Predgovor

U nekoliko poslednjih godina, XML je počeo da se upotrebljava u najrazličitijim oblastima, među kojima su: pravo, aeronautika, finansije, osiguranje, robotika, multimedija, ugostiteljstvo, turizam, likovne umetnosti, graditeljstvo, telekomunikacije, softver, poljoprivreda, fizika, novinarstvo, teologija, maloprodaja i stripovi. XML je postao najpoželjnija sintaksa za nove formate dokumenata u gotovo svim računarskim aplikacijama. Upotrebljava se na Linuxu, Windowsu, Macintoshu i mnogim drugim računarskim platformama. Veliki centralni računari u njujorškom Volstritu međusobno trguju deonicama tako što razmenjuju odgovarajuće XML dokumente. Deca koja igraju igrice na kućnim računarima snimaju svoje dokumente u XML-u. Navijači na ekranima svojih mobilnih telefona u realnom vremenu primaju rezultate utakmica, opet u formatu XML. XML je jednostavno najrobusnija, najpouzdanija i najfleksibilnija dosad izumljena sintaksa za dokumente.

XML za programere je sveobuhvatan vodič kroz brzo rastući svet XML-a. Obrađuje sve vidove XML-a – od osnovnih pravila sintakse, preko pojedinosti pravljenja DTD-ova i šema, do interfejsa za programiranje aplikacija (API-ja) pomoću kojih možete čitati i pisati XML dokumente na raznim programskim jezicima.

Šta obuhvata ova knjiga

Postoje hiljade XML aplikacija koje su formalno ustanovili konzorcijum W3C i druga tela za standardizaciju, kao sto su OASIS i Object Management Group. Jos je više nestandardizovanih aplikacija, poput Microsoftove Channel Definition Format i Mind Reading Markup Language Johna Guajarda. Sve njih nije moguće opisati u jednoj knjizi, kao što se u jednoj knjizi ne mogu opisati ni svi programi koji se mogu napisati ili su već napisani na Javi. Ova knjiga se usredsređuje na sam XML. Obuhvaćena su osnovna pravila kojih se moraju pridržavati svi XML dokumenti i njihovi autori – od Web dizajnera koji upotrebljavaju SMIL za dodavanje animacija na Web stranice, do C++ programera koji za razmenu serijalizovanih objekata sa udaljenom bazom podataka koriste SOAP.

U knjizi su opisane i sledeće opšte tehnologije za podršku XML-a, koje se koriste u raznim XML aplikacijama:

XLink
: Atributska sintaksa za hiperveze između XML i ne–XML dokumenata. One mogu biti: proste jednosmerne veze poznate iz HTML-a, višesmerne veze između više dokumenata i veze između dokumenata u koje ne možete upisivati.

XSLT
: XML aplikacija koja opisuje transformacije jednog dokumenta u drugi, pri čemu se koriste isti ili različiti XML rečnici.

XPointer
: Sintaksa za URI identifikatore odlomaka, kojima se biraju određeni delovi XML dokumenta na koji upućuje URI identifikator; često se koristi zajedno sa XLinkom.

XPath
: Ne-XML sintaksa koju XPointer i XSLT upotrebljavaju za identifikovanje određenih delova XML dokumenata. Primera radi, pomoću XPatha može se izdvojiti treći element adrese dokumenta ili svi elementi sa atributom `email` čija je vrednost `elharo@metalab.unc.edu`.

XInclude
: Sredstvo za pravljenje velikih XML dokumenata kombinovanjem drugih gotovih dokumenata i njihovih odlomaka.

Prostori imena
: Sredstvo za uočavanje razlika između elemenata i atributa iz različitih, ali istoimenih XML rečnika; na primer, za razlikovanje naslova knjige od naslova Web stranice koja je sastavni deo Web lokacije o knjigama.

Šeme
: XML rečnik za opisivanje dozvoljenog sadržaja XML dokumenata napravljenih pomoću drugih XML rečnika.

SAX
: Akronim od *Simple API for XML*, što je naziv interfejsa za programiranje aplikacija zasnovanih na događajima; implementiraju ga mnogi analizatori (raščlanjivači) XML teksta.

DOM
: Akronim od *Document Object Model* (objektni model dokumenta), što je ime jezički nezavisnog interfejsa za programiranje aplikacija, koji XML dokument tretira kao stablo ugnežđenih objekata različitih svojstava.

XHTML
: XML-izovana verzija HTML-a, koja se može proširiti drugim XML aplikacijama, kao što su MathML i SVG.

RDDL
: Akronim od *Resource Directory Description Language* (jezik za opisivanje kataloga resursa), što je naziv XML aplikacije zasnovane na jeziku XHTML, koja je namenjena dokumentima smeštenim na kraj URL adrese prostora imena.

Navedene tehnologije se koriste u više različitih XML aplikacija, bez obzira na to da li su definisane u XML-u (kao XLinks, XSLT, XInclude, prostori imena, šeme, XHTML i RDDL) ili u nekoj drugoj sintaksi (kao XPointers, XPath, SAX i DOM).

U ovoj knjizi nisu detaljno objašnjene XML aplikacije koje su važne samo manjem delu korisnika XML-a. Među takvim aplikacijama su:

SVG
: *Scalable Vector Graphics* – standard za XML kodiranje vektorskih crteža, koji je usvojio W3C.

MathML
: *Mathematical Markup Language* –XML aplikacija za ugradnju jednačina u Web stranice i druge dokumente, koju je W3C usvojio kao standard.

RDF
: *Resource Description Framework* –XML aplikacija za opisivanje resursa, naročito metapodataka koji se mogu naći u kartičnom katalogu biblioteka. Prihvaćena u W3C kao standard.

Ponekad ćemo neku od tih aplikacija upotrebiti u primerima, ali nećemo detaljno objašnjavati sve aspekte odgovarajućeg rečnika. Te aplikacije su zanimljive i važne, ali su (kao i hiljade drugih) prvenstveno namenjene upotrebi u specijalnom softveru koji poznaje sve pojedinosti njihovih formata. Na primer, većina grafičkih dizajnera ne radi neposredno sa SVG-om, nego SVG dokumente pravi pomoću svojih uobičajenih alatki, kao što je Adobe Illustrator. Oni ne moraju ni znati da koriste XML.

Ova knjiga se usredsređuje na standarde koji su važni gotovo svim programerima što rade u XML-u. Istražićemo XML tehnologije zajedničke velikom broju primena XML-a, a ne one koje su važne u samo nekoliko ograničenih domena.

Šta je novo u trećem izdanju

XML nije stajao u mestu tokom dve godine protekle od drugog izdanja knjige *XML in a Nutshell*. Najvidljivija promena u ovom izdanju jeste to što ono opisuje XML 1.1. Međutim, izmene u verziji 1.1 nisu toliko velike kao što bi novi broj verzije (.1) implicirao. Zapravo, ako ne govorite mongolski, burmanski, amharski, kambodžanski ili još nekoliko manjih jezika, u XML-u 1.1 ima veoma malo novog materijala zanimljvog za vas. XML 1.0 i 1.1 isti su u gotovo svakom praktičnom pogledu. Razlika između njih je sigurno mnogo manja od razlike između Jave 1.0 i 1.1. Zato ćemo u ovoj knjizi razmatrati XML kao jedan entitet, a na XML 1.1 eksplicitno upućivati samo u retkim prilikama kada se te verzije zaista razlikuju. Verovatno se otprilike 98% knjige jednako dobro odnosi i na XML 1.0 i na XML 1.1.

Dodali smo i poglavlje posvećeno XIncludeu, novijem pronalasku konzorcijuma W3C; koristi se za pravljenje velikih XML dokumenata kombinovanjem manjih dokumenata i njihovih delova. Elliotte je napisao gotovo polovinu prvih realizacija XIncludea, a i prvu knjigu napravljenu pomoću XIncludea, pa nam je ova tema posebno značajna. Ostala poglavlja knjige prerađena su koliko je zahtevao uticaj verzije 1.1 XML-a na obrađene teme, kao i uticaj promena datih tehnologija u pomenute dve godine. Mnoge teme su ažurirane tako da odgovaraju poslednjoj verziji odgovarajuće specifikacije, i to:

- SAX 2.0.1
- Prostori imena 1.1
- DOM Level 3
- XPointer 1.0
- Unicode 4.0.1

I najzad, u knjizi su ispravljene mnoge male greške i propusti.

Organizacija knjige

Deo I, *Koncepti XML-a*, uvodi osnovne standarde koji čine jezgro XML-a i kojih se moraju pridržavati sve XML aplikacije i softver. U njemu ćete ukratko saznati šta je dobro oblikovan XML dokument, i šta su definicija tipa dokumenta (DTD), prostor imena i Unicode.

Deo II, *Narativni dokumenti*, istražuje tehnologije koje se uglavnom upotrebljavaju za XML dokumente narativnog tipa, kao što su Web stranice, knjige, članci, dnevnici i drame. Saznaćete šta su i kako rade XSLT, CSS, XSL-FO, XLinks, XPointers, XPath, XInclude i RDDL.

Među najneočekivanijim događajima u istoriji XML-a bilo je to što je široko prihvaćen za izradu strukturiranih dokumenata punih podataka, kao što su proračunske tabele (engl. *spreadsheets*), finansijski statistički podaci, matematičke tabele i formati datoteka. Deo III, *Dokumenti slični zapisima baze podataka*, istražuje upotrebu XML-a u takvim primenama. U njemu ćemo se usredsrediti na alatke i interfejse za programiranje aplikacija koji su potrebni za pisanje softvera za obradu XML-a – a to su SAX, DOM i šeme.

Najzad, deo IV, *Referentni pregled osnovnih XML tehnologija*, predstavlja niz referentnih poglavlja koja čine jezgro svake knjige iz ove edicije. U njima su data detaljna sintaktička pravila za osnovne XML tehnologije, među kojima su XML, DTD-ovi, šeme, XPath, XSLT, SAX i DOM. Pređite u taj deo knjige kada hoćete da brzo pronađete preciznu sintaksu nečega što znate da možete da uradite, ali ne znate kako.

Konvencije korišćene u knjizi

`Tekst konstantne širine` koristi se za:

- Sve što se može pojaviti u XML dokumentu, kao što su imena elemenata, oznake (engl. *tags*), vrednosti atributa, reference entiteta i instrukcije za obradu.
- Sve što se može pojaviti u programu, kao što su rezervisane reči, operatori, imena metoda, imena klasa i literali.

`Polucrni tekst konstantne širine` koristi se za:

- Ono što unosi korisnik.
- Naglašavanje u primerima i odlomcima koda.

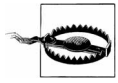
`Kurziv konstantne širine` upotrebljen je za:

- Zamenljive elemente u programskom kodu.

Kurziv se upotrebljava za:

- Nove termine na mestu definicije i termine na engleskom.
- Naglašavanje u telu teksta.
- Putanje, imena datoteka i programa. (Međutim, ako je ime programa ujedno i ime Javine klase, napisano je fontom konstantne širine, kao i ostala imena klasa.)
- Imena domena i mrežnih računara (*cafeconleche.org*).

Ova sličica ukazuje na savet, predlog ili opštu napomenu.



Ova sličica ukazuje na upozorenje ili opomenu.

Značajni delovi koda, celi programi i dokumenti obično su ovako smešteni u zaseban pasus:

```
<?xml version="1.0"?>
<?xml-stylesheet href="person.css" type="text/css" ?>
<osoba>
  Alen Tjuring
</osoba>
```

XML uzima u obzir razliku između velikih i malih slova. Element `OSOBA` nije jednak elementu `osoba`, niti elementu `Osoba`. Pisci na jezicima koji uzimaju u obzir razliku između velikih i malih slova ne mogu uvek da se drže gramatičkih pravila standardnog engleskog jezika. Obično se rečenica može preformulisati tako da se poštuju pravila i XML-a i engleskog, i mi smo se potrudili da tako uradimo kad god je bilo moguće. Međutim, u retkim slučajevima problem nije bilo moguće zaobići, i tada je engleski lošije prolazio.

Premda je većina primera upotrebljenih u knjizi pojednostavljena da bi se mogla još negde upotrebiti, nekoliko njih ima stvarnu vrednost. Slobodno ih upotrebite u svom kodu – cele ili samo delove. Za to vam nije potrebna specijalna dozvola. Što se nas tiče, oni su javno vlasništvo (što svakako ne važi za tekst objašnjenja).

Komentari čitalaca

Biće nam drago da čujemo opšte komentare čitalaca o tome kako bi se knjiga mogla poboljšati – konkretnim ispravkama ili temama koje bi trebalo obuhvatiti. Adrese elektronske pošte autora knjige jesu *elharo@metalab.unc.edu* i *smeans@ewm.biz*. Molimo vas, uzmite u obzir da nas dvojica svakodnevno primamo više stotina elektronskih poruka i da ne možemo odgovoriti na svaku. Najveće šanse da dobiju odgovor imaju oni koji se predstave kao čitaoci ove knjige. Takođe vas molimo da svoje poruke šaljete s naloga e-pošte na koji treba poslati odgovor i u kome je adresa za odgovor tačno zadata. Ništa nas toliko ne obeshrabruje kao kada nam se, nakon što utrošimo sat i više vremena na proučavanje i pisanje detaljnog odgovora na zanimljivo pitanje, odgovor vrati zato što je sagovornik svoju poruku poslao s javnog

terminala, a prethodno se nije potrudio da upiše svoju pravu adresu e-pošte među parametre čitača Weba.

Proverili smo i testirali informacije koje navodimo u knjizi, ali se ipak može desiti da se mogućnosti opisanih tehnologija promene (ili ćete pronaći grešku u našem opisu). Pristalice smo stare izreke „Ako vam se knjiga svidi, recite to svojim prijateljima. Ako vam se ne svidi, recite to nama". Naročito bismo hteli da nam skrenete pažnju na eventualne greške. Koliko god da su se pisci i redakcija potrudili oko ove knjige, mora biti da nam je ipak promaklo nekoliko grešaka. Ako pronađete neku, molimo vas da nas obavestite kako bismo mogli da je ispravimo u sledećem izdanju. Sve podatke o greškama molimo da pošaljete neposredno autorima ove knjige na prethodno navedene adrese e-pošte.*

Otvorena je i Web lokacija originala ove knjige, na kojoj su navedene pronađene greške, primeri i dodatne informacije. Adresa te lokacije je:

http://www.cafeconleche.org/books/xian3/

Pre nego što utrošite vreme na prijavu greške, molimo vas da posetite ovu lokaciju i pogledate da li je tamo već objavljena ispravka. Tehnička pitanja i komentare šaljite e-poštom neposredno autorima knjige ili izdavačima, na adresu:

bookquestions@oreilly.com, odnosno, *redakcija@mikroknjiga.co.yu*

Više informacija o drugim izdanjima, softveru i ostalim resursima koje O'Reilly stavlja na raspolaganje javnosti, potražite na adresama:

http://www.oreilly.com
http://xml.oreilly.com
http://www.xml.com

Više informacija o drugim izdanjima i aktivnostima Mikro knjige potražite na adresi:

www.mikroknjiga.co.yu

* Većina primera je prevedena, pa je moguće da su se i tu potkrale greške. Ako primetite grešku u prevedenom izdanju, molimo vas da pošaljete poruku na adresu *redakcija@mikroknjiga.co.yu*.

Koncepti XML-a

1

Uvod u XML

XML je akronim od eXtensible Markup Language, a konzorcijum W3C prihvatio ga je kao standard za označavanje (engl. *markup*) dokumenata. On definiše opštu sintaksu za označavanje podataka jednostavnim, svima razumljivim oznakama (engl. *tags*). XML obezbeđuje standardan format za računarske dokumente, dovoljno fleksibilan da se može prilagoditi za najrazličitije oblasti kao što su Web lokacije, elektronska razmena podataka, vektorska grafika, genealogija, ponuda nekretnina, serijalizacija objekata, pozivanje udaljenih procedura, sistemi glasovne pošte itd.

Možete pisati sopstvene programe za rad s podacima u XML dokumentima i njihovu obradu. Ako to uradite, postaće vam dostupan veliki broj besplatnih biblioteka na raznim jezicima na kojima se može čitati i pisati XML, pa ćete moći da se usredsredite baš na ono što je potrebno vašem programu. Za rad sa XML dokumentima možete upotrebiti i gotov softver, kao što su čitači Weba (engl. *Web browsers*) ili programi za obradu teksta. Jedan deo alatki može da radi sa svim XML dokumentima. Druge su prilagođene za podršku određenoj XML aplikaciji u nekoj oblasti – poput vektorske grafike – i izvan nje su najčešće neupotrebljive. No, u svim slučajevima koristi se ista sintaksa, čak i ako je namerno sakrivena u alatkama koje se lakše koriste ili ograničena na jednu aplikaciju.

Prednosti XML-a

XML je metajezik za označavanje tekstualnih dokumenata. Podaci se smeštaju u XML dokumente u obliku znakovnih nizova (engl. *strings*), između tekstualnih oznaka koje ih opisuju. Osnovne jedinice podataka i oznaka u XML-u nazivaju se *elementi*. XML specifikacija precizno definiše sintaksu koje se morate pridržavati pri pisanju oznaka: kako su elementi razgraničeni oznakama, kako oznaka izgleda, kakva imena elementi mogu imati, gde se stavljaju atributi itd. Površno gledano, oznake XML dokumenta mnogo liče na oznake HTML dokumenta, ali među njima postoje bitne razlike.

Najvažnije je da je XML *jezik za metaoznačavanje*. To znači da on nema fiksan skup oznaka i elemenata koji bi trebalo da zadovolje svačije potrebe u svim oblastima i zauvek. Svaki pokušaj da se napravi konačan skup takvih oznaka, osuđen je na neuspeh. Umesto toga, XML omogućava programerima i piscima da izmišljaju potrebne elemente onda kad im zatrebaju. Hemičari mogu upotrebljavati elemente koji opisuju molekule, atome, veze, reakcije i ostale entitete koji se sreću u hemiji. Agenti za prodaju nekretnina mogu upotrebljavati elemente koji opisuju stanove, stanarine, provizije, lokacije i druge entitete potrebne za nekretnine. Muzičari mogu upotrebljavati elemente koji opisuju četvrtine nota, polovine nota, violinske ključeve, tekstove pesama i ostale objekte uobičajene u muzici. Slovo X u imenu XML potiče od reči *Extensible* (proširiv), što znači da se jezik može proširivati i prilagođavati da bi zadovoljio različite potrebe.

Iako je XML veoma fleksibilan kad je reč o elementima koji se mogu koristiti, veoma je strog u mnogim drugim aspektima. Specifikacija XML-a definiše gramatiku XML dokumenata, koja kazuje gde se oznake moraju staviti, kako one moraju izgledati, kakva su imena elemenata dozvoljena, kako se elementima pridružuju atributi itd. Ta gramatika je dovoljno specifična da omogućava pravljenje raščlanjivača i analizatora sintakse XML-a, u daljem tekstu – analizatora (engl. *parser*), koji mogu pročitati svaki XML dokument. Za dokumente koji zadovoljavaju pravila te gramatike kažemo da su *dobro oblikovani* (engl. *well-formed*). Dokumenti koji nisu dobro oblikovani bivaju odbijeni, kao što biva odbijen svaki C program koji sadrži sintaksnu grešku. Programi za obradu XML-a (engl. *XML processors*) odbijaju dokumente koji nisu dobro oblikovani.

Radi interoperabilnosti, pojedinci i organizacije mogu se dogovoriti da upotrebljavaju samo određene oznake. Takve skupove XML oznaka nazivamo *primene XML-a* (engl. *XML applications*) ili XML aplikacije. XML aplikacija nije softverska aplikacija koja upotrebljava XML, kao što su Mozilla ili Microsoftov Word. To su primene XML-a u određenoj oblasti – na primer u vektorskoj grafici ili u kulinarstvu.

Oznake u XML dokumentu opisuju njegovu strukturu. Pomoću njih možete videti koji elementi su pridruženi kojim drugim elementima. U dobro projektovanom XML dokumentu, oznake opisuju i semantiku dokumenta. Primera radi, oznaka može ukazati na to da je element datum ili ime osobe ili bar-kod. U dobro projektovanim XML aplikacijama, oznake ništa ne kazuju o tome kako dokument treba prikazati. Drugim rečima, ne kazuju da li je određeni element ispisan polucrno ili kurzivom ili je stavka nabrajanja u listi. XML je jezik za označavanje strukture i semantike, a ne za označavanje načina prikazivanja.

Postoji nekoliko XML aplikacija za opisivanje načina predstavljanja teksta; jedna takva je XSL Formatting Objects (XSL-FO). Međutim, to su izuzeci koji potvrđuju pravilo. Iako XSL-FO opisuje prezentaciju, XSL-FO dokument se nikada ne piše direktno. Umesto njega napisali biste semantički bolje strukturiran XML dokument, a potom biste upotrebili opis stilova XSL Transformations da biste strukturno orijentisani XML izmenili u prezentacijski orijentisan XML.

Oznake dozvoljene u određenoj primeni XML-a mogu se dokumentovati u *šemi* (engl. *schema*). Sa šemom se mogu porediti pojedini primerci dokumenta. Za dokumente koji zadovoljavaju šemu kažemo da su *validni* (engl. *valid*). Za one koji ne

zadovoljavaju tu šemu kažemo da su, u odnosu na nju, *nevalidni* (engl. *invalid*). Dakle, validnost dokumenta zavisi od šeme s kojom se dokument poredi. Ne moraju svi dokumenti biti validni. Za mnoge primene dovoljno je da dokument bude dobro oblikovan.

Postoji mnogo raznih jezika za XML šeme i njihovi nivoi izražajnosti su različiti. Najrašireniji jezik za XML šeme i jedini definisan u samoj specifikaciji XML-a jeste *definicija tipa dokumenta* (engl. *document type definition*, DTD). Svaki DTD navodi sve dozvoljene oznake i određuje gde se i na koji način one mogu pojaviti u dokumentu. U XML-u DTD-ovi nisu obavezni, nego opcioni. S druge strane, DTD-ovi nisu uvek dovoljni. Sintaksa DTD-ova je veoma ograničena i ne omogućava pravljenje raznih korisnih iskaza poput „Ovaj element sadrži broj" ili „Ovaj znakovni niz je datum koji pada između 1974. i 2032". Takva ograničenja možete izraziti jezikom XML Schema Language, koji je prihvatio W3C. (Taj jezik se ponekad pogrešno naziva opštim imenom *schemas*, tj. šeme.) Pored DTD-a i XML Schema Languagea, postoji još mnogo jezika za opisivanje šema, među kojima su RELAX NG, Schematron, Hook, Examplotron itd.

Trenutno su svi jezici za šeme čisto deklarativni. Međutim, uvek ima ograničenja koja se ne mogu izraziti ukoliko programski jezik nije potpun u Tjuringovom smislu. Na primer, ako je dati XML dokument narudžbenica, jezik mora biti potpun u Tjuringovom smislu da bi se `cena` svake `stavke_narudžbenice` pomnožila njenom `količinom`, sve to sabralo, i proverilo da li je zbir jednak elementu `međuzbir`. Današnji jezici za šeme ne mogu proveriti ni ograničenja koja su spoljna u odnosu na dokument, kao što je „Svaki SKU element odgovara SKU polju zapisa u tabeli proizvoda unutar baze podataka inventara". Ako pišete program za čitanje XML dokumenata, možete mu dodati kôd za proveru takvih iskaza, kao što biste uradili da pišete kôd za čitanje tekstualne datoteke kojoj kao graničnik služi znak tabulator. Razlika je u tome što XML analizatori predstavljaju podatke u mnogo podesnijem formatu i obavljaju veći deo posla umesto vas, pa sami morate da dopišete manje koda.

Šta XML nije

XML je jezik za označavanje i ništa više od toga. To treba zapamtiti. Priča o XML-u se toliko naduvala da ima ljudi koji od XML-a očekuju da radi sve i svašta, ako treba i kola da opere.

Pre svega, *XML nije programski jezik*. Ne postoji kompajler XML-a koji čita XML datoteke i daje izvršni kôd. Eventualno biste mogli definisati skript jezik koji koristi format XML-a kao matični, a interpretira ga neki binarni program, ali bi čak i ta primena bila neobična. XML se može upotrebiti kao format naredaba u programima koji nešto rade, kao što i tradicionalni programi mogu čitati tekstualne konfiguracijske datoteke i preduzimati razne akcije na osnovu pročitanog. Zaista nema razloga da konfiguracijska datoteka ne bude u formatu XML, umesto u formatu nestrukturiranog teksta. Neki noviji programi upotrebljavaju XML konfiguracijske datoteke; ali je uvek program taj koji preduzima akciju, a ne sam XML dokument. XML dokument *ne radi* ništa, on samo *postoji*.

Barem jedna XML aplikacija – XSL Transformations (XSLT) – dokazano je potpuna u Tjuringovom smislu. (Dokaz je izveden konstrukcijom, tj. pravljenjem mašine koja je potpuna.) Na lokaciji *http://www.unidex.com/turing/utm.htm* možete videti univerzalnu Tjuringovu mašinu napisanu u XSLT-u.

Drugo, *XML nije protokol za mrežni prenos*. XML ne šalje podatke preko mreže, kao ni HTML. Podaci poslati preko mreže HTTP-om, FTP-om, NFS-om ili nekim drugim protokolom, mogu biti kodirani u XML-u; ali opet mora postojati neki softver izvan XML dokumenta koji će poslati dokument.

Najzad, da pomenemo primer gde priče najčešće sakrivaju istinu, *XML nije baza podataka*. XML-om ne možete zameniti Oracle ili MySQL server. Baza podataka može sadržati XML podatke, bilo kao VARCHAR ili BLOB ili neki namenski XML tip podataka, ali sama baza podataka nije XML dokument. XML podatke možete smestiti u bazu podataka na serveru i preuzeti ih iz nje u formatu XML, ali vam za to treba softver napisan na pravom programskom jeziku kao što su C ili Java. Da bi smestio XML podatke u bazu podataka, softver na klijentskoj strani šalje ih serveru pomoću ustanovljenog mrežnog protokola kao što je TCP/IP. Softver na serverskoj strani prima XML podatke, raščlanjuje ih i smešta u bazu. Da biste preuzeli XML dokument iz baze podataka, po pravilu ćete morati da upotrebite neki posrednički program (engl. *middleware product*) kao što je Enhydra, koja će napraviti i poslati SQL upite bazi podataka, a skup rezultata formatirati kao XML pre nego što ih vrati klijentu. Činjenica je da neke baze podataka integrišu taj softver u svoje servere ili za obavljanje te funkcije obezbeđuju softverske dodatke, kao što je Oracleov servlet XSQL. U tim scenarijima, XML služi veoma dobro kao sveprisutan prenosni format, nezavisan od platforme. Međutim, on nije baza podataka, niti ga tako treba upotrebljavati.

Prenosivi podaci

XML pruža primamljivu mogućnost pravljenja dugotrajnih formata podataka nezavisnih od platforme. Već dugo dokumenti pisani na jednoj platformi nisu uvek čitljivi na drugim platformama, niti su čitljivi u različitim programima na istoj platformi, pa čak ni u prethodnim ili budućim verzijama istog programa na istoj platformi. A kada je dokument čitljiv, to još ne znači da će se baš svi podaci uspešno pročitati. Veliki deo podataka prikupljenih tokom prvih iskrcavanja na Mesec, krajem šezdesetih i početkom sedamdesetih godina, danas je izgubljen. Čak i ako pronađete uređaj s trakama koji može da čita podatke s traka kakve se već odavno ne koriste, niko više ne zna u kom formatu su ti podaci bili zapisani!

XML je neverovatno jednostavan i dobro dokumentovan format podataka. XML dokumenti su tekstualni, pa ih može čitati svaka alatka koja ume da čita tekstualne datoteke. Tekstualni su ne samo podaci, nego i oznake, a smeštene su u samoj XML datoteci. Ne morate brinuti o tome da li je svaki osmi bajt nasumična popuna (engl. *padding*), pogađati da li je četvorobajtna vrednost ceo broj zapisan kao komplement broja dva ili broj u formatu pokretnog zareza zapisan po standardu IEEE 754, niti pokušavati da dešifrujete koji se celobrojni kôd odnosi na koje svojstvo formatiranja. Nazive oznaka možete čitati neposredno i iz njih ćete tačno saznati šta je u dokumentu. Slično tome, pošto oznake definišu granice elemenata, mali su izgledi

da vas prevari neočekivana konvencija o završetku reda ili broj uzastopnih razmaka koji se pretvara u tabulator. U XML-u su sve važne pojedinosti strukture dokumenta eksplicitne. Ne morate da rekonstruišete format unazad, niti da se oslanjate na nepotpunu i često nedostupnu dokumentaciju.

Možda će neki proizvođači softvera pokušati da zarobe svoje kupce nedokumentovanim i nestandardnim binarnim formatom podataka. Međutim, dugoročno će svima biti bolje ako budemo mogli da koristimo jasno dokumentovane, dobro poznate tekstualne formate koji se lako raščlanjuju, kakve pruža XML. On omogućava da se podaci i dokumenti prenose s jednog sistema na drugi, uz razumno očekivanje da će prijemni sistem moći da ih shvati. Nadalje, proverom validnosti prijemna strana može ustanoviti da li je dobila ono što je očekivala. Java je obećala prenosiv programski kôd; XML daje prenosive podatke. Na mnogo načina, XML je najprenosiviji i najfleksibilniji format dokumenata od pojave tekstualne ASCII datoteke.

Kako XML radi

U primeru 1-1 prikazan je jednostavan XML dokument, kakav se sreće u sistemu za upravljanje inventarom ili u bazi podataka skladišta. Njegovi podaci su obeleženi oznakama (engl. *tags*) i atributima koji opisuju boju, veličinu, broj bar-koda, ime proizvođača, ime proizvoda itd.

Primer 1-1. XML dokument

```
<?xml version="1.0"?>
<product barcode="2394287410">
    <manufacturer>Verbatim</manufacturer>
    <name>Datalife MF 2HD</name>
    <quantity>10</quantity>
    <size>3.5" </size>
    <color>crna</color>
    <description>diskete</description>
</product>
```

Ovaj dokument je tekst, pa se može smestiti u tekstualnu datoteku koju možete uređivati u svakom standardnom programu za obradu teksta, kao što su BBEdit, jEdit, UltraEdit, Emacs ili vi. Za XML vam nije potreban specijalan program za obradu teksta. Štaviše, smatramo da većina XML editora opšte namene više košta nego što vredi i da ih je mnogo teže koristiti od klasičnih programa za obradu teksta.

Programi koji zaista hoće da shvate sadržaj XML dokumenta – dakle, koji s njim imaju veće ambicije nego da ga tretiraju kao bilo koju drugu datoteku – za čitanje dokumenta upotrebljavaju *XML analizator*. Analizator raščlanjuje dokument na elemente, atribute i ostale komponente. On aplikaciji prosleđuje sadržaj XML dokumenta parče po parče. Ako analizator u bilo kom trenutku otkrije da dokument krši XML-ova pravila o dobrom oblikovanju, aplikaciji će prijaviti grešku i prestaće da analizira dokument. U nekim slučajevima, analizator i posle prve pronađene greške nastavlja da čita dokument, da bi mogao da otkrije i prijavi i ostale greške. Međutim, nakon otkrivanja prve greške u dobrom oblikovanju, analizator uvek prestaje da prosleđuje sadržaj elemenata i atributa koje prepoznaje.

Aplikacije najčešće imaju preciznija pravila o tome koji su elementi i atributi dozvoljeni na kom mestu u dokumentu. Na primer, u dokumentu o biologiji nema mesta za element `violinski_ključ`. Neka od tih pravila mogu biti precizno definisana pomoću šeme napisane na jeziku kao što je W3C XML Schema Language, RELAX NG i DTD-ovi. Dokument može sadržati URL adresu šeme. Neki XML analizatori uočavaju taj podatak, pa tokom čitanja porede dokument s njegovom šemom i proveravaju da li dokument zadovoljava ograničenja navedena u njoj. Takav analizator nazivamo *analizator validnosti* (engl. *validation parser*). Kršenje tih ograničenja nazivamo *greška validnosti*, a postupak provere sadržaja dokumenta u odnosu na šemu je *validacija*. Ako analizator validnosti nađe grešku validnosti, prijaviće je aplikaciji u čije ime analizira dokument. Potom ta aplikacija odlučuje želi li da nastavi analiziranje dokumenta. Međutim, greške validnosti ne moraju biti fatalne (za razliku od grešaka u dobrom oblikovanju), pa aplikacija može odlučiti da ih zanemari. Nisu svi analizatori ujedno i analizatori validnosti – neki proveravaju samo dobru oblikovanost.

Aplikacija koja od analizatora prima podatke, može biti:

- čitač Weba poput Netscape Navigatora ili Internet Explorera, koji korisniku prikazuje dokument
- program za obradu teksta poput StarOffice Writera, koji XML dokument učitava radi obrade
- baza podataka kao što je Microsoftov SQL Server, koji XML podatke smešta u nov zapis
- program za crtanje kao što je Adobe Illustrator, koji XML interpretira kao dvodimenzionalne koordinate sadržaja crteža
- program za tabelarne proračune poput Gnumerica, koji analizira XML da bi pronašao brojeve i funkcije koje se upotrebljavaju u određenom proračunu
- program za lične finansije kao što je Microsoftov Money, koji XML vidi kao bankovni izveštaj
- program koji čita XML dokument i iz njega izdvaja naslove za dnevne vesti
- program koji ste sami napisali na Javi, C-u, Pythonu ili nekom drugom jeziku, i koji radi ono što vi hoćete da radi
- gotovo bilo šta drugo.

XML je *izuzetno* fleksibilan format podataka. Upotrebljava se za sve navedeno i u još mnogo situacija. To su samo primeri iz prakse. Teorijski se svi podaci koji se uopšte mogu smestiti u računar, mogu snimiti u formatu XML. Praktično je XML podesan za skladištenje i razmenu onih podataka koji se mogu lako pretvoriti u tekst. Nepodesan je jedino za digitalizovane podatke kao što su fotografije, snimci zvuka, video i druge veoma velike sekvence bitova.

Evolucija XML-a

XML je potomak jezika SGML (Standardized Generalized Markup Language). Jezik od kog je nastao SGML sedamdesetih godina izumili su Charles F. Goldfarb, Ed Mosher i Ray Lorie iz IBM-a, a razvijalo ga je više stotina ljudi širom sveta, sve

dok ga 1986. ISO nije prihvatio kao standard 8879. SGML je rešavao mnoge probleme koje rešava XML, i to na isti način na koji ih rešava XML. To je semantički i strukturiran jezik za označavanje tekstualnih dokumenata. SGML je izuzetno moćan, pa je bio prihvaćen u američkoj vojsci, državnim službama, avionskoj, kosmičkoj industriji i drugim oblastima kojima je trebao efikasan način rukovanja tehničkim dokumentima dugačkim desetine hiljada stranica.

Najveći uspeh SGML-a je HTML, koji je primenjeni SGML. Međutim, HTML je samo jedna od primena SGML-a. HTML ni izdaleka ne omogućava sve što SGML. Pošto autore dokumenata ograničava konačnim skupom oznaka za opisivanje Web stranica – a pri tom ih opisuje prvenstveno za potrebe prikazivanja – HTML je zapravo malo više od klasičnog jezika za označavanje, koji je prihvaćen u čitačima Weba. Nije podesan za upotrebu izvan oblasti izrade Web stranica. Na primer, HTML se ne može upotrebiti za razmenu podataka između nekompatibilnih baza podataka, niti za slanje ažuriranih kataloga proizvoda maloprodajnim objektima. HTML je namenjen za Web stranice i to radi veoma dobro, ali to je sve što on može.

SGML je bio očigledan izbor za druge aplikacije koje koriste Internet, a nisu proste Web stranice namenjene običnim posetiocima. Nedostatak SGML-a je njegova komplikovanost – on je izuzetno komplikovan. Službena specifikacija SGML-a sadrži preko 150 teško razumljivih stranica. Ona obuhvata više specijalnih slučajeva i malo verovatnih scenarija. Toliko je složen da gotovo nikada nije napravljen softver koji bi ga u potpunosti realizovao. Programi koji su realizovali ili koristili različite podskupove SGML-a često su bili nekompatibilni. Specijalna funkcionalnost jednog programa, koju su njegovi autori smatrali suštinskom, u drugom programu bila je smatrana suvišnom i izostavljena je.

Godine 1996, Jon Bosak, Tim Bray, C.M. Sperberg-McQueen, James Clark i nekolicina drugih počeli su rad na „laganoj“ verziji SGML-a, koja je zadržala najveći deo njegovih funkcionalnosti, ali izostavila one koje su se u prethodnih 20 godina iskustva pokazale redundantnim, teško ostvarivim, zbunjujućim za krajnje korisnike ili – prosto – nekorisnim. Rezultat je bio XML 1.0 objavljen februara 1998.; on je odmah postigao uspeh. Punim srcem su ga prihvatili mnogi programeri kojima je trebao strukturirani jezik za označavanje, ali se nisu mogli naterati da savladaju SGML-ovu složenost. XML je počeo da se upotrebljava u najrazličitijim oblastima – od sudskih dokumenata do uzgajališta svinja.

Međutim, XML 1.0 predstavljao je tek početak. Sledeći standard bio je Namespaces in XML (Prostori imena u XML-u), što je bio pokušaj da se oznake različitih XML aplikacija upotrebljavaju u istom dokumentu – i to bez sukoba. Tako je Web stranica o knjigama mogla imati jedan element `title` koji se odnosio na naslov stranice i drugi element `title` koji se odnosio na naslov knjige, a da se ta dva elementa ne sukobe.

Sledeći je bio Extensible Stylesheet Language (XSL), što je XML aplikacija za primenu stilova na XML dokumente, da bi se dobio oblik koji mogu prikazati čitači Weba. XSL se uskoro podelio na XSL Transformations (XSLT) i XSL Formatting Objects (XSL-FO). XSLT je postao jezik opšte namene za pretvaranje jednog XML dokumenta u drugi, bez obzira na to da li se radi o prikazivanju dokumenta u obliku Web stranice ili nekoj drugoj nameni. XSL-FO je XML aplikacija za opisivanje strukture stranica koje će biti odštampane na podlozi ili prikazane na Webu. Po mogućnostima i izražajnosti, ona se približila PostScriptu.

Međutim, XSL nije jedina mogućnost za predstavljanje XML dokumenata. Kada je bio izumljen XML, u HTML-u su se već upotrebljavali kaskadni opisi stilova (engl. *cascading style sheets*, CSS), i pokazalo se da oni prilično dobro odgovaraju i XML-u. Kada je nastao CSS Level 2, W3C je CSS izričito namenio za opis predstavljanja XML dokumenata. Za tu ulogu je, bez obzira na to da li se radi o štampi ili Webu, u svetu SGML-a od svog nastanka bio prihvaćen DSSSL (Document Style Sheet and Semantics Language).

XLink (Extensible Linking Language) počeo je definisanjem moćnih struktura za povezivanje XML dokumenata u hipertekstualnu mrežu; u poređenju s njim, HTML-ova oznaka `A` (koja služi istoj svrsi) podseća nas na reč „anemična". I XLink se podelio na dva zasebna standarda: XLink za opisivanje veza između dokumenata i XPointer za adresiranje pojedinačnih delova XML dokumenta. Tada je primećeno da XPointer i XSLT razvijaju prefinjene, ali nekompatibilne sintakse za isti posao: identifikovanje određenih elemenata XML dokumenata. Zato su adresni delovi obe specifikacije odvojeni i objedinjeni u trećoj specifikaciji nazvanoj XPath. Malo kasnije izdvojio se još jedan deo XLinka, XInclude, kao sintaksa za pravljenje složenih dokumenata objedinjavanjem pojedinačnih dokumenata i njihovih odlomaka.

Sledeći deo slagalice bio je jednoobrazni interfejs za pristupanje sadržaju XML dokumenta iz Java, JavaScript ili C++ programa. Najjednostavniji interfejs za programiranje aplikacija (API) samo je tretirao dokument kao objekat koji sadrži druge objekte. Štaviše, već se, i u W3C i izvan njega, radilo na definisanju takvog *objektnog modela dokumenta* (Document Object Model, DOM) za HTML. Nije bilo teško proširiti ga tako da obuhvati i XML.

Izvan W3C, David Megginson, Peter Murray-Rust i drugi članovi liste slanja *xml-dev*, uočili su da razni XML analizatori, iako kompatibilni u pogledu dokumenata koje mogu analizirati, nisu kompatibilni unutar njihovih API-ja. To je dovelo do razvoja interfejsa za programiranje Simple API for XML (SAX-a). Godine 2000. objavljen je SAX2, koji je doneo čistiji API, veće mogućnosti konfigurisanja i podršku za prostore imena.

Jedno od iznenađenja tokom evolucije XML-a bilo je to da su ga programeri više upotrebljavali za strukture slične zapisima, kao što su serijalizovani objekti i tabele baza podataka, nego za narativne strukture za koje se tradicionalno upotrebljavao SGML. DTD-ovi su veoma lepo radili za narativne strukture, ali su imali neka ograničenja u pogledu dokumenata sličnih zapisima, a upravo su njih programeri pravili. Konkretno, glavni problemi su bili nepostojanje tipova podataka i činjenica da sâm DTD nije XML dokument. Više kompanija i pojedinaca počelo je da radi na jezicima za opisivanje šema koji bi otklonili te nedostatke. Mnogi od tih predloga bili su podneseni konzorcijumu W3C; on je oformio radnu grupu koja je trebalo da objedini najbolje delove tih predloga i napravi nešto još bolje. Grupa je 2001. objavila verziju 1.0 jezika W3C XML Schema Language. Nažalost, pokazalo se da je jezik previše složen i težak. Stoga se nekoliko proizvođača softvera vratilo poslu i napravilo čistije i elegantnije jezike za šeme, kao što su RELAX NG i Schematron.

Posle svih tih događaja postalo je jasno da XML 1.0, XPath, SAX, DOM i W3C XML Schema Language imaju slične, ali ipak pomalo različite konceptualne modele strukture XML dokumenta. Primera radi, XPath i SAX smatraju `CDATA` odeljke običnom sintaksom, dok ih DOM tretira različito od čvorova (engl. *nodes*)

običnog teksta. Zato je W3C oformio radnu grupu XML Core Working Group, koja priprema Skup informacija o XML-u (XML Information Set) koji će biti zajednički za sve navedene standarde.

Kako se sve više XML dokumenata sve veće vrednosti prenosi preko Interneta, uočena je potreba da se te transakcije obezbede i da se proverava identitet njihovih učesnika. Pored upotrebe postojećih mehanizama kao što su SSL i HTTP, koji identitet proveravaju pomoću softvera ugrađenog u upotrebljene prenosne protokole, razvijeni su formati za obezbeđenje samih XML dokumenata tokom celog njihovog životnog veka, umesto samo tokom prenosa. Tako je nastao XML Encryption – standardna XML sintaksa za šifrovanje digitalnog sadržaja, uključujući tu i delove poverljivih XML dokumenata. Za proveru identiteta namenjen je XML Signature (XML potpis), zajednički IETF i W3C standard za digitalno potpisivanje sadržaja i ugradnju tih potpisa u XML dokumente. Pošto digitalno potpisivanje i algoritmi za šifrovanje rade s nizovima bajtova, a ne s modelima XML podataka, i XML Signature i XML Encryption zasnovani su na standardnom formatu za serijalizaciju, Canonical XML, koji uklanja sve nebitne razlike između dokumenata kao što su razmaci (beline) unutar oznaka i upotreba navodnika ili polunavodnika za razdvajanje vrednosti atributa.

Tokom svih ovih događaja, osnovna specifikacija XML 1.0 ostala je nepromenjena. Dodata nova funkcionalnost nije menjala tu osnovu nego ju je nadogradila. To je dokaz solidnog dizajna i snage XML-a. Međutim, sam XML 1.0 zasnovan je na standardu Unicode 2.0. Kako je Unicode nastavio da se razvija i da obuhvata nova pisma kao što su mongolsko, kambodžansko i burmansko, XML je počeo da zaostaje. Prvenstveno zato je početkom 2004. objavljen XML 1.1. Treba napomenuti da XML 1.1 nudi malo toga novog programerima koji koriste engleski, španski, japanski, kineski, arapski, ruski, francuski, nemački, holandski ili bilo koji drugi jezik koji je već Unicode 2.0 podržavao.

Nema sumnje da ćemo morati da izumimo još mnogo proširenja za XML. Čak se i ova bogata kolekcija specifikacija bavi samo tehnologijama koje leže u osnovi XML-a. Mnogo je već napravljeno i još brže se radi na XML aplikacijama, među kojima su SOAP, SVG, XHTML, MathML, Atom, XForms, WordprocessingML i hiljade drugih. XML se pokazao kao čvrst temelj za mnogo različitih tehnologija.

2

Osnove XML-a

U ovom poglavlju naučićete kako se pišu jednostavni XML dokumenti. Saznaćete da se XML dokument sastoji od tekstualnog sadržaja obeleženog tekstualnim oznakama kao što su `<SKU>`, `<identifikator_zapisa>` i `<autor>`, koje pomalo liče na HTML oznake. Međutim, HTML sadrži oko stotinu unapred definisanih oznaka koje opisuju formatiranje Web stranice. U XML-u pravite proizvoljan broj oznaka prema svom nahođenju. Štaviše, te oznake uglavnom opisuju tip sadržaja dokumenta, a ne njegove formate i strukturu. U XML-u se ne kaže da je nešto ispisano kurzivom ili reljefno ili polucrno, nego da je to knjiga ili biografija ili kalendar.

Premda je XML opušteniji od HTML-a kada se radi o oznakama koje dopušta, mnogo je stroži što se tiče njihovog položaja i načina na koji su napisane. Konkretno, svi XML dokumenti moraju biti dobro oblikovani. Pravila dobrog oblikovanja postavljaju ograničenja kao što su „Svaka početna oznaka mora imati odgovarajuću završnu oznaku“ i „Vrednosti atributa moraju biti napisane u navodnicima“. Ta pravila se ne smeju kršiti, pa je XML dokumente nešto teže napisati, ali ih je zato lakše analizirati; oni ipak omogućavaju gotovo neograničenu fleksibilnost izražavanja.

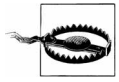
XML dokumenti i XML datoteke

XML dokument sadrži tekst. Pošto on nikada ne sadrži binarne podatke, može se otvoriti u svakom programu koji ume da čita tekstualne datoteke. Primer 2-1 sadrži otprilike najjednostavniji XML dokument koji se može zamisliti. Bez obzira na to, reč je o dobro oblikovanom XML dokumentu, pa ga XML analizatori mogu čitati i shvatiti (ukoliko se za računarski program može reći da nešto shvata).

Primer 2-1. Krajnje jednostavan, ali potpun XML dokument

```
<osoba>
   Alen Tjuring
</osoba>
```

U najčešćem scenariju, ovaj dokument bi predstavljao celokupan sadržaj datoteke nazvane *osoba.xml* ili možda *2-1.xml*. Međutim, XML nije sitničav u pogledu imena datoteka. Što se tiče analizatora, ova datoteka bi se mogla zvati *osoba.txt*, *osoba* ili *Ej ti, u ovoj datoteci je XML!* Možda se vašem operativnom sistemu takva imena ne bi svidela, ali se XML analizator neće buniti. Dokument uopšte ne mora biti smešten u datoteku – može i u zapis ili u polje baze podataka. Mogao ga je u letu generisati određeni CGI program kao odgovor na upit nekog čitača. S druge strane, mogao bi biti smešten u više datoteka, premda je to malo verovatno za tako jednostavan dokument. Ako se nalazi na Web serveru, verovatno će mu dodeliti MIME tip medija `application/xml` ili `text/xml`. Međutim, XML aplikacije bi mu mogle dodeliti određeniji MIME tip medija, kao što su `application/mathml+xml`, `application/xslt+xml`, `image/svg+xml`, `text/vnd.wap.wml`, pa čak i `text/html` (u veoma specijalnim slučajevima).



U generičkim XML dokumentima, prednost treba dati MIME tipu `application/xml` umesto `text/xml`, premda su mnogi Web serveri fabrički podešeni da koriste `text/xml`. Tip `text/xml` podrazumevano upotrebljava ASCII skup znakova, što je u većini XML dokumenata netačno.

Elementi, oznake i znakovni podaci

Dokument u primeru 2-1 sadrži samo jedan *element* nazvan `osoba`. Taj element je razgraničen *početnom oznakom* (engl. *start-tag*) `<osoba>` i *završnom oznakom* (engl. *end-tag*) `</osoba>`. Sve što se nalazi između početne i završne oznake elementa (isključujući njih) naziva se *sadržaj* (engl. *content*) elementa. Sadržaj pomenutog elementa je tekst:

```
Alen Tjuring
```

Razmaci (beline) predstavljaju deo sadržaja, mada ih mnoge aplikacije zanemaruju. *Markiranje* (engl. *markup*) dokumenta čine oznake `<osoba>` i `</osoba>`. Znakovni niz „Alen Tjuring“ i razmaci koji ga okružuju jesu *znakovni podaci* (engl. *character data*). Oznaka je najčešći oblik markiranja XML dokumenta, mada postoje i druge vrste koje ćemo razmotriti kasnije.

Sintaksa oznaka

XML oznake površno podsećaju na HTML oznake. Početna oznaka počinje znakom `<` a završava se znakom `>`; završna oznaka počinje znakom `</` a završava se sa `>`; između oznaka stoji ime elementa. Međutim, za razliku od HTML oznaka, nove XML oznake možete praviti tokom pisanja dokumenta. Da biste opisali osobu, upotrebite oznake `<osoba>` i `</osoba>`. Da biste opisali kalendar, upotrebite oznake `<kalendar>` i `</kalendar>`. Imena oznaka po pravilu odražavaju tip sadržaja unutar elementa, a ne način formatiranja tog sadržaja.

Prazni elementi

Postoji i specijalna sintaksa za prazne elemente, one koji nemaju sadržaja. Takav element može biti predstavljen jednom *oznakom praznog elementa* (engl. *empty-element tag*), koja počinje znakom `<`, a završava se znakovima `/>`. Na primer, u jeziku XHTML, što je „XML-izovana“ verzija standardnog HTML-a, elementi prekid reda

(engl. *line break*) i horizontalna linija (engl. *horizontal rule*) opisuju se oznakama `<br />` i `<hr />`. Te oznake su ekvivalentne parovima oznaka `<br></br>` odnosno `<hr></hr>`. Sami odlučite koji oblik oznaka ćete upotrebljavati za prazne elemente. U XML-u i XHTML-u (za razliku od HTML-a) ne možete upotrebiti samo početnu oznaku – na primer, `<br>` ili `<hr>` – a da ne upotrebite i odgovarajuću završnu oznaku. To bi bila greška u pogledu dobrog oblikovanja.

Uzimanje u obzir razlike između malih i velikih slova

Za razliku od HTML-a, XML uzima u obzir razliku između malih i velikih slova. Element `OSOBA` nije jednak elementu `osoba`, niti elementu `Osoba`. Ako element otvorite oznakom `<osoba>`, ne možete ga zatvoriti oznakom `</OSOBA>`. Možete upotrebljavati i mala i velika slova po svom izboru, ali svaki element morate dosledno ispisati.

XML stabla

Pogledajmo nešto složeniji XML dokument. U primeru 2-2 imamo element `osoba` koji sadrži više podesno označenih podataka, kako bi se videlo njihovo značenje.

Primer 2-2. Složeniji XML dokument koji opisuje određenu osobu

```
<osoba>
  <ime_i_prezime>
    <ime>Alen</ime>
    <prezime>Tjuring</prezime>
  </ime_i_prezime>
  <zanimanje>naučnik u oblasti računarstva</zanimanje>
  <zanimanje>matematičar</zanimanje>
  <zanimanje>kriptograf</zanimanje>
</osoba>
```

Roditelji i potomci

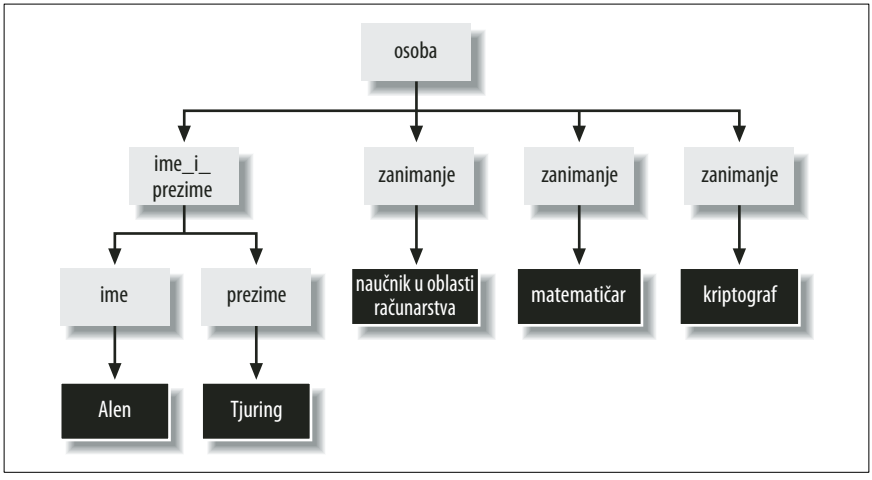
XML dokument u primeru 2-2 još uvek sadrži samo jedan element `osoba`. Međutim, sada taj element ne sadrži samo nediferencirane znakovne podatke. On ima četiri *elementa potomka* (engl. *child elements*): jedan element `ime_i_prezime` i tri elementa `zanimanje`. Element `ime_i_prezime` ima dva sopstvena elementa potomka, element `ime` i element `prezime`.

Za element `osoba` kažemo da je *roditelj* (roditeljski element) elementa `ime_i_prezime` i triju elemenata `zanimanje`. Element `ime_i_prezime` je roditelj elementa `ime` i elementa `prezime`. Katkada se kaže da su element `ime_i_prezime` i elementi `zanimanje` *braća* (engl. *siblings*) ili bratski elementi. Međusobno bratski su i elementi `ime` i `prezime`.

Kao i u ljudskom društvu, roditelji mogu imati više potomaka. Za razliku od ljudskog društva, XML svakom potomku daje tačno jednog, a ne dva ili više roditelja. Svakom elementu (uz jedan izuzetak koji ćemo uskoro objasniti) roditelj je tačno jedan element. Drugim rečima, svaki element se nalazi unutar određenog drugog elementa. Ako je početna oznaka elementa unutar određenog elementa, onda i njegova završna oznaka mora biti unutar istog elementa. U XML-u su zabranjene oznake koje se preklapaju, kao u `<strong><em>ovo je uobičajeno u HTML-u</strong></em>`. Pošto element `em` počinje unutar elementa `strong`, on se mora i završiti unutar istog elementa.

Korenski element

Svaki XML dokument ima jedan element koji nema roditelja. To je prvi element u dokumentu i element koji sadrži sve druge elemente. U primerima 2-1 i 2-2, tu ulogu je imao element `osoba`. Njega nazivamo *korenski element* (engl. *root element*) dokumenta. Katkada ga nazivaju i *element (celog) dokumenta* (engl. *document element*). Svaki dobro oblikovan XML dokument ima jedan korenski element. Pošto se elementi ne smeju preklapati, i pošto svi elementi osim korenskog imaju jednog roditelja, XML dokumenti tvore strukturu podataka koju programeri nazivaju *stablo* (engl. *tree*). Na slici 2-1 prikazan je taj odnos za primer 2-2. Svaki sivi pravougaonik predstavlja jedan element. Svaki crni pravougaonik predstavlja znakovne podatke. Svaka strelica predstavlja odnos sadržavanja.



Slika 2-1. Dijagram stabla za primer 2-2.

Mešovit sadržaj

U primeru 2-2, sadržaj elemenata `ime`, `prezime` i `zanimanje` bili su znakovni podaci; dakle, tekst bez ikakvih oznaka. Sadržaj elemenata `osoba` i `ime_i_prezime` bili su elementi potomci i razmaci (beline), koje većina aplikacija zanemaruje. Takva razlika između elemenata koji sadrže samo znakovne podatke i elemenata koji sadrže samo elemente potomke (i eventualno nekoliko razmaka – belina) uobičajena je u dokumentima sličnim zapisima. Međutim, XML se može upotrebiti i za narativne dokumente slobodnijeg oblika, kao što su poslovni izveštaji, članci za časopise, studentski eseji, kratke priče, Web stranice itd.; ilustraciju daje primer 2-3.

Primer 2-3. Narativno organizovan XML dokument

```
<biografija>
   <pasus>
   <ime_i_prezime><ime>Alen</ime> <prezime>Tjuring</prezime>
   </ime_i_prezime> bio je jedan od prvih ljudi koji su zaista zaslužili titulu
   <istaknuto>naučnika u oblasti računarstva</istaknuto>. Njegovi doprinosi
   toj oblasti previše su brojni da bismo ih ovde sve naveli,
   ali najpoznatiji su epohalni: <istaknuto>Tjuringov test</istaknuto>
```

Primer 2-3. Narativno organizovan XML dokument (nastavak)

```
   i <istaknuto>Tjuringova mašina</istaknuto>.
   </pasus>

   </definicija><termin>Tjuringova mašina</termin> je do dana današnjeg
   standardan test za utvrđivanje da li je neki računar zaista inteligentan.
   Još ga nijedan računar nije položio. </definicija>

   <definicija><termin>Tjuringova mašina</termin> je apstraktan automat
   koji se može nalaziti u konačno mnogo stanja, ima beskonačnu memoriju,
   i za njega se može dokazati da je ekvivalentan svakom drugom automatu
   s proizvoljno velikom memorijom, koji se može nalaziti u konačno mnogo stanja.
   Dakle, ono što važi za jednu Tjuringovu mašinu, važi za sve njih,
   bez obzira na to kako su napravljene.
   </definicija>

   <pasus>
   <ime_i_prezime><prezime>Tjuring</prezime></ime_i_prezime> je bio i perfektan
   <zanimanje>matematičar</zanimanje> i <zanimanje>kriptograf</zanimanje>.
   U savezničkom razbijanju nemačkog šifratora Enigma, njegov doprinos
   je bio odlučujući. Tjuring se ubio <datum><dan>7</dan>. <mesec>juna</mesec>
   <godina>1954</godina></datum>, nakon što je bio osuđen kao homoseksualac
   i prisiljen da prima injekcije ženskih hormona.
   </pasus>

</biografija>
```

Korenski element ovog dokumenta jeste `biografija`. Ta `biografija` ima elemente potomke `pasus` i `definicija`, kao i razmake (beline). I elementi `pasus` i `definicija` sadrže druge elemente: `termin`, `istaknuto`, `ime_i_prezime` i `zanimanje`, a i neke neoznačene znakovne podatke. Za elemente kao što su `pasus` i `definicija`, koji sadrže elemente potomke i znakovne podatke koji nisu samo razmaci (beline), kažemo da imaju *mešovit* sadržaj (engl. *mixed content*). Mešovit sadržaj je uobičajen u XML dokumentima u kojima su članci, eseji, priče, knjige, romani, izveštaji, Web stranice i sve drugo što je organizovano kao pisano pripovedanje. Mešovit sadržaj je ređi (a s njim je i teže raditi) u računarski generisanim i obrađenim XML dokumentima, koji se upotrebljavaju za razmenu baza podataka, serijalizaciju objekata, trajne formate datoteka itd. Jedna od prednosti XML-a je to da ga lako možete prilagoditi veoma različitim zahtevima računarski generisanih dokumenata i dokumenata koje su napisali ljudi.

Atributi

XML elementi mogu imati atribute. Atribut je par ime–vrednost pridruženo početnoj oznaci elementa. Imena atributa su od njihovih vrednosti razdvojena znakom jednakosti i opcionim razmakom. Vrednosti atributa moraju biti zatvorene u navodnike ili polunavodnike. Primera radi, sledeći element `osoba` ima atribut `rođena` čija je vrednost `1912-06-23` i atribut `umrla` čija je vrednost `1954-06-07`:

```
<osoba rođena="1912-06-23" umrla="1954-06-07">
  Alen Tjuring
</osoba>
```

Što se tiče XML analizatora, prethodni element je identičan sledećem. U njemu su umesto navodnika upotrebljeni polunavodnici, redosled atributa je drugačiji i dodato je nekoliko razmaka oko znaka jednakosti.

```
<osoba umrla = '1954-06-07' rođena = '1912-06-23'>
  Alen Tjuring
</osoba>
```

Razmake oko znaka jednakosti svako dodaje po svom nahođenju. Polunavodnici su korisni u slučajevima kada vrednost atributa sadrži navodnik. Redosled atributa nije važan.

U primeru 2-4 prikazano je kako se atributi mogu upotrebiti za kodiranje velikog dela istih podataka navedenih u dokumentu sličnom zapisu iz primera 2-2.

Primer 2-4. XML dokument koji osobu opisuje pomoću atributa

```
<osoba>
  <ime_i_prezime ime="Alen" prezime="Tjuring"/>
  <zanimanje vrednost="naučnik u oblasti računarstva"/>
  <zanimanje vrednost="matematičar"/>
  <zanimanje vrednost="kriptograf"/>
</osoba>
```

Ovime se otvara pitanje treba li i kada treba za čuvanje podataka upotrebljavati elemente potomke, a kada atribute. To je predmet žestoke rasprave. Neki informatičari tvrde da su atributi podesni za metapodatke oko elementa, dok su elementi za same podatke. Drugi ukazuju na to da nije uvek očigledno šta su podaci, a šta metapodaci. Zaista, odgovor može zavisiti od toga gde se podaci koriste.

Nesporno je da svaki element može imati samo jedan atribut datog imena. Malo je verovatno da će to postati problem za datum rođenja ili smrti; moglo bi da bude problem za zanimanje, ime, adresu i sve drugo od čega element može imati više primeraka. Nadalje, struktura atributa je sasvim ograničena. Vrednost atributa je prosto nediferenciran tekst. Podela datuma crticama na godinu, mesec i dan u prethodnim odlomcima koda predstavlja maksimalnu podstrukturu koja je prikladna za ugradnju u atribut. Mnogo je fleksibilnija i proširivija struktura zasnovana na elementima. Uprkos tome, za neke primene atributi su sigurno podesniji. Najzad, ako sami pišete svoj XML rečnik, na vama je da odlučite kada ćete upotrebiti element, a kada atribut.

Atributi su korisni i u narativnim dokumentima, kao što je pokazano u primeru 2-5. U njemu je možda nešto očiglednije šta pripada elementima, a šta atributima. Sirov narativni tekst predstavljen je u obliku znakovnih podataka unutar elemenata. Dodatne informacije koje objašnjavaju te podatke predstavljene su u obliku atributa. Među tim informacijama su reference izvora, URL adrese slika, hiperveze, i datumi rođenja i smrti. Međutim, čak i ovo se moglo uraditi drugačije. Na primer, brojevi fusnota mogu biti atributi elementa `fusnota`, a ne znakovni podaci.

Primer 2-5. Narativni dokument u kome su upotrebljeni atributi

```
<biografija xmlns:xlink="http://www.w3.org/1999/xlink/">

   <slika izvor="http://www.turing.org.uk/turing/pi1/busgroup.jpg"
   širina="152" visina="345"/>

   <pasus><osoba rođena='1912-06-23'
   umrla='1954-06-07'> <ime>Alen</ime>
   <prezime>Tjuring</prezime> </osoba> je bio jedan od prvih ljudi koji su
   zaista zaslužili titulu <istaknuto>naučnika u oblasti računarstva</istaknuto>.
   Njegovi doprinosi toj oblasti previše su brojni da bismo ih ovde sve naveli,
   ali najpoznatiji su epohalni:
   <istaknuto xlink:type="simple"
   xlink:href="http://cogsci.ucsd.edu/~asaygin/tt/ttest.html">Tjuringov
   test</istaknuto> i <istaknuto  xlink:type="simple"
   xlink:href="http://mathworld.wolfram.com/TuringMachine.html">Tjuringova
   mašina</istaknuto>.</pasus>

   <pasus><prezime>Tjuring</prezime> je bio i perfektan <zanimanje>matematičar
   </zanimanje> i <zanimanje>kriptograf</zanimanje>. U savezničkom razbijanju
   nemačkog šifratora Enigma, njegov doprinos bio je odlučujući.
   <fusnota izvor="The Ultra Secret, F.W. Winterbotham, 1974">1</fusnota></pasus>

   <pasus>
   <prezime>Tjuring</prezime>se ubio <datum><dan>7</dan>. <mesec>juna</mesec>
   <godina>1954</godina>.</datum> nakon što je bio osuđen kao homoseksualac
   i prisiljen da prima injekcije ženskih hormona.<fusnota izvor="Alan Turing:
   the Enigma, Andrew Hodges, 1983">2</fusnota>
   </pasus>
</biografija>
```

XML imena

Specifikacija XML-a ume da bude sitničava i izbirljiva što se tiče imena. Bez obzira na to, ona pokušava da bude efikasna kada je moguće. Jedan od načina da to postigne jeste, po mogućstvu, korišćenje istih pravila za različite stavke. Na primer, pravila za imena XML elemenata jednaka su pravilima za imena XML atributa, kao i za imena nekih ređe upotrebljavanih komponenata. Sve njih skupno nazivamo *XML imena* (engl. *XML names*).

U suštini, XML imena mogu sadržati sve alfanumeričke znakove. Među njima su standardna engleska slova od *A* do *Z* i od *a* do *z*, kao i cifre od *0* do *9*. XML imena mogu sadržati i neengleska slova, brojeve i ideograme, kao što su *ö*, *ç*, **Ω**, 串. U imenima se mogu koristiti i sledeća tri znaka interpunkcije:

_ donja crta

- crtica

. tačka

U XML imenima ne smeju se javljati drugi znakovi interpunkcije, kao što su navodnici, polunavodnici, znak za dolar, kapica (^), znak za procenat i tačka i zarez (;).

Dvotačka je dozvoljena, ali je rezervisana za prostore imena, što je objašnjeno u poglavlju 4. XML imena ne smeju sadržati beline bilo koje vrste, bez obzira na to da li se radi o razmaku, znaku za vraćanje na početak reda, znaku za prelazak u novi red, nelomivom razmaku itd. Najzad, sva imena koja počinju znakovnim nizom „XML“ (u svim kombinacijama malih i velikih slova) rezervisana su za standardizaciju u XML specifikacijama organizacije W3C.

Primarni novitet u XML-u 1.1 jeste da XML imena mogu sadržati samo znakove definisane u standardu Unicode 3.0 i njegovim novijim verzijama. XML 1.0 je ograničen na znakove definisane u standardu Unicode 2.0. XML 1.1 dozvoljava u imenima i znakove iz sledećih dodatnih pisama: burmanskog, mongolskog, Thaana, kambodžanskog, Yi i amharskog. (U XML-u 1.0 sva su ta pisma bila dozvoljena u tekstualnom sadržaju. Tada se nisu smela upotrebljavati za imena elemenata, atributa i entiteta.) XML 1.1 ne pruža gotovo ništa novo programerima koji u svojim oznakama ne upotrebljavaju navedena pisma.

XML 1.1 dozvoljava da imena sadrže i neuobičajene simbole, kao što je muzički simbol za šestostruno cimbalo, pa čak i približno milion kodova kojima još nisu dodeljeni znakovi. Međutim, korišćenje tih simbola u imenima bilo bi veoma nerazumno. Toplo vam preporučujemo da, čak i u XML-u 1.1, imena ograničite na slova, cifre, ideograme i izričito dozvoljene ASCII znakove interpunkcije.

XML imena smeju da počnu isključivo slovom, ideogramom ili znakom donja crta. Ne mogu početi cifrom, crticom niti tačkom. Dužina XML imena nije ograničena. Zato su sledeći elementi dobro oblikovani:

- `<Drivers_License_Number>98 NY 32</Drivers_License_Number>`
- `<month-day-year>7/23/2001</month-day-year>`
- `<ime_i_prezime>Alen Tjuring</ime_i_prezime >`
- `<_4-lane>I-610</_4-lane>`
- `<téléphone>011 33 91 55 27 55 27</téléphone>`
- `<персна>ГаЛинаоНванов</персна>`

Sledeći elementi su neprihvatljivi:

- `<Driver's_License_Number>98 NY 32</Driver's_License_Number>`
- `<month/day/year>7/23/2001</month/day/year>`
- `<ime i prezime>Alen Tjuring</ime i prezime>`
- `<4-lane>I-610</4-lane>`

Reference

Znakovni podaci unutar elementa ne smeju sadržati znak za manje od (`<`) koji nema odgovarajuću izlaznu (engl. *escape*) sekvencu (`</`). Znak `<` uvek se tumači kao početak oznake. Ukoliko vam zatreba u tekstu, pretvorite ga u izlaznu sekvencu pomoću *reference entiteta* (engl. *entity reference*) `<`, *numeričke reference znaka* (engl. *numeric*

character reference) `&#60;` ili *heksadecimalne numeričke reference znaka* (engl. *hexadecimal numeric character reference*) `&#x3C;`. Kada analizator bude čitao dokument, zameniće sve reference `<`, `&#60` ili `&#x3C` znakom <, a neće se zbuniti i protumačiti < kao početak nove oznake. Na primer:

```
<SCRIPT LANGUAGE="JavaScript">
   if (location.host.toLowerCase().indexOf("ibiblio") &lt; 0)  {
       location.href="http//&&ibiblio.org/xml/"<;
   }
</SCRIPT>
```

Znakovni podaci ne smeju sadržati ni sirov znak ampersend (&) koji nema svoju izlaznu sekvencu. Taj znak se uvek tumači kao početak reference entiteta. Ovako se ampersend referencom entiteta `&` pretvara u izlaznu sekvencu:

```
<company>W.L. Gore &amp; Associates</company>
```

Pošto je znak ampersend u Unicode sistemu definisan kao kôd 38, mogli smo upotrebiti i numeričku referencu znaka, `&#38;`:

```
<company>W.L. Gore &#38; Associates</company>
```

Reference entiteta kao što je `&` i reference znakova kao što je `&#60;` spadaju u markiranja. Kada aplikacija raščlanjuje i analizira XML dokument, ta markiranja se zamenjuju znakom ili znakovima na koje referenca upućuje. XML unapred definiše pet referenci entiteta. To su:

`<`
: Znak „manje od“, poznat i kao otvorena ugaona zagrada (<)

`&`
: Ampersend (&)

`>`
: Znak „veće od“, poznat i kao zatvorena ugaona zagrada (>)

`&quot;`
: Ravan navodnik (")

`&apos;`
: Ravan polunavodnik, poznat i kao apostrof (')

U sadržaju elemenata, samo se `<` i `&` moraju upotrebljavati umesto doslovno napisanih znakova < i &. Ostale reference su opcione. `&quot;` i `&apos;` su korisne unutar vrednosti atributa, gde bi sirov " ili ' mogao biti pogrešno protumačen kao završetak vrednosti atributa. Na primer, u sledećoj XML oznaci slike upotrebljena je referenca entiteta `&apos;` da bi se ubacio polunavodnik u ime O'Reilly:

```
<slika izvor='oreilly_koala3.gif' širina='122' visina='66'
          alt='Powered by O&apos;Reilly Books'
/>
```

Iako se znak „veće od“ ne može pogrešno protumačiti kao da završava oznaku koju nije trebalo da zatvori, referenca `>` je dozvoljena najviše zbog simetrije s referencom `<`.

Postoji jedan neobičan slučaj kada znak „veće od“ zaista morate pretvoriti u izlaznu sekvencu. U znakovnim podacima ne sme se pojaviti niz `]]>`. Umesto njega morate pisati `]]>`.

Pored pet unapred definisanih referenci entiteta, možete koristiti i druge koje ste sami naveli u definiciji tipa dokumenta. U poglavlju 3 objasnićemo kako se to radi.

Reference entiteta i znakova možete upotrebljavati samo u sadržaju elemenata i vrednostima atributa. Ne smete ih koristiti u imenima elemenata, imenima atributa, niti u drugim vrstama markiranja. Tekst kao što je `&` ili `&#60;` može se pojaviti unutar komentara ili instrukcije za obradu. Međutim, na tim mestima reference neće biti razrešene. Analizator zamenjuje samo reference koje pronađe u sadržaju elemenata i vrednostima atributa. On ne prepoznaje reference na drugim mestima.

Odeljci CDATA

Kada XML dokument sadrži uzorke XML ili HTML izvornog koda, znakovi `<` i `&` u tim uzorcima moraju biti napisani kao `<` odnosno `&`. Što više odeljaka s doslovno navedenim kodom dokument sadrži i što su oni duži, to kodiranje postaje zamornije. Umesto da se gnjavite s tim, svaki uzorak doslovno navedenog koda možete zatvoriti u *odeljak CDATA* (engl. *CDATA section*). Odeljak `CDATA` razgraničavate znakovima `<![CDATA[` i `]]>`. Sve što se zatekne između graničnika `<![CDATA[` i `]]>` tretira se kao sirov znakovni podatak. Zato u odeljku `CDATA` znak `<` ne otvara XML oznaku, znak `&` ne započinje referencu entiteta itd. Svi znakovi unutar odeljka `CDATA` tumače se kao obični znakovni podaci, a ne kao delovi markiranja.

Primera radi, u udžbeniku za Scalable Vector Graphics (SVG), napisanom na XHTML-u, možete naići na ovako nešto:

```
<p>You can use a default <code>xmlns</code> attribute to avoid
having to add the svg prefix to all your elements:</p>
<pre><![CDATA[
       <svg xmlns="http://www.w3.org/2000/svg"
            width="12cm" height="10cm">
         <ellipse rx="110" ry="130"/>
         <rect x="4cm" y="1cm" width="3cm" height="6cm" />
       </svg>
     ]]></pre>
```

SVG izvorni kôd uključen je neposredno u XHTML datoteku, a da nismo morali pažljivo da zamenimo svaki znak `<` referencom `<`. Tako smo dobili SVG dokument, a ne ugrađenu SVG sliku, što bi se u ovom primeru desilo da SVG kôd nismo smestili u odeljak `CDATA`.

Jedino što se u odeljku `CDATA` ne sme pojaviti jeste završni graničnik odeljka `CDATA`, `]]>` .

Odeljci `CDATA` postoje kao pogodnost za programere, a ne za programe. Analizatori vas ne moraju obavestiti da li je određen blok teksta potekao iz odeljka `CDATA`, od normalnih znakovnih podataka ili od znakovnih podataka koji sadrže reference entiteta kao što su `<` i `&`. Kada vam podaci postanu dostupni, te razlike će već biti izbrisane. Kôd koji pišete ne sme zavisiti od tih razlika.

Komentari

XML dokumenti mogu sadržati komentare, tako da koautori mogu ostavljati napomene jedni drugima i sebi, dokumentujući zašto su uradili to što su uradili i šta je još ostalo da se uradi. XML komentari su sintaktički jednaki HTML komentarima. Kao u HTML-u, komentari počinju sa `<!--` i završavaju se prvim primerkom niza `-->`. Na primer:

```
<!-- Ove hiperveze treba da proverim i ažuriram kada stignem. -->
```

Dve crtice se ne smeju pojaviti unutar komentara pre završnog `-->`. Izričito je zabranjen završetak komentara s tri crtice, `--->`.

Komentare možete stavljati bilo gde unutar znakovnih podataka u dokumentu. Možete ih stavljati pre i posle korenskog elementa. (Komentari nisu elementi, pa se time ne krši XML pravilo o strukturi stabla niti ono o jedinstvenom korenskom elementu.) Međutim, komentari se ne smeju pojaviti unutar oznake ili drugog komentara.

Aplikacije koje čitaju i obrađuju XML dokumente mogu, ali ne moraju proslediti informacije iz komentara. Svaka aplikacija slobodno može da ispusti sve komentare, ako se tako sviđa njenom autoru. Ne pišite dokumente ili aplikacije koji zavise od dostupnosti komentara. Oni služe isključivo tome da sirov izvorni kôd XML dokumenta učine razumljivijim ljudima. Komentari nisu namenjeni računarskim programima. U te svrhe upotrebite *instrukciju za obradu* (engl. *processing instruction*).

Instrukcije za obradu

U HTML-u se komentari ponekad zloupotrebljavaju za podršku nestandardnim proširenjima. Primera radi, sadržaj elementa `script` katkada se zatvara u komentar da bi se zaštitio od prikazivanja u čitaču koji ne ume da radi sa skriptovima. Web server Apache raščlanjuje i analizira komentare u *.shtml* datotekama da bi prepoznao datoteke za umetanje na serverskoj strani. Nažalost, ti dokumenti, nakon obrade u raznim HTML editorima, ponekad ne prežive s netaknutim komentarima i njima pridruženom semantikom. Što je još gore, bezazlen komentar može biti pogrešno protumačen kao ulaz u aplikaciju.

Kao alternativno sredstvo prosleđivanja informacija određenim aplikacijama koje će čitati dokument, XML ima *instrukciju za obradu*. Instrukcija za obradu počinje sa `<?`, a završava sa `?>`. Neposredno posle `<?` dolazi XML ime *cilja*, što može biti ime aplikacije kojoj je ta instrukcija namenjena ili identifikator instrukcije. Ostatak instrukcije za obradu sadrži tekst u formatu koji odgovara ciljnim aplikacijama.

Na primer, u HTML-u se robotskom oznakom `META` saopštava pretraživačima i ostalim robotima treba li i kako da indeksiraju stranicu. Za XML dokumente predložena je kao ekvivalentna sledeća instrukcija za obradu:

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

Cilj ove instrukcije za obradu je `robots`. Sintaktički, ova instrukcija sadrži dva pseudoatributa, jedan nazvan `index`, a drugi `follow`. Njihove vrednosti su `yes` ili `no`. Semantika ove instrukcije je sledeća: ako atribut `index` ima vrednost `yes`, onda bi

roboti pretraživača trebalo da indeksiraju stranicu. Slično tome, ako atribut `follow` ima vrednost `yes`, onda će roboti pretraživača posetiti lokacije na koje upućuju hiperveze stranice; ukoliko ima vrednost `no`, to neće biti učinjeno.

Druge instrukcije za obradu mogu imati potpuno različite sintakse i semantike. Primera radi, instrukcije za obradu mogu sadržati praktično neograničenu količinu teksta. PHP smešta velike programe u instrukcije za obradu. Na primer:

```
<?php
  mysql_connect("database.unc.edu", "clerk", "password");
  $result = mysql("HR", "SELECT LastName, FirstName FROM Employees
    ORDER BY LastName, Firstname");
  $i = 0;
  while ($i < mysql_numrows ($result))   {
     $fields = mysql_fetch_row($result);
     echo "<person>$fields[1] $fields[0] </person>\r\n";
     $i++;
  }
  mysql_close( );
?>
```

Instrukcije za obradu spadaju u označavanje, ali nisu elementi. Stoga ih možete pisati na bilo kom mestu u XML dokumentu osim unutar oznaka, kao komentare. Smete ih pisati i pre i posle korenskog elementa. Najčešća instrukcija za obradu, `xml-stylesheet`, dokumentu pridružuje opis stilova. Ona se uvek piše pre korenskog elementa, kao u primeru 2-6. U njemu instrukcija za obradu `xml-stylesheet` saopštava čitaču da na dokument, pre nego što ga prikaže čitaocu, primeni CSS opis stilova *osoba.css*.

Primer 2-6. XML dokument sa instrukcijom za obradu smeštenom u prolog

```
<?xml-stylesheet href="osoba.css" type="text/css"?>

<osoba>
   Alen Tjuring
</osoba>
```

Da bi se izbegla zabuna s deklaracijom XML-a, zabranjene su instrukcije za obradu nazvane `xml`, `XML`, `XmL` itd., u bilo kojoj kombinaciji malih i velikih slova. Instrukcijama za obradu možete davati sva druga imena koja zadovoljavaju pravila XML-a.

Deklaracija XML-a

XML dokumenti bi trebalo da otpočnu deklaracijom XML-a (ali ne moraju). Deklaracija XML-a izgleda kao instrukcija za obradu nazvana `xml`, koja sadrži pseudoatribute `version`, `standalone` i `encoding`. Strogo uzev, to nije instrukcija za obradu, nego deklaracija XML-a – ni više ni manje od toga. Ilustraciju daje primer 2-7.

Primer 2-7. Veoma jednostavan XML dokument s deklaracijom XML-a

```
<?xml version="1.0" encoding="ASCII" standalone="yes"?>
<osoba>
   Alen Tjuring
</osoba>
```

XML dokumenti ne moraju imati deklaraciju XML-a, ali ako je imaju, ona mora biti prva stavka dokumenta. Pre nje ne sme biti komentara, razmaka (belina), instrukcija za obradu itd. Razlog za to je što XML analizator na osnovu prvih pet znakova (`<?xml`) zaključuje kakvo je kodiranje znakova u dokumentu – recimo, da li je upotrebljen jednobajtni ili višebajtni skup znakova. Pre deklaracije XML-a sme biti samo nevidljiva Unicode oznaka redosleda bajtova. Razmotrićemo to ponovo u poglavlju 5.

Atribut version

Atribut `version` bi trebalo da ima vrednost 1.0. Pod veoma neuobičajenim okolnostima možete mu dati vrednost 1.1. Pošto zadavanje verzije 1.1 ograničava dokument na najnovije verzije malog broja analizatora, a svi analizatori za XML 1.1 moraju podržavati i XML 1.0, ne bi trebalo da olako zadajete verziju 1.1.

Ne verujete? Najpre odgovorite na nekoliko pitanja:

1. Govorite li burmanski, mongolski, kambodžanski, amharski ili divehi?
2. Sadrže li vaši podaci zastarele, netekstualne C0 kontrolne znakove kao što su vertikalni tabulator, prelazak na novu stranicu ili znak za zvonce?

Ukoliko ste na oba pitanja odgovorili „ne“, korišćenjem XML-a 1.1 ne dobijate apsolutno ništa. Ako ste na jedno pitanje odgovorili „da“, možda imate razloga za korišćenje XML-a 1.1. XML 1.0 dozvoljava da se burmanski, mongolski, kambodžanski itd. upotrebljavaju u znakovnim podacima i vrednostima atributa. XML 1.1 dozvoljava da se ta pisma upotrebljavaju i u imenima elemenata i atributa, što XML 1.0 ne dopušta. XML 1.1 dozvoljava i da se C0 kontrolni znakovi (sem znaka null) upotrebljavaju u znakovnim podacima i vrednostima atributa (ukoliko su pretvoreni u numeričke reference znakova poput `&#x07;`), što XML 1.0 ne dopušta. Ako ijedan od ova dva uslova važi za vas, mogli biste upotrebiti XML 1.1 (premda bi trebalo da budete svesni da time znatno sužavate potencijalnu publiku svog XML dokumenta). U protivnom, trebalo bi da upotrebljavate isključivo XML 1.0.

Atribut encoding

Dosad smo bili neodređeni u pogledu skupova znakova i kodiranja znakova. Rekli smo da su XML dokumenti sastavljeni od čistog teksta, ali nismo rekli kako su znakovi tog teksta kodirani. Po standardu ASCII? Latin-1? Unicode? Nekom četvrtom?

Kratak odgovor na ovo pitanje je „da“. Dugačak odgovor je da su XML dokumenti podrazumevano kodirani UTF-8 kodovima promenljive dužine u skupu znakova Unicode. Pošto se radi o nadskupu skupa znakova ASCII, tekstualne datoteke napisane u čistom ASCII-ju automatski su kodirane i po standardu UTF-8. Međutim, većina programa za obradu XML-a (naročito oni napisani na Javi) mogu obrađivati mnogo veći broj skupova znakova. Analizatoru treba reći samo koji je standard za kodiranje znakova upotrebljen u dokumentu. To je najbolje uraditi preko metapodataka, koji su smešteni u sistem datoteka ili ih daje server. Međutim, ne pružaju svi sistemi podatke o skupu znakova, pa XML dozvoljava dokumentima da sami naznače svoj skup znakova pomoću *deklaracije kodiranja* (engl. *encoding declaration*) unutar deklaracije XML-a. U primeru 2-8 pokazano je kako biste naznačili da je dokument napisan u skupu znakova ISO-8859-1 (Latin-1), koji obuhvata i znakove kao što su ö i ç, potrebne u mnogim zapadnoevropskim jezicima.

Primer 2-8. XML dokument kodiran u skupu znakova Latin-1

```
>?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<osoba>
   Erwin Schrödinger
>/osoba>
```


U deklaraciji XML-a atribut `encoding` je opcion. Ako se on izostavi, a metapodaci su nedostupni, analizator pretpostavlja da je upotrebljen skup znakova Unicode. Na osnovu prvih nekoliko bajtova datoteke, analizator može pokušati da utvrdi koje je Unicode kodiranje upotrebljeno. Ako su metapodaci dostupni, ali su suprotni deklaraciji o kodiranju, onda analizator veruje metapodacima. Na primer, ukoliko HTTP čitač kaže da je dokument kodiran u ASCII-ju, a deklaracija o kodiranju kaže da je kodiran po standardu UTF-8, analizator će izabrati ASCII.

Razna kodiranja i pravilno rukovanje neengleskim XML dokumentima razmotrićemo detaljnije u poglavlju 5.

Atribut standalone

Ukoliko atribut `standalone` (samostalan) ima vrednost `no`, onda aplikacija može učitati spoljni DTD (tj. DTD smešten u neku drugu datoteku, a ne u onu koja sadrži dokument) da bi utvrdila prave vrednosti određenih delova dokumenta. Primera radi, DTD može sadržati podrazumevane vrednosti atributa koje analizator treba da prijavi, premda one u dokumentu ne postoje.

Dokument koji nema DTD, a takvi su svi dokumenti u ovom poglavlju, može imati `yes` kao vrednost atributa `standalone`. I dokument koji ima DTD može imati `yes` kao vrednost atributa `standalone`, ukoliko taj DTD ni na koji način ne menja sadržaj dokumenta ili je DTD potpuno interni. Pojedinosti o dokumentima s DTD-ovima objašnjene su u poglavlju 3.

U deklaraciji XML-a, atribut `standalone` je neobavezan. Ako je izostavljen, pretpostavlja se da ima vrednost `no`.

Provera dobre oblikovanosti dokumenta

Svaki XML dokument, bez izuzetka, mora biti dobro oblikovan. To znači da mora zadovoljiti više pravila, među kojima i sledeća:

1. Svaka početna oznaka mora imati odgovarajuću završnu oznaku.
2. Elementi mogu biti ugnežđeni, ali se ne smeju preklapati.
3. Mora postojati tačno jedan korenski element.
4. Vrednosti atributa moraju biti zatvorene u navodnike.
5. Element ne sme imati dva istoimena atributa.
6. Unutar oznaka ne sme biti komentara i instrukcija za obradu.
7. U znakovnim podacima elemenata i atributa ne sme biti znakova `<` ili `&` koji nisu pretvoreni u izlaznu sekvencu.

Ovo nije celokupna lista pravila. Dokument može biti loše oblikovan na veoma mnogo načina. Celokupna lista pravila navedena je u poglavlju 21. Neka od njih se odnose na strukture koje još nismo razmatrali, kao što su DTD-ovi. Druga pravila ćete veoma retko kršiti ako budete sledili primere iz ovog poglavlja (na primer, nemojte stavljati razmak između početnog < i imena elementa u oznaci).

Bez obzira na to da li je greška mala ili velika i da li se sreće retko ili često, XML analizator koji čita dokument mora da je prijavi. Analizator može, ali ne mora da prijavi sve greške u dobrom oblikovanju koje pronađe u dokumentu. Međutim, analizator ne sme pokušati da popravi dokument i da doda ono što smatra da je autor dokumenta izostavio. Ne sme dodati izostavljene navodnike oko vrednosti atributa ili izostavljenu završnu oznaku, niti sme zanemariti komentar napisan unutar početne oznake. Analizator mora prijaviti grešku. Tako je napravljeno da bi se izbegli ratovi kompatibilnosti („vaš čitač prijavljuje 5 naših grešaka, pa će i naš čitač prijavljivati 5 vaših grešaka"), koji čitače Weba prate od njihovih početaka do danas. Zato proverite da li je vaš XML dobro oblikovan pre nego što ga objavite, bez obzira na to da li se radi o Web stranici, ulazu u bazu podataka ili nečem trećem.

Najjednostavniji način da to uradite jeste da učitate dokument u čitač Weba koji ume da radi s XML dokumentima, kao što je Mozilla. Ukoliko je dokument dobro oblikovan, čitač će ga prikazati. Ako nije, prikazaće poruku o grešci.

Umesto učitavanja dokumenta u čitač Weba, možete neposredno upotrebiti XML analizator. Većina XML analizatora nije namenjena krajnjim korisnicima. Oni su zapravo biblioteke klasa projektovane za ugradnju u program koji se lakše koristi, kao što je Mozilla. Njihov interfejs komandne linije je minimalan, ako uopšte postoji; često nije dobro dokumentovan. Bez obzira na sve to, ponekad je brže provući grupu datoteka kroz interfejs komandne linije nego ih jednu po jednu učitavati u čitač Weba. Nadalje, kada naučite da radite s DTD-ovima i šemama, iste alatke moći ćete da upotrebljavate za proveru validnosti dokumenata, što većina čitača Weba ne radi.

Postoji mnogo XML analizatora dostupnih u raznim jezicima. Ovde ćemo prikazati proveru dobre oblikovanosti analizatorom *libxml* kompanije Gnome Project, koji možete preuzeti na lokaciji *http://xmlsoft.org*. Taj paket otvorenog izvornog koda napisan je na prilično prenosivom C-u i radi na većini glavnih platformi, uključujući Windows, Linux i Mac OS X. (Unapred je instaliran u mnogim distribucijama Linuxa.) Postupak bi trebalo da bude sličan i s drugim analizatorima, premda se pojedinosti mogu razlikovati.

libxml je zapravo biblioteka, ali sadrži i program *xmllint* koji pomoću te biblioteke proverava dobru oblikovanost datoteka. *xmllint* se pokreće pisanjem njegovog imena u Unixovom komandnom okruženju (engl. *Unix shell*) ili posle DOS-ovog odzivnika (engl. *DOS prompt*), kao i svaki drugi program koji ima komandnu liniju. Njegovi argumenti su URL adrese ili imena datoteka dokumenta koji treba proveriti. Evo rezultata koje je *xmllint* dao za jednu od ranijih verzija primera 2-5. Već prvi red ispisa saopštava gde je prvi problem u datoteci:

```
% xmllint 2-5.xml
2-5.xml:5: error: Unescaped '<' not allowed in attribute values
  <osoba rođena='1912/06/23'
  ^
2-5.xml:5: error: attributes construct error
  <osoba rođena='1912/06/23'
  ^
```

```
2-5.xml:5: error: error parsing attribute name
  <osoba rođena='1912/06/23'
  ^
2-5.xml:5: error: attributes construct error
  <osoba rođena='1912/06/23'
  ^
2-5.xml:5: error:xmlParseStartTag: problem parsing attributes
  <osoba rođena='1912/06/23'
  ^
2-5.xml:5: error: Coludn't find end of Start Tag image line 3
  <osoba rođena='1912/06/23'
  ^
```


Kao što vidite, analizator je pronašao grešku. U ovom slučaju, poruka o grešci nam nije naročito pomogla. Stvarni problem nije bio u tome što je vrednost atributa sadržala znak <, nego u tome što je izostavljen završni navodnik u vrednosti atributa `visina`. Ipak smo pomoću datih podataka uspeli da pronađemo i otklonimo problem. Uprkos dugačkom ispisu, *xmllint* je prijavio samo prvu grešku u dokumentu, pa ćete morati da ga pokrećete više puta dok ne pronađete i ne ispravite sve greške. Kada smo primer 2-5 popravili tako da bude dobro oblikovan, *xmllint* je samo odštampao datoteku koju je pročitao:

```
% xmllint 2-5.xml
<biografija xmlns:xlink="http://www.w3.org/1999/xlink/">

   <slika izvor="http://www.turing.org.uk/turing/pi1/busgroup.jpg"
   širina="152" visina="345"/>

   <pasus><osoba rođena='1912-06-23'
   umrla='1954-06-07'> <ime>Alen</ime>
...
```

Pošto je dokument ispravljen i dobro oblikovan, možete ga proslediti čitaču Weba, bazi podataka ili nekom drugom programu koji ga očekuje. Gotovo svi netrivijalni ručno pisani dokumenti na početku nisu dobro oblikovani, pa je važno da proverite šta ste uradili pre nego što to objavite.