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.