerički (velika i mala slova i znamenke) znakovi uključujući i one iz ostalih alfabeta (Grčki, Arapski, Hebrejski, ...) i ideograma (Kineski, Japanski, Koreanski). Od znakova interpunkcije dozvoljeni su samo: dvotočka ':', točka '.', crtica '-' i podvlaka '_'. XML token mora početi ili slovom ili podvakom '_'. Takva ograničenja na niz podataka u atributima koristimo najčešće u prebrojivim ključnim riječima programskih jezika, ali također i u svakodnevnoj uporabi, npr. za opis atributa koji obrađuje telefonske brojeve:

```
<!ATTLIST   poruka   tel   NMTOKEN   #REQUIRED>
```

Ključne riječi NOTATION i NOTATIONS dolaze u atributnoj listi neposredno povezane s DTD deklaracijom <!NOTATION> koja se poput <!DOCTYPE ... > ili neke druge DTD naredbe nalazi u opisu XML dokumenta.

Primjer 3.7.

NOTATION	Opis
<!NOTATION jpg SYSTEM "ACDSee.exe"> <!NOTATION gif SYSTEM "imaging.exe"> <!NOTATION bmp SYSTEM "paint.exe"> <!NOTATION png SYSTEM "netscape.exe"> <!ATTLIST mm gledaj NOTATIONS (jpg | gif | png) #REQUIRED>	Element 'mm' može za svoj atribut 'gledaj' odabrati jednu od tri vrste programa. Primjetite, da iako deklariran 'png' tip, ipak 'gledaj' ne može imati 'netscape.exe' kao vrijednost argumenta.

Unutar DTD-a moguće je preko ENTITY deklaracije povezati niz znakova s prikladnim imenom koje se potom na bilo kojim mjestima unutar XML dokumenta može koristiti kao refenca, bez ponovnog ispisa (općenito) duže niza znakova. To odgovara 'macro' naredbi u programskim jezicima. Opći oblik deklaracije izgleda ovako:

<!ENTITY *ime* "*znakovi_koje_zamjenjujemo_imenom*">

a referenca u XML dokumentu se koristi kao '*&ime;*'. Reference već definiranih zamjenskih imena u ENTITY deklaraciji mogu se koristiti unutar novo definiranog niza nove ENTITY deklaracije, ali samo u slučaju da ne nastanu cirkularne reference bez čvrstog početnog niza.
Na sličan način, upotrebom ključne riječi SYSTEM moguće je deklarirati vanjsku cjelinu (XML datoteku na nekoj lokaciji), koja se potom uz referentno ime uključuje (presnimava, kopira) u dokument.

<!ENTITY *ime* SYSTEM "*http://adresa/datoteka.xml*">

Ova deklaraciju moguće je proširiti i na sadržaje koji nisu XML dokumenti (npr. slike), upotrebom ključne riječi NDATA (engl. *notation data*) i specifikacijom tipa podatka preko NOTATION deklaracije.
Za upotrebu referentnog imena unutar DTD deklaracija koristi se drugi oblik ENTITY deklaracije:

<!ENTITY % *ime* "*znakovi_koje_zamjenjujemo_imenom*">

dakle, ispred zamjenskog imena nalazi se znak postotka '%', koji služi i kao prefiks reference imena tj. '*%ime;*' unutar DTD bloka. I ovdje također nije dozvoljeno činiti cirkularne reference u deklaracijama, a niti se zamjenska referenca smije napisati prije nego što je deklarirana (što je u prvom obliku ENTITY deklaracije bilo moguće).

Primjer 3.8.

ENTITY	Opis
<!-- ovo je DTD deklaracija --> <!ENTITY FSB "Fakultet strojarstva i brodogradnje"> <!-- a ovo je XML dokument --> <ustanova> Gdje je &FSB bio nekad uvaženi fakultet. </ustanova>	Referencom &FSB u XML dokument se upisuje puni naziv kratice, tj. 'Fakultet strojarstva i brodogradnje'.
<!-- ovo je DTD deklaracija --> <!ENTITY dodatak SYSTEM "http://zrno.fsb.hr/daj.xml"> <!-- ovo je XML dokument --> <ustanova> Gdje je nekad uvaženi fakultet imao i &dodatak; znanja. </ustanova>	U XML dokument pod element 'ustanova' preko reference '&dodatak;' umeće se cijeli dokument s mreže spremljen pod imenom 'daj.xml'.

<!-- ovo je DTD deklaracija -->
<!ENTITY slika SYSTEM "http://zrno.fsb.hr/daj.gif"
NDATA GIF89a>
<!NOTATION GIF89a SYSTEM "-
//CompuServe//NOTATION Graphics Interchange Format 89a
//EN">
 	
<!-- ovo je XML dokument -->
<ustanova>
Gdje je nekad uvaženi fakultet imao i <image src="&slika;">
adekvatan image.
</ustanova>
	U XML dokument pod element 'ustanova' preko reference '&slika;' umeće se slika s mreže spremljena pod imenom 'daj.gif' čiji je format opisan NOTATION tipom podatka u 'NOTATION' deklaraciji.

<!ENTITY A "Kolo sreće &B;"> <!ENTITY B "se okreće">	Dopuštena referenca.
<!ENTITY A "Kolo sreće &B;"> <!ENTITY B "&Ase okreće">	Nedopuštena cirkularna referenca.

Unutar DTD-a moguće je grupirati više naredbi koje se preko INCLUDE naredbe aktiviraju, a preko IGNORE zanemaruju u interpretaciji XML sadržaja. Opći oblik grupiranja DTD naredbi je:

```
<! [   (IGNORE | INCLUDE)
       [
           DTD naredbe .....
       ]
]>
```

koji se najčešće povezuje s uvjetovanim uključivanjem, preko reference.

Primjer 3.9.

INCLUDE, IGNORE	Opis
<!ENTITY % akoref "INCLUDE"> <! [%akoref; [<!ELEMENT knjiga ((naslov, autor, izdavač)* | pregled+) <!ELEMENT naslov (#PCDATA)> <!ELEMENT autor (ime_autora+)> <!ELEMENT ime_autora (#PCDATA)> <!ELEMENT naslov (#PCDATA)> <!ELEMENT izdavač (tvrtka | logo)?> <!ELEMENT tvrtka (#PCDATA)> <!ELEMENT logo EMPTY> <!ELEMENT pregled (#PCDATA)>]] >	Referencom '%ako;' u DTD dokument se upisuje skupina DTD deklaracija, ovisno o referenci koja se mijenja na jednom (ENTITY) mjestu. Ako je referenci 'akoref' pridružena ključna riječ 'INCLUDE' skupina će se obraditi, a ako je pridružena riječ 'IGNORE' skupina će se zanemariti.

3.11. XML schema

XML Schema je noviji način određivanja pravila strukturiranja XML dokumenta. Format se stvara prema pravilima XML jezika. Način određivanja strukture sličan je načinu izgradnje baze podataka. Korištenjem XML Scheme moguće je na vrlo detaljnoj razini odrediti opis sadržaja odgovarajućeg elementa:

- **prebrojivost**
- **tip podatka**
- **format podatka** (npr. može se odrediti format telefonskog broja koji se sastoji od pozivnog broja države, mjesta i samog telefonskog broja).

XML schemu je izvorno načinio Microsoft, ali je od 2001. godine to W3C preporuka.

3.11.1. Sintaksna pravila

Na početku XML Schema dokumenta navodi se identifikator koji povezuje dokument s pravilima formiranja, kao i određeni XML prostor imena. Nakon toga slijedi dio "annotation" u kojem se opisuje namjena dokumenta. Korijenski <schema> element mora se navesti u svakom XML Schema dokumentu. Taj element, kao i svi drugi može imati odgovarajuće atribute.

3.11.2. Tipovi podataka

Jednostavni tipovi podataka ugrađeni su u XML Schema specifikaciju. Jednostavni tipovi podataka mogu se proširivati specifičnim skupovima, ovisno o određenoj namjeni XML dokumenta koji se opisuje. Npr. kod XML dokumenata koji opisuju strukturu baze podataka postoje jednostavni tipovi podataka identični svim tipovima podataka koje određena baza može imati. Prednost korištenje jednostavnih tipova podataka je u tome što sve kontrole i provjere ispravnosti podataka vrši sam XML program i nije potrebno pisati posebne kontrolne procedure. Neki od jednostavnih tipova podataka su:

- numerički tipovi podataka ("byte", "float", "long")
- tipovi podataka za opis datuma vremena i trajanja ("time", "date", "timeinstant", "timeduration")
- logički tip podataka ("boolean" – može imati vrijednost "true" ili "false")
- tip podataka za unos binarnih brojeva ("binary")
- oznaka jezika koja se koristi ("language" – npr. "en-US")
- oznaka web adrese ("uri-reference" – npr. "http://www.w3c.org/")

3.11.3. Tipovi elemenata

U XML Schemi mogu se koristiti jednostavni i složeni tipovi elemenata.

Jednostavni tipovi elemenata

Jednostavni tipovi elemenata sadrže samo tekst i ne smiju unutar sebe sadržavati druge elemente i atribute (osim name i type atributa koji određuju naziv i tip određenog elementa).

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

Jednostavni elementi mogu imati pretpostavljenu (default) ili fiksno zadanu (fixed) vrijednost atributa. Ako u XML dokumentu nije zadana neka druga vrijednost atributa primjenjuje se inicijalna vrijednost. Fiksno zadana vrijednost ima kontrolni karakter i vrijednost u XML dokumentu ne smije biti različita od nje.

Složeni tipovi elemenata

Složeni tipovi elemenata su korisnički definirani elementom "complexType". Definira ih korisnik i njihovu kontrolu vrši sam. Postoje 4 različite vrste složenih elemenata

- prazni elementi ne smiju imati sadržaj već samo atribute. Prazni elementi imaju samo početnu oznaku.

```
<artikal sifra="1234"/>
```

- elementi koji sadrže druge elemente sadrže samo druge elemente koji sadrže text. Osim drugih elemenata ne smiju sadržavati unutar sebe text.

```
<osoba>
        <ime>Pero</ime>
        <prezime>Perić</prezime>
</osoba>
```

- elementi koji sadrže samo tekst sadrže samo tekst i stributem, a ne smiju unutar sebe imati sadržane druge elemente.

```
<racunalo type="rucno">Palm</racunalo>
```

- elementi koji sadrže i tekst i druge elemente unutar sebe mogu imati sadržan i tekst i druge elemente.

```
<opis>
  Osoba je rođena <datum lang="croatian">01.01.1989.</datum>
</opis>
```

3.11.4. Atributi elemenata

Atributi detaljnije opisuju elemente i opcionalni su te se ne moraju navoditi. Vrijednost atributa navodi se pod znakovima navodnika. Atribut "name" opisuje naziv atributa ili elementa, a atribut "type" njegov dozvoljeni tip. Tip podatka može biti jednostavan ili složen.

Prebrojivost elemenata

Kardinalnost podataka definirana je atributima "minOccurs" i "maxOccurs". Minimalni broj pojavljivanja određenog elementa postavljen je u osnovi na "1". Ako se želi navesti neki drugi broj, npr. da se element ne mora niti jednom pojaviti, potrebno je u vrijednost tog atributa upisati "0". Atribut "maxOccurs" navodi maksimalni broj puta što se može pojaviti vrijednost određenog elementa. Ako se neki element može pojaviti neograničen broj puta kao vrijednost se treba postaviti "*". Ako je broj puta što se element smije pojaviti fiksiran, potrebno je kao vrijednost atributa navesti taj broj.

Ograničenja elemenata

Odgovarajućim elementima i atributima mogu se precizno odrediti odgovarajuća ograničenja na vrijednost XML elemenata. Element <restriction> koristi se za ograničavanje vrijednosti određenog elementa na određene vrijednosti. Primjer ograničenja podatka starosti koje mora upasti u interval između 0 i 110:

```
<xs:element name="starost">
<xs:simpleType>
        <xs:restriction base="xs:integer">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="110"/>
        </xs:restriction>
</xs:simpleType>
</xs:element>
```

Primjer ograničenja podatka vrsta automobila koji mora biti jedan od ponuđenih tipova:

```
<xs:element name="automobil">
<xs:simpleType>
        <xs:restriction base="xs:string">
                <xs:enumeration value="Golf"/>
                <xs:enumeration value="Fiat"/>
        </xs:restriction>
</xs:simpleType>
</xs:element>
```

Primjer ograničenja na niz slova engleske abecede:

```
<xs:element name="slovo">
<xs:simpleType>
        <xs:restriction base="xs:string">
                <xs:pattern value="[a-z]"/>
        </xs:restriction>
</xs:simpleType>
</xs:element>
```

Primjer ograničenja duljine podatka:

```
<xs:element name="jmbg">
<xs:simpleType>
        <xs:restriction base="xs:string">
                <xs:length value="13"/>
        </xs:restriction>
</xs:simpleType>
</xs:element>
```

Ostala svojstva elemenata

Osim prebrojivosti i ograničenja podataka postoji niz predefiniranih svojstava koja se mogu koristiti prilikom opisa svakog elementa. Ta predefinirana svojstva nazivaju se "facets". Neki od najčešće korištenih su *precision, scale, encoding, whiteSpace, totalDigits, fractionDigits* ***i*** *pattern.*

3.12. Primjer upotrebe

XML dokument

Dokument **studenti.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<Studenti xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="studenti.xsd">
```

```
        <Osoba>
                <Ime>Pero</Ime>
                <Prezime>Perić</Prezime>
        </Osoba>
        <Studij>Računarstvo</Studij>
        <Adresa>
                <Ulica>U gradu</Ulica>
                <Broj>247</Broj>
                <Grad>Zagreb</Grad>
        </Adresa>
        <KontaktPodaci>
                <Telefon>01-111-777</Telefon>
                <Telefon>098-111-1778</Telefon>
                <Email>pperic@zg.tel.hr</Email>
                <KorisnickoIme>pperic</KorisnickoIme>
        </KontaktPodaci>
</Studenti>
```

XSD dokument

Dokument **studenti.xsd**

```
<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
   <xsd:element name='Studenti'>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref='Osoba'/>
            <xsd:element ref='Studij' minOccurs='0' maxOccurs='1'/>
            <xsd:element ref='Adresa' minOccurs='0' maxOccurs='1'/>
            <xsd:element ref='KontaktPodaci' minOccurs='0' maxOccurs='1'/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name='Osoba'>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref='Ime'/>
            <xsd:element ref='Prezime' minOccurs='0' maxOccurs='1'/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name='Ime' type='xsd:string'>
   </xsd:element>
   <xsd:element name='Prezime' type='xsd:string'>
   </xsd:element>
   <xsd:element name='Studij' type='xsd:string'>
   </xsd:element>
   <xsd:element name='Adresa'>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref='Ulica' minOccurs='0' maxOccurs='1'/>
            <xsd:element ref='Broj' minOccurs='0' maxOccurs='1'/>
            <xsd:element ref='Grad' minOccurs='0' maxOccurs='1'/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name='Ulica' type='xsd:string'>
   </xsd:element>
   <xsd:element name='Broj' type='xsd:string'>
   </xsd:element>
   <xsd:element name='Grad' type='xsd:string'>
   </xsd:element>
   <xsd:element name='KontaktPodaci'>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref='Telefon' minOccurs='0' maxOccurs='unbounded'/>
            <xsd:element ref='Email' maxOccurs='unbounded'/>
            <xsd:element ref='KorisnickoIme' maxOccurs='unbounded'/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name='KorisnickoIme' type='xsd:string'>
```

```
   </xsd:element>
   <xsd:element name='Telefon' type='xsd:string'>
   </xsd:element>
   <xsd:element name='Email' type='xsd:string'>
   </xsd:element>
</xsd:schema>
```

3.12.1. Prednosti i nedostaci korištenja XML Scheme

Prednost korištenja XML Scheme je što se sam XML Schema dokument XML dokument i stvara se prema istim pravilima. Time nije potrebno učiti pravila 2 različita jezika, a i sami programi koji provjeravaju ispravnost XML dokumenta mogu se koristiti za provjeru formalne ispravnosti Schema dokumenta.

Osim toga unutar XML Schema dokumenta moguće je provesti puno precizniju kontrolu sadržaja dokumenta i ispitati više različitih svojstava. Npr. ako se neki podatak opiše tipom datum, taj podatak u datumskom formatu trebao bi biti čitljiv u svim zemljama iz razloga jer tip podataka datum programi bi trebali interpretirati na ispravan način, ovisno o lokalnim datumskim postavkama odgovarajuće zemlje.

Nedostatak korištenja XML Schema dokumenta je u dosta složenoj sintaksi koju je potrebno svladati kako bi se stvorili dokumenti.

Izvor: Wikipedija, w3school