# Škola XML-a *1/3*

## Uvod

Do relativno skoro, XML (Extensible Markup Language – proširiv jezik za označavanje) se koristio uglavnom u rezimeu kada ste hteli da ostavite dobar utisak. Od nekakve primene jedva da je bilo traga. Tako se kod nas u knjižarama i danas mogu kupiti knjige gde doslovno piše da je XML tehnologija budućnosti (retki su autori koji su se usudili da dodaju "bliske"). Međutim XML pomalo iznenađuje brzinom prodora iako kompjuterska zajednica ne važi baš za konzervativnu sredinu. Do pre otprilike godinu dana učenje XML-a je ličilo na investiciju za budućnost a danas je to uslov za ozbiljnije bavljenje tematikom koja je u trendu. Ovo nikako ne znači da je XML prolazna moda – nasuprot tome XML je potreba koja će u budućnosti biti sve veća. Naime, kako raste www i njegov sadržaj tako rastu i naše potrebe za informacijama (ili barem potrebe za brzinom dobijanja istih). Pored toga, kako napreduje hardverska infrastruktura tako se menja i pogled na internet. Internet sve manje doživljavamo kao carstvo šarenih reklama a sve više kao medijum za ozbiljno poslovanje. Informacija na sajtu više nije nekakav statičan sadržaj već je uglavnom živa informacija - posledica upita nad bazom a internet polako dobija atribute mrežne infrastrukture koja liči na one tipa LAN. Ono što je nekada predstavljao HTTP za razvoj internet danas je XML za moderno poslovanje putem interneta posebno u segmentu B2B (Buisness to buisness – tipično veliki informacioni sistemi).

## Šta je XML

XML je na prvom mestu tehnologija. Nakon toga dolazi definicija da je XML jezik mada je XML manje jezik a više konvencija za kodiranje. XML je i skup srodnih tehnologija i sam za sebe ne predstavlja nešto posebno već tek sa srodnim tehnologijama daje pune rezultate. Srodne tehnologije su:

- DTD
- CSS
- XLS
- DOM
- ADO
- XLink
- XFragments
- XPointer
- ... itd.

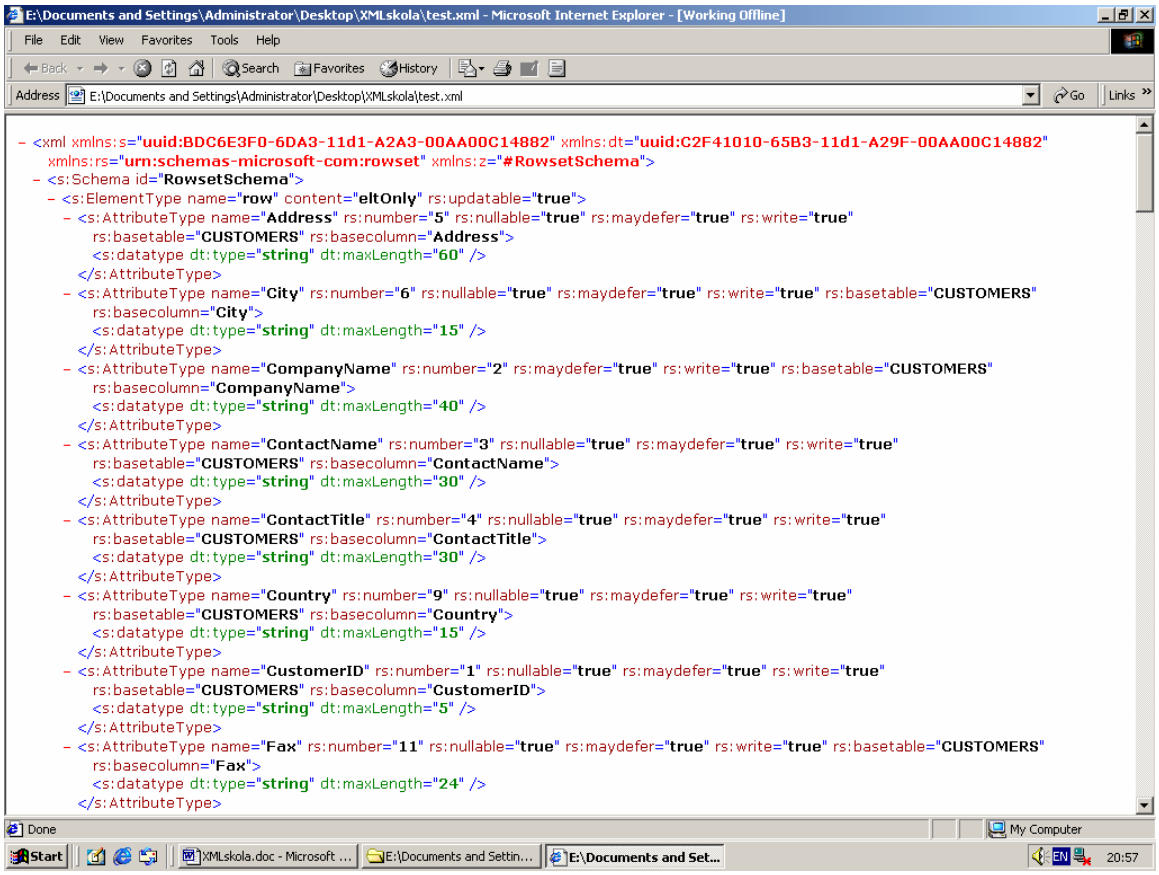
U nekim od pomenutih tehnologija XML se ponaša kao klijent dok je za neke server a može biti i jedno i drugo istovremeno.
Jedan od najvećih problema kada je u pitanju transfer informacija je njihov sadržaj u logičkom smislu te reči. XML služi kao konterner za transfer jer u sebi pored informacije ima i njenu poziciju u odnosu na ostale informacije – XML pored informacije opisuje i strukturu.
Detaljna specifikacija XML-a je data na www.w3.org i svakako je vredi pročitati.

# Pregled XML-a i njegove osobine

XML je u svojoj osnovi informacija o informaciji. Više nije dovoljno imati samo informaciju jer to u današnjim uslovima znači tražiti i pronaći istu već je potreban način da opišemo informaciju a da taj opis informacije upotrebimo dalje za pronalaženje iste i za njenu dalju obradu. Zamislite sledeću situaciju: potreban vam je tačan kurs DEM u odnosu na USD. Možete otići na neki sajt i tamo pronaći tu informaciju ali to zahteva da neka osoba to i uradi. Možda biste mogli i da napišete neki parser koji će sadržaj tog sajta rasčlaniti i pronaći potrebnu informaciju ali zamislite šta vam je sve potrebno da znate da napišete takav parser. Morate znati gde je i kako je ta informacija smeštena. Prilično komplikovano a rezultati su i dalje diskutabilni. Međutim kada biste imali informaciju koja opisuje informaciju stvari bi bile mnogo lakše. Lako biste mogli da dođete do potrebne informacije – to bi mašina mogla da uradi umesto vas a kad neko stalno nudi takve informacije kojima biste mogli da pristupite lako onda je pojam *web servisa* lako razumljiv. Ideja se dalje razvija i dobijamo internet kao mrežnu infrastrukturu tako da na kraju krajeva internet ili barem jedan njegov segment će biti uskoro čisto poslovna mreža kojom će se razmenjivati samo informacije u sirovom obliku. Uslov te razmene je da informacije na neki način budu obeležene da bi se mogle identifikovati i koristiti a upravo to je ono što XML nudi.

XML je u osnovi tekst. Doduše nije formatiran tako da lepo izgleda a može biti i komplikovan tako da ga to potencijalno diskvalifikuje za čitanje od strane ljudi. Kao ilustarcija XML, ovako izgleda ADO rekordset (SELECT * FROM Customers - legendarna NorthWind baza) u XML-u parsiran u Internet Exploreru 5.5:

*Slika 1. ADO rekordset u XML-u*

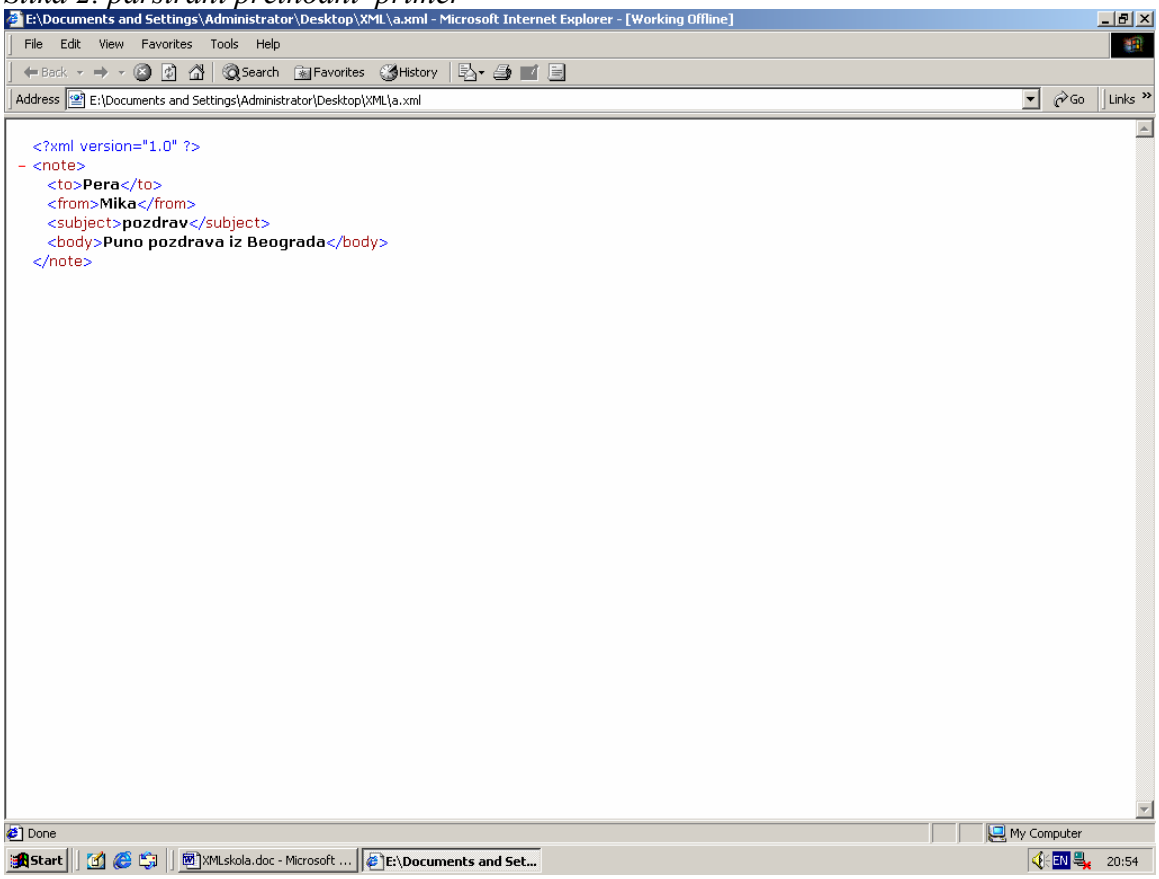



```
- <xml xmlns:s="uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882" xmlns:dt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882"
    xmlns:rs="urn:schemas-microsoft-com:rowset" xmlns:z="#RowsetSchema">
  - <s:Schema id="RowsetSchema">
    - <s:ElementType name="row" content="eltOnly" rs:updatable="true">
      - <s:AttributeType name="Address" rs:number="5" rs:nullable="true" rs:maydefer="true" rs:write="true"
          rs:basetable="CUSTOMERS" rs:basecolumn="Address">
          <s:datatype dt:type="string" dt:maxLength="60" />
        </s:AttributeType>
      - <s:AttributeType name="City" rs:number="6" rs:nullable="true" rs:maydefer="true" rs:write="true" rs:basetable="CUSTOMERS"
          rs:basecolumn="City">
          <s:datatype dt:type="string" dt:maxLength="15" />
        </s:AttributeType>
      - <s:AttributeType name="CompanyName" rs:number="2" rs:maydefer="true" rs:write="true" rs:basetable="CUSTOMERS"
          rs:basecolumn="CompanyName">
          <s:datatype dt:type="string" dt:maxLength="40" />
        </s:AttributeType>
      - <s:AttributeType name="ContactName" rs:number="3" rs:nullable="true" rs:maydefer="true" rs:write="true"
          rs:basetable="CUSTOMERS" rs:basecolumn="ContactName">
          <s:datatype dt:type="string" dt:maxLength="30" />
        </s:AttributeType>
      - <s:AttributeType name="ContactTitle" rs:number="4" rs:nullable="true" rs:maydefer="true" rs:write="true"
          rs:basetable="CUSTOMERS" rs:basecolumn="ContactTitle">
          <s:datatype dt:type="string" dt:maxLength="30" />
        </s:AttributeType>
      - <s:AttributeType name="Country" rs:number="9" rs:nullable="true" rs:maydefer="true" rs:write="true"
          rs:basetable="CUSTOMERS" rs:basecolumn="Country">
          <s:datatype dt:type="string" dt:maxLength="15" />
        </s:AttributeType>
      - <s:AttributeType name="CustomerID" rs:number="1" rs:nullable="true" rs:maydefer="true" rs:write="true"
          rs:basetable="CUSTOMERS" rs:basecolumn="CustomerID">
          <s:datatype dt:type="string" dt:maxLength="5" />
        </s:AttributeType>
      - <s:AttributeType name="Fax" rs:number="11" rs:nullable="true" rs:maydefer="true" rs:write="true" rs:basetable="CUSTOMERS"
          rs:basecolumn="Fax">
          <s:datatype dt:type="string" dt:maxLength="24" />
        </s:AttributeType>
```

Za dalju diskusiju su nam potrebni primeri pa predlažem da razmotrimo sledeći:

```
<?xml version="1.0" ?>
<note>
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</subject>
<body>Puno pozdrava iz Beograda</body>
</note>
```

*Slika 2. parsirani prethodni primer*




Ovako bi mogao da izgleda jedan običan i jednostavan XML fajl kada ga posmatramo iz Notepada. Odmah je uočljivo da postoje neki tagovi (oznake) koje uokviruju neki tekst. Upravo te oznake su ono što XML čini XML-om. One opisuju sadržaj odnosno informaciju. Međutim postoji još nešto što ovaj primer možda ne oslikava jasno – XML može da opiše svojom strukturom i strukturu informacija odnosno njihov odnos u smislu hijerarhije. Dve veoma bitne stvari. Informacija o informaciji i informacija o strukturi u jednom dokumentu je prilično lucidna stvar. To nam dopušta da kada već imamo relacione baze podataka da sadržaj istih interpretiramo na način da to bude lako upotrebljivo, brzo dostupno i, možda najbitnije, razmenljivo između aplikacija.

## Istorijat

Prvo se u IBM-u pojavio SGML (Standard Generalized Markup Language) kao odgovor na problem prebacivanja dokumenata sa jedne na drugu platformu. Zatim se krajem osamdesetih u CERN-u (evropskoj laboratoriji za fiziku atomskih čestica) pojavio HTML a kada je posle uspešne promocije na internetu postalo očigledno da HTML ne može baš sve rodila se ideja o XML-u negde 1996 godine. Međutim XML nije evoluirani HTML. On je komplement HTML-u i dizajniran je ne da zameni HTML već da ponudi ono što HTML ne može.

## XML činjenice

- XML je metod za smeštanje strukuiranih podataka u tekstualni fajl
  XML ima podatke i strukturu u tekstualnom fajlu. To ga kvalifikuje za rad sa bazama podataka.
- XML liči na HTML ali nije HTML
  Iako koristi tagove kao i HTML on se suštinski razlikuje od HTML-a jer proizvoljno proširiv tagovima koje sami izmišljate.
- XML je tekst ali nije namenjen čitanju od strane ljudi već mašina
  Iako je tekst nije namenjen čitanju već parsiranju od strane računara.
- XML je porodica tehnologija
  XML čini čitava porodica tehnologija i on sam po sebi ne predstavlja posebno mnogo funkcionalnosti ali u sadejstvu sa ostalim tehnologijama (CSS, XLink, XPointer, XFragments, XSL ...itd) daje odlične rezultate.
- XML za posledicu ima obiman fajl ali to nije problem
  Iako je XML fajl obiman zbog upotrebe oznaka to nije problem. Stvar se kompenzuje time što dobijate fleksibilnost u primeni.
- XML je nov ali nije baš toliko nov
  XML je relativno nov ali njegovi koreni sežu u početke osamdesetih. On predstavlja evoluciju ideje a ne evoluciju jezika.
- XML se može koristiti za razvoj novih jezika
  XML je preteča WAP-a i WML-a. Wireless Markup Language (WML), koji se koristi na primer u obeležavanju Internet aplikacija za mobilne telefone je zapravo usko specijalizovani XML. Slična stvar je i sa WAP-om.
- XML ne pripada nikome, ne zavisi od platforme i dobro je podržan
  XML je samo specifikacija W3 konzorcijuma. Ujedno je i preporuka priznatog autoriteta. Nema profitnu pozadinu i slobodan je za upotrebu. Da biste ga koristili niste ničim obavezani, ni platformom, ni proizvođačima, ni licencama i ugovorima. Shvatite njegovu suštinu i koristite ga onako kako vama odgovara.

# Namena XML-a

Kao što je to već pomenuto XML je komplement HTML-u a ne njegova bolja verzija. Zapravo XML samo liči na HTML jer koristi tagove kao i HTML ali je to posledica toga što je i on jezik za obeležavanje. HTML je namenjen da prikazuje dokument ali on nema jasnu granicu između sadržaja i forme. Sa XML-om je situacija potpuno drugačija. Forma je izdvojena u poseban fajl (na primer .css – cascading style sheets ako baš želite da formatirate fajl mada to nije prvenstvena ideja upotrebe XML-a) a sadržaj je u posebnom fajlu i on je suština XML-a. Koristi se da pruži podatke i njihov opis a sam prikaz često i ne postoji. Jednostavno, XML služi kao *format* - *spona* između dve aplikacije koja od jedne prihvata podatke i pruža je drugoj aplikaciji. U ovoj varijanti, prednost mu je što može koristiti HTTP protokol. Međutim teško je dati odgovor tip: "XML se koristi za to i to... " jer su mogućnosti primene praktično neograničene. Bolje je upoznati osobine XML-a i koristiti ga tamo gde je to prikladno nego se ograničavati nekim stvarima koje su više ilustracija primene nego pravilo.

## Kako radi XML

Za funkcionalnost koji pruža XML potreban je parser. U Internet Exploreru 5.0 i nadalje postoji ugrađen parser a za parsiranje iz npr. Visual Basica potrebno je postaviti referencu na XML parser. Parsiranje, kada je XML u pitanju znači sledeće – rasčlanjivanje tekstualnog fajla i pravljenje strukture koja se rekurzivno puni elementima XML stabla. To znači da parser izvodi sledeće operacije:

1. Isčitava preprocesorski deo dokumenta (deo na početku dokumenta između ? znakova) da bi došao od informacija koje se odnose na dokument a nisu deo samog XML stabla. Na primer: `<?xml version="1.0" encoding="windows-1252"?>`
2. Zatim se isčitava prvi tag u XML strukturi i zapisuje njegovo ime – ovo je top level ili startni tag.
3. Zatim se zapisuje ime elementa.
4. Zatim se isčitavaju ostali elementi redom da bi se odredilo koja svojstva ima dati element strukture i zatim se ta svojstva upisuju - ako je u pitanju element upisuje se njegova vrednost ili u formi uređenih parova *atribut* = *vrednost* ako je u pitanju atribut.
5. Ako sledeći tag nakon prvog nađenog nije oznaka za zatvaranje isčitava se sledeći tag i on se definiše kao dete trenutnog elementa. Onda se parser vraća na korak 3. Ako je nađeni tag oznaka za zatvaranje onda je element definisan.
6. Ovaj proces se ponavlja dok se ne obradi čitav dokument.

## Imenovanje elemenata

XML elementi moraju da poštuju sledeća pravila:

- Imena mogu sadržavati slova brojeve i druge karaktere.
- Imena ne smeju počinjati brojem ili interpunkcijskim karakterom.
- Imena ne smeju počinjati slovima xml ili XML ili Xml.
- Imena ne mogu imati prazan prostor u sebi.

Neke opšte preporuke bi bile:
Imena treba da budu samo opisujuća. Primeri su:

```
<prezime>, <adresa_stanovanja>
```

Imena treba da budu kratka i jasna jer to olakšava rukovanje:

```
<naslov_knjige>
```

a ne:

```
<naslov_knige_u_biblioteci>
```

XML dokumenti imaju često odgovarajuću bazu podataka pa nazivi elemenata treba da odgovaraju poljima u bazi. Moguće je koristiti i karaktere koji nisu engleski ali to vodi u

potencijalni rizik da stvar ne funkcioniše zbog nekog od elemenata softvera tako da bi naša lokalno važeća preporuka bila da se ovo izbegava.

Karakter ":" se ne treba koristiti u imenima jer je rezervisan za nešto drugo (namespace) o čemu će kasnije biti reči.

## *XML atributi*

XML elementi mogu imati atribute u otvarajućem tagu kao i HTML. Oni se koriste za dodatne informacije o elementu.
Iz HTML-a se sećamo ovoga:

```
<IMG SRC="slika.gif">
```

SRC atribut daje dodatne informacije o IMG elementu. Atributi često pružaju informacije koje nisu deo podataka. U sledećem primeru tip fajla je irelevantan za podatke ali je veoma bitan za softver koji manipuliše elemenom:

```
<file type="gif">slika.gif</file>
```

**Znaci navoda**

Vrednosti atributa moraju uvek biti unutar znaka navoda. Međutim moguće je koristiti jednostruke ili dvostruke znake navoda:

```
<ime="Krcun">
```

ili:

```
<ime='Krcun'>
```

Dupli znaci navoda su češći međutim nekada je neophodno koristiti jednostruke kao u sledećem primeru:

```
<ime='Slobodan "Krcun" Penezić'>
```

**Šta koristiti - element ili atribut?**

Podaci se mogu skladištiti ili u elementima ili u atributima. Element ima sledeću formu:

```
<ime>Krcun</ime>
```

dok je atribut u formi:

```
<nesto ime="Krcun">
```

Pogledajmo sledeća dva primera:

```
<komintent tip="nabavljač">
<ime>Pera</ime>
<prezime>Perić</prezime>
</komintent>
```

ili

```
<komintent>nabavljač</komintent>
<ime>Pera</ime>
<prezime>Perić</prezime>
```

U prvom primeru *tip* je atribut. U drugom primeru *tip* je element. Oba primera daju iste informacije. Ne postoje određena pravila kada koristiti atribute a kada elemente. Neka načelna preporuka je da se elementi koriste kada je u pitanju nešto što je samo po sebi celokupna informacija a ne neki njen pomoćni deo.

**Potencijalni problemi prilikom korišćenja atributa**

- Atributi ne mogu sadržavati višestruke vrednosti (elementi mogu)
- Atributi nisu lako proširivi
- Atributi ne opisuju strukturu
- Atributima se teže manipuliše u programskom kodu
- Vrednosti atributa se teško testiraju u odnosu na DTD (Document Type Definition – definicija tipa dokumenta)

**XML validacija**

XML dokument sa ispravnom sintaksom je **ispravno formirani XML dokument**.
XML proveren u odnosu na definiciju tipova dokumenata (DTD) je **ispravan XML dokument**.

**Ispravno formirani XML dokumenti**

Ispravno formirani XML dokumenti ima ispravnu XML sintaksu.

```
<?xml version="1.0" ?>
<note>
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</subject>
<body>Puno pozdrava iz Beograda</body>
</note>
```

**Ispravni XML dokumenti**
Ispravni XML dokumenti poštuju DTD. Oni su ispravno formirani XML dokumenti koji poštuju pravila DTD-a:
Ovde je InternalNote.dtd eksterna definicija tipa dokumenta. O DTD-u će više reči biti u sledećem nastavku.

```
<?xml version="1.0"?><!DOCTYPE note SYSTEM
"InternalNote.dtd">
<note>
```

```
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</subject>
<body>Puno pozdrava iz Beograda</body>
</note>
```

**Objašnjenja pojmova koji će se nadalje koristiti u tekstu**

**XML DTD**
DTD definiše legalne elemente XML dokumenta.
Svrha DTD-a je da definiše legalne gradivne blokove XML dokumenta. DTD definiše strukturu dokumenta sa listom elemenata.

**XML šema**
XML šema je na XML-u bazirana alternativa DTD-u.

# Sintaksa XML-a

Sintaksna pravila XML-a su veoma jednostavna i striktna. Lako se uče i još lakše primenjuju. Zbog toga je kreiranje aplikacija koje čitaju i manipulišu XML-om relativno jednostavno. Pogledajmo opet primer:

```
<?xml version="1.0" ?>
<note>
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</ subject>
<body>Puno pozdrava iz Beograda</body>
</note>
```

Prva linija XML dokumenta - XML deklaracija - određuje XML verziju dokumenta. U ovom slučaju dokument poštuje specifikaciju 1.0 XML-a koju propisuje W3Consortium. Ovaj red ujedno i govori Internet Exploreru da parsira (rasčlani) dokument XML parserom odnosno da dokument tretira kao XML fajl  a ne kao HTML fajl. Bez ove linije dobili bismo poruku o grešci od IE. Ova linija nema svoj yatvarajući ekvivalent jer ona nije deo XML dokumenta već njegova deklaracija. Sledeće je osnovni tag koji dokument formira kao poruku (<note>). Moguć je samo jedan osnovni tag inače opet dobijamo poruku o grešci. Sledeće četiri linije opisuju četiri podčlana osnovnog člana(to, from, subject, i body). Poslednja linija zatvara osnovni tag (</note>).

- Svi XML elementi moraju da budu zatvoreni
  U XML-u, izostavljanje završnog taga vodi u grešku. Dok je u HTML-u prolazilo:

  ```
  <p>ovo je paragraf<p>ovo je još jedan paragraf
  ```

  u XML-u ovo ne bi bilo ispravno već bi ispravan dokument izgledao ovako:

```
<p>ovo je paragraf</p><p>ovo je još jedan paragraf</p>
```

- XML tagovi razlikuju mala i velika slova
  Za razliku od HTML-a, XML tagovi su *case sensitive*.
  U XML-u, tag `<Poruka>` nije isti kao tag `<poruka>`. Stoga treba voditi računa da otvarajući i zatvarajući tagovi budu potpuno identični. I po nazivu i po upotrebljenim karakterima:

```
<Poruka>Ovo je neispravno</poruka>
<poruka>Ovo je ispravno</poruka>
```

- Svi XML elementi moraju biti propisno ugnežđeni
  Neispravno ugnežđeni elementi nemaju smisla u XML-u. Dok se u HTML-u elementi mogu preklapati u XML to nikako nije slučaj. Pogledajmo sledeći primer:

  HTML *ispravno*

```
<b><i>Ovo je tekst</b></i>
```

  XML *ispravno*

```
<b><i>Ovo je tekst</i></b>
```

- Svi XML dokumenti moraju da imaju osnovni (top level) ili startni tag
  Prvi tag u XML dokumentu je osnovni tag. Svi XML dokumenti moraju da imaju jedan par tagova koji definiše osnovni tag. Svi ostali elementi su ugnežđeni u osnovni tag. Gnežđenje u dubinu je neograničeno. Znači element može imati neograničen broj elemenata-dece. Odnos koji vlada je takozvani roditelj-dete odnos.

```
<note>
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</ subject>
<body>Puno pozdrava iz Beograda</body>
</note>
```

  Ovde je par osnovnih tagova `<note>` i `</note>` dok su podčlanovi parovi:

```
<to> i </to>
<from> i </from>
<subject> i </ subject>
<body> i </body>
```

- Vrednosti atributa moraju biti pod znacima navoda

U XML-u se vrednosti atributa moraju uokviriti znacima navoda. XML elementi mogu imati atribute i formi ime=vrednost parova (kao i u HTML). Pogledajmo ova dva XML dokumenta. Prvi je neispravan a drugi je ispravan:

```
<?xml version="1.0"?>
<note date=10/06/2000>
<note>
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</ subject>
<body>Puno pozdrava iz Beograda</body>
</note>

<?xml version="1.0"?>
<note date="10/06/2001">
<note>
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</ subject>
<body>Puno pozdrava iz Beograda</body>
</note>
```

- U XML-u je sačuvan prazan prostor
  Korišćenjem XML-a prazan prostor je prikazan u parsiranom dokumentu. Na primer:

  ```
  <body>Puno          pozdrava iz Beograda</body>
  ```

  će u parseru biti:

  **Puno pozdrava iz Beograda**

  dok to sa HTML-om nije slučaj.

- U XML-u, CR / LF karakteri se pretvaraju u LF karakter
  U XML-u, nov red u tekstu je uvek sačuvan kao LF (line feed). U Windows aplikacijama nov red je par CR ( carriage return) i LF (line feed) karaktera. Kod UNIX sistema karakter za nov red je LF mada neke aplikacije koriste i samo CR. Ova razlika među operativnim sistemima često za posledicu ima da se podaci vraćaju u obliku strima (engl. toka) a ne u željenom formatu.

- XML nije nešto specijalno ali ima svoje male tajne
  XML je zapravu samo tekst dizajniran tako da ga čita mašina odnosno softver a ne čovek. Softver koji podržava čisti tekst može da obrađuje XML. Na primer, u Notepadu se može obrađivati XML dokument. XML može da sadrži ne-engleske karaktere (č,ć,ž,đ...) međutim tada je potrebno dokument sačuvati u Unicode formatu što nije moguće u nekim verzijama Windows na primer u 95/98 dok je pod Windows 2000 operativnim sistemima to moguće. Stoga je u samu

deklaraciju XML fajla uveden i atribut encoding (engl. dešifrovanje) što zapravo govori browseru koju kodnu stranu da koristi.

```
<?xml version="1.0" encoding="windows-1252"?>
```

Međutim ovde je potrebno obratiti pažnju. Fajlovi sačuvani kao Unicode ne mogu imati i encoding atribut inače se pojavljuje greška u Internet Exploreru.

Pripremio *Lučić Aleksandar, MCP*

# Škola XML-a – DTD 2/3

## Uvod

U prethodnom članku smo se upoznali sa XML-om. Upoznali smo njegovu strukturu, sintaksu i stekli smo predstavu o njegovoj nameni. Međutim priča o XML-u se tu ne završava. Naime, iako je kao tehnologija relativno nezavisan XML se u praksi obično ne može posmatrati izolovano. Postoje tehnologije koje su komplementarne i koje dopunjuju XML u logičkom smislu. Jedna od tih tehnologija je DTD. DTD ili Document Type Definition (eng. definicija tipa dokumenta) je u odnosu na XML veoma bliska tehnologija. Komplementarna u logičkom smislu, ona proširuje XML. Namenjena je da opisuje gradivne elemente XML dokumenta. Šta to znači? Kao što smo videli, XML ima neku strukturu koju sačinjavaju elementi. Ta struktura je relativno fleksibilna tako da ako vam treba neki kalup za podatke (što XML jeste u izvesnom smislu) potrebna vam je upravo neka definicija koja će sa jedne strane ograničiti fleksibilnost XML ali će vam kao kompenzaciju pružiti validaciju podataka. Drugim rečima, pomoću DTD-a ćete definisati strukturu dokumenta tako što ćete napraviti listu dopuštenih elemenata. Na izvestan način pomoću DTD-a vi možete standardizovati XML dokument i na taj način ćete dobiti informacije o informacijama koje su standardizovane i koje posle toga možete razmenjivati između aplikacija a da pri tom budete sigurni da aplikacija koja prima podatke neće primiti podatke koji ne odgovaraju bilo da je u pitanju format, struktura ili sadržaj.

## Šta je DTD?

Videli smo da je XML više konvencija kôdiranja nego programski jezik. Logika je slična kada je i DTD u pitanju – on nije apsolutno samostalna tehnologija već je standard koji govori o implementaciji XML tehnologije. Naime, kada se stvari logički dovoljno iskomplikuju u XML-u DTD uvodi red. Na primer, imate neke podatke u XML fajlu. Sve je u redu dok je ta količina podataka kojom operišete relativno mala. Lako ćete pronaći neispravne podatke, lako ćete naći tagove koji ne sadrže podatke i lako ćete proveriti sadržaj fajla. Međutim kada je u pitanju veći skup podataka stvari počinju da se komplikuju a što su stvari komplikovanije veća je i mogućnost greške. U takvoj situaciji ćete definitivno poželeti da imate način da verifikujete podatke. Upravo to vam DTD omogućuje. Kombinovanjem XML-a i DTD-a dobijate univerzalno razmenjive podatke, koji sami o sebi vode računa. U prethodnom članku smo pomenuli da XML opisuje podatke i njihovu strukturu što veoma podseća na relacioni model baze podataka. Sličnost se nastavlja i u DTD-u. Naime, kao što u relacionoj bazi podataka možete postaviti neka pravila to isto možete uraditi i u XML-u pomoću DTD-a. Na taj način možete implementirati kompleksnu logiku u aplikacije koje koriste XML a da pritom ta logika bude laka za održavanje – smeštena je ili u sam XML fajl ili je u posebnom DTD fajlu. Ovde dolazimo do bitnog zaključka koji se odnosi na XML tehnologiju; kada je reč o XML-u i srodnm tehnologijama radi se o logičkoj implementaciji modela a ne toliko o tehnologiji radi tehnologije. Pomoću XML-a logički koncepti koji se već duže vreme koriste u informacionim tehnologijama dobijaju novu i lakšu za upotrebu, namenu a to je

razmena podataka među aplikacijama koja ima kvalitet više a to je mogućnost da te podatke razume sam softver i da na osnovu toga može da obavlja složenije poslove od dosadašnjih.

## Implementacija DTD-a

DTD poseduje jasna pravila koja pomalo liče na XML. DTD se može implementirati interno i eksterno. To znači da DTD može biti u okviru XML fajla ili može biti implementiran kao poseban fajl. U ovoj drugoj varijanti radi se takođe o tekstualnom fajlu koji ima .dtd ekstenziju. Bez obzira kako je implementiran on u oba slučaja pruža istu funkcionalnost a to je validacija podataka. Kako ćete ga implementirati zavisi od slučaja ali neka uopštena preporuka je se DTD implementira interno kada je reč o jednostavnijoj strukturi podataka a kada se radi o složenijim strukturama poželjno je da to bude eksterni fajl jer ćete na taj način i fizički razdvojiti pravila od podataka što je poželjno sa aspekta održavanja a veoma je verovatno da ćete DTD definisati pomoću nekakvog softverskog alata (na primer - XML Spy) što opet obično zahteva eksternu implementaciju.

Postoji i problem nasleđenih aplikacija. Naime, informacioni sistemi evoluiraju tokom vremena a ranije pisan softver vremenom postaje prevaziđen u smislu da više nije u stanju da se prilagodi zahtevima poslovanja tako da XML tehnologija pruža mehanizam da se premosti jaz između starijih aplikacija i novog načina poslovanja. Na primer, ako posedujete neki informacioni sistem koji je oslonjen na bazu podataka (ilustracije radi neka je to nekakva flat baza podataka bez relacija) pomoću XML-a odnosno DTD-a možete simulirati relacioni model baze podataka a da pri tome ne intervenišete u samoj bazi. Ona ostaje onakva kakva je bila a nadalje stvari funkcionišu kao u relacionom modelu.

Za dalju diskusiju predlažem da razmotrimo pravila DTD koja su relativno jednostavna a u kombinacijama daju veoma složene logičke modele.

## Pravila DTD-a

Pravila su kao što je to već pomenuto jednostavna i potrebno ih je striktno poštovati. Za početak DTD definiše dopuštene elemente i njihovu strukturu u XML fajlu. Tako se razni proizvođači softvera mogu dogovoriti da koriste zajedničke DTD-e prilikom razmene XML podataka. DTD takođe možete upotrebiti i za verifikaciju sopstvenih podataka.

Ispravno formatiran XML dokument je dokument čija sintaksa je ispravna. Međutim, da bi bio ispravno formatiran dokument mora da poštuje niz pravila. Ona uključuju ali nisu ograničena na sledeća pravila:

- Dokument se mora sastojati od jednog ili više elemenata.
- Dokument mora imati samo jedan element (top element) a ostali elementi moraju biti ugnežđeni u top elementu.

- Svi elementi moraju imati oznake (tagove) za otvaranje i zatvaranje.

Ispravno formatiran XML dokument koji poštuje i pravila DTD-a se naziva validnim XML dokumentom. XML parseri (procesori) mogu da provere na osnovu DTD-a XML dokumente. Na primer, Internet Explorer 5.0 (i na dalje) mogu da provere XML dokument.

Ako je DTD interno definisan onda je potrebno da on bude ugnežđen u DOCTYPE definiciju. Na primer <!DOCTYPE osnovnielement [element-deklaracije]>. Razmotrićemo primer iz prethodnog članka:

```
<?xml version="1.0" ?>
<note>
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</subject>
<body>Puno pozdrava iz Beograda</body>
</note>
```

Za validaciju potrebno je da proverimo sadržaj elemenata **to, from, subject, body**. Stoga DTD definicija treba da sadrži navede elemente. Interna DTD definicija bi za prethodni XMl dokumenat izgledala ovako:

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,subject,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT subject (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
```

*dok je ovo ostatak XML dokumenta*:

```
<note>
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</subject>
<body>Puno pozdrava iz Beograda</body>
</note>
```

DTD deo XML dokumenta će biti interpretiran kao:

!DOCTYPE **note** (linija2) definiše da je ovo dokument tipa **note**.
!ELEMENT **note** (linija3) definiše da **note** element ima sledeće podelemente:

**to, from, subject, body**

!ELEMENT **to** (linija4) definiše da je **to** element tipa "#PCDATA".
!ELEMENT **from** (linija5) definiše da je **from** element tipa "#PCDATA".

!ELEMENT **subject** (linija4) definiše da je **subject** element tipa "#PCDATA".
!ELEMENT **body** (linija4) definiše da je **body** element tipa "#PCDATA".

U varijanti sa spoljnom deklaracijom DTD postojaće fajl sa imenom *note.dtd* koji će sadržavati prethodno pomenutu definiciju:

```
<!ELEMENT note (to,from,subject,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Sa DTD stanovišta sav sadržaj XML fajla se može klasifikovati kao jedan od sledećih:

| | |
|---|---|
| Elementi | Elements |
| Tagovi | Tags |
| Atributi | Attributes |
| Entiteti | Entities |
| PCDATA parsirani karakter podaci | PCDATA |
| CDATA karakter podaci | CDATA |

Svaki od ovih elemenata ćemo ima svoju specifičnu funkciju prilikom provere na DTD ispravnost. Razmotrićemo svaki od pomenutih elemenata posebno.

## ELEMENTI

Elementi su osnovni gradivni blokovi XML dokumenta. Isto važi i za HTML dokumente. U HTML-u element je na primer paragraf <P>Ovo je paragraf</P>. U XML to je neki od tagova. Na primer u prethodnom primeru to su: **to, from, subject, body**. Elementi mogu sadržavati druge elemente ali mogu biti i prazni ili mogu sadržavati neki tekst. U primeru elementa, <P>**Ovo je paragraf**</P> podebljan je samo element.

## TAGOVI

Tagovi uokviruju elemente. Postoje dve vrste tagova: početni (na primer: **<note>**) i završni (na primer **</note>**). Kako su tagovi objašnjeni u prethodnom članku ovde se nećemo zadržavati na tome jer je stvar sa tagovima relativno jednostavna.

## Atributi

Atributi pružaju dodatne informacije o elementima. Bitna stvar kod atributa je to da su oni uvek smešteni u početni tag elementa. Po svojoj strukturi oni su uređeni parovi po

sistemu ime/vrednost. Na primer, "img" elemenat ima dodatnu informaciju o izvornom fajlu:

```
<img src="computer.gif" />
```

Ime elementa u gornjem primeru je "img". Ime atributa je "src" dok je vrednost atributa "computer.gif". Pošto je elemenat prazan on se zatvara sa " /".

## Entiteti

Entiteti su uslovno rečeno varijable. Oni se koriste da definišu tekst. Reference entiteta ukazuju na entitete. Verovatno vam je poznata HTML referenca entiteta " " koja se koristi da se doda mesto u dokumentu. Entiteti se obrađuju prilikom parsiranja XML parserom.

Sledeći entiteti su predefinisani u XML-u:

< < (less than – manje od)
> > (greater than - veće od)
& & (ampersand)
" " (navodnik)
' ' (apostrof)

Kao što možete videti njihova imena su prilično asocijativna što je zgodno jer ih ne morate pamtiti – logika je sledeća uvek počinju sa & iza koda sledi tekst koji opisuje njihovo značenje.

## PCDATA

PCDATA je skračenica od "parsed character data" – parsiranih karaktera podataka. Ono "parsirani" je verovatno jasno ali šta tačno znači "karakteri podataka"? I ovde je stvar jednostavna (zapravo deluje komplikovano a u suštini je jednostavno) radi se o tekstu koji se nalazi između početnog i završnog taga XML elementa. Znači napravljena je razlika između sadržaja elementa i njegovog naziva u logičkom smislu.

PCDATA je tekst koji će biti parsiran od strane parsera a tagovi unutar teksta će se tretirati kao oznake.

## CDATA

CDATA znači "karakteri podataka". Slično prethodnoj logici CDATA je tekst koji neće biti parsiran od strane parsera. Tagovi unutar teksta neće se tretirati kao oznake a enteti neće biti prikazani.

## Deklarisanje elemenata

U DTD-u, XML elementi se deklarišu pomoću DTD deklaracije elemenata. Deklaracija elemenata je u logičkom smislu veoma slična deklaracijama koje nalazimo na proizvodima. Na primer u deklaraciji keksa piše da su sastojci brašno, šećer, kakao itd. Slična stvar je i kod DTD-a samo što su u tom slučaju sastojci sami elementi XML dokumenta. Deklaracija se razlikuje od one na keksu i ima neka pravila koja se moraju poštovati. Sintaksa je jedno od njih i ona izgleda ovako:

<!ELEMENT elemenat-ime kategorije>

ili

<!ELEMENT ime-elementa(sadržaj-elementa)>

## Prazni elementi

Prazni elementi se deklarišu pomoću ključne reči kategorije EMPTY.

<!ELEMENT ime-elementa EMPTY>

DTD primer:
<!ELEMENT oznaka EMPTY>Primer<oznaka />

Ovde je elemet sa imenom oznaka deklarisan kao prazan elemenat.

## Elementi koji sadrže samo karakter podatke

Elementi koji sadrže samo karakter podatke se deklarišu sa #PCDATA unutar zagrada:

<!ELEMENT ime-elementa (#PCDATA)>

DTD primer:
<!ELEMENT subject (#PCDATA)>

Ovde je elemet sa imenom oznaka deklarisan da sadrži samo karakter podatke.

## Elementi sa bilo koji sadržajem

Elementi koji su deklarisani sa ključnom rečju kategorije ANY mogu da sadrže bilo koju kombinaciju podataka koji se mogu parsirati.

<!ELEMENT ime-elementa ANY>

DTD primer:
<!ELEMENT note ANY>
Ovde je elemet sa imenom poruka deklarisan da sadrži bilo koje podatke.

## Elementi koji sadrže podelemente (sekvence)

Elementi koji sadrže podelemente (takvi elementi se često nazivaju sekvence podelemenata) se definišu sa imenima podelemenata u zagradi.

<!ELEMENT ime-elementa (ime-podelementa)>

ili ako je više podelemenata

<!ELEMENT element-name (ime-podelementa, ime-podelementa,.....)>

DTD primer:
<!ELEMENT note (to,from,subject,body)>

Međutim ovde treba obratiti pažnju na jedan sitan detalj. Naime, kada se podelementi deklarisani na ovaj način:

<!ELEMENT note (to,from,subject,body)>

onda se oni moraju u XML dokumentu pojavljivati ovim redosledom:

to
from
subject
body

znači istim redosled kojim su i deklarisani. Ovakva vrsta deklaracije se naziva sekvencijalna deklaracija (postoji i potpuna deklaracija). Potpuna deklaracija poruke bi izgledala ovako:

<!ELEMENT note (to,from,subject,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT body (#PCDATA)>

U potpunoj deklaraciji podelementi mogu imati dalje podelemente.

## Deklarisanje samo jednog pojavljivanja istog elementa

Upotrebom DTD-a možete odrediti koliko puta će se neki sadržaj pojavljivati u XML dokumentu. Tako na primer možete odrediti da se jedan element može pojaviti samo jeadnput u XML dokumentu. Ako se kojim slučajem elemenat pojavi još koji put pojaviće se greška prilikom validacije dokumenta.

<!ELEMENT ime-elementa (ime-podelementa)>
DTD primer:
<!ELEMENT note (message)>

U prethodnom primeru je deklarisano da se podelement message može pojaviti samo jedanput u note elementu.

## Deklarisanje barem jednog pojavljivanja istog elementa

Možete deklarisati da se elemant mora pojaviti jedanput ili više puta u dokumentu.

<!ELEMENT ime-elementa (ime-podelementa+)>

DTD primer:
<!ELEMENT note (message+)>

Znak + naznačuje da se podelement message mora  pojaviti jednom ili više puta.

## Deklarisanje nula ili više pojavljivanja istog elementa

Možete deklarisati da se elemant mora pojaviti nula ili više puta u dokumentu.

<!ELEMENT ime-elementa (ime-podelementa*)>

DTD primer:
<!ELEMENT note (message*)>

Znak * naznačuje da se podelement message mora  pojaviti nula ili više puta.

## Deklarisanje nula ili jednog pojavljivanja istog elementa

Možete deklarisati da se elemant mora pojaviti nula ili jedanput u dokumentu.

<!ELEMENT ime-elementa (ime-podelementa?)>

DTD primer:
<!ELEMENT note (message?)>

Znak ? naznačuje da se podelement message mora  pojaviti nula ili jedanput.

## Deklarisanje *ili* sadržaja

Da bi se postigla fleksibilnost prilikom deklaracija postoji i opciona deklaracija. To znači da možete deklarisati više elemenata od koji se mora pojavljivati jedan od njih.

DTD primer:
<!ELEMENT note (to,from, subject,(message|body))>

U prethodnom primeru, moraju se pojavljivati u note elementu sledeći podelementi:
to
from
subject

i mora se pojaviti samo message ili samo body podelement.

## Deklaracija mešovitog sadržaja

Možete kombinovati sadržaj prilikom deklaracije. Kombinovanjem sadržaja postižete potpunu fleksibilnost a da pritom zadržavate osnovnu prednost DTD-a – a to je kontrola sadržaja XML dokumenta.

DTD primer:
<!ELEMENT note (#PCDATA|to|from|subject|message)*>

U prethodnom primeru element note može sadržavati parsirane karakter podatke ili bilo koji broj to, from, subject, message elemenata.

## Deklaracija atributa

U DTD-u, atributi se deklarišu u ATTLIST deklaraciji.

**Deklarisanje artibuta**

Deklaracija atributa ima sledeću sintaksu:

<!ATTLIST ime-elementa ime-atributa tip-atributa podrazumevana-vrednost>

DTD primer:
<!ATTLIST vrsta uplate CDATA "gotovina">

ekvivalentni XML primer:

<vrsta uplate="gotovina" />

Tip atributa može imati sledeće vrednosti:

**CDATA** atribut ima vrednost karakter podataka

**(en1|en2|..)** vrednost mora biti jedna od navedenih vrednosti u listi

**ID** vrednst je jedinstveni identifikator (nema ponavljanja)

**IDREF** vrednost je referenca na ID drugog elementa.

**IDREFS** vrednost je lista drugih ID

**NMTOKEN** vrednost je validno XML ime

**NMTOKENS** vrednost je lista validnih XML imena

**ENTITY** vrednost je entitet

**ENTITIES** vrednost je lista entiteta

**NOTATION** vrednost je ime notacije

**xml**: vrednost je predefinisana xml vrednost

podrazumevana vrednost može biti neka od sledećih:

**Podrazumevana vrednost atributa**

**#DEFAULT** vrednost podrazumevena vrednost atributa

**#REQUIRED** vrednost atributa mora biti dodata

**#IMPLIED** atribut ne mora biti dodat

**#FIXED** vrednost vrednost atributa je fiksna

***Primeri deklaracije atributa:***

Opšti DTD primer:

<!ELEMENT provougaonik EMPTY>
<!ATTLIST pravougaonik visina CDATA "10">

XML primer:
<pravougaonik visina="100"></pravougaonik>

U prethodnom primeru, elemenat pravougaonik je definisan kao prazan elemenat sa visinom koja je tipa CDATA. Ako se artibut visine ne zada on ima vrednost koja je podrazumevana i ona iznosi 10.

*Razmotrićemo svaku od mogućih vrednosti atributa u posebnom primeru*

**#DEFAULT vrednost**

Sintaksa:
<!ATTLIST ime-elementa ime-atributa tip-atributa "podrazumevana-vrednost">

DTD primer:
<!ATTLIST vrsta uplate CDATA "gotovina">

XML primer:
<vrsta uplate="gotovina" />

Određivanjem podrazumevane vrednosti za atribut se postiže da atribut ima vrednost čak ako se ona i ne unese. Ovo podseća na *default* vrednost u bazi podataka.

**#IMPLIED vrednost**

Sintaksa:
<!ATTLIST ime-elementa ime-atributa tip-atributa #IMPLIED>

DTD primer:
<!ATTLIST broj telefona CDATA #IMPLIED>

XML primer:
<broj telefona="555-667788" />

Implicitna vrednost atributa se može koristiti kada želite da autor ne unosi atribut a nemate podrazumevanu vrednosti.

**#REQUIRED vrednost**

Sintaksa:
<!ATTLIST ime-elementa ime-atributa tip-atributa #REQUIRED>

DTD primer:
<!ATTLIST licni broj CDATA #REQUIRED>

XML primer:
<licni broj="5677" />

Zahtevana vrednost atributa se može koristiti kada nemate podrazumevanu vrednost ali želite da atribut ipak bude unet.

**#FIXED vrednost**

Sintaksa:
<!ATTLIST ime-elementa ime-atributa tip-atributa #FIXED "vrednost">

DTD primer:
<!ATTLIST posiljalac CDATA #FIXED "Microsoft">

XML primer:
<posiljalac="Microsoft" />

Fiksna vrednost atributa se koristi kada ne želite da se menja vrednost nekog atributa. Ako se vrednost nekim slučajem promeni XML parser će vratiti grešku.

## Enumerisani atributi

Sintaksa:
<!ATTLIST ime-elementa ime-atributa (en1|en2|..) "podrazumevana vrednost">

DTD primer:
<!ATTLIST vrsta uplate (cek|gotovina) "gotovina">

XML primer:
<vrsta uplate="gotovina" />
*ili*
< vrsta uplate ="cek" />

Enumerisane atribute možete koristiti kada želite da vrednost atributa bude jedna od unapred definisanih fiksnih vrednosti.

## Deklaracija entiteta

Entiteti su promenljive koje predstavljaju prečice do nekog zajedničkog teksta.

- Entiteti mogu biti deklarisani interno ili eksterno.

**Interna deklaracija entiteta**
Sintaksa: <!ENTITY ime-entiteta "vrednost-entiteta">

DTD primer:<!ENTITY autor "Pera Peric">
<!ENTITY godina "2001">

XML primer:<opis>&autor;&godina;</opis>

**Eksterna deklaracija entiteta**

Sintaksa: <!ENTITY ime-entiteta SYSTEM "URI/URL">

DTD primer:<!ENTITY autor SYSTEM "http://www.cet.co.yu/entiteti/entitet.xml">
<!ENTITY godina SYSTEM "http://www.cet.co.yu/entiteti/entitet.dtd">

XML primer:<opis>&autor;&godina;</opis>

## Validacija XML dokumenata

Internet Explorer 5.0 može da proveri da li XML dokument poštuje pravila definisana u DTD-u.

**Validacija pomoću XML parsera**

Ako pokušate da otvorite XML dokument koji ima neku grešku XML parser će vratiti grešku. Objekat parseError sadrži tačan broj greške, poruku o grešci i broj linije u dokumentu na kome se pojavila greška:
*JavaScript primer:*

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.validateOnParse="true"
xmlDoc.load("note_dtd_error.xml")

document.write("<br>Error Code: ")
document.write(xmlDoc.parseError.errorCode)
document.write("<br>Error Reason: ")
document.write(xmlDoc.parseError.reason)
document.write("<br>Error Line: ")
document.write(xmlDoc.parseError.line)
```

Međutim validaciju XML dokumeta možete i isključiti ako vam to odgovara. Potrebno je da podesite svojstvo parsera **validateOnParse** na **false**.

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.validateOnParse="false"
xmlDoc.load("note_dtd_error.xml")

document.write("<br>Error Code: ")
document.write(xmlDoc.parseError.errorCode)
document.write("<br>Error Reason: ")
document.write(xmlDoc.parseError.reason)
document.write("<br>Error Line: ")
document.write(xmlDoc.parseError.line)
```

**DODATNI PRIMERI DTD dokumenata**

**DTD za TV program**

```
<!DOCTYPE TVPROGRAM
[       <!ELEMENT TVPROGRAM (KANAL+)>
        <!ELEMENT KANAL (BANER,DAN+)>
        <!ELEMENT BANER (#PCDATA)>
        <!ELEMENT DAN (DATUM,(PRAZNIK|EMISIJA+))+>
        <!ELEMENT PRAZNIK (#PCDATA)>
        <!ELEMENT DATUM (#PCDATA)>
        <!ELEMENT EMISIJA (VREME,NAZIV,OPIS?)>
        <!ELEMENT VREME (#PCDATA)>
        <!ELEMENT NAZIV (#PCDATA)>
        <!ELEMENT OPIS (#PCDATA)>

        <!ATTLIST TVPROGRAM IME CDATA #REQUIRED>
        <!ATTLIST KANAL PROGRAM CDATA #REQUIRED>
        <!ATTLIST EMISIJA TRAJANJE CDATA #IMPLIED>
        <!ATTLIST NAZIV GLEDANOST CDATA #IMPLIED>
        <!ATTLIST NAZIV JEZIK CDATA #IMPLIED>
]>
```

**DTD za novinski članak**

```
<!DOCTYPE NOVINSKICLANAK
[       <!ELEMENT NOVINSKICLANAK (CLANAK+)>
        <!ELEMENT CLANAK
(NASLOV,PODNASLOV,UVOD,TEKSTCLANKA,NAPOMENE)>
```

```
        <!ELEMENT NASLOV (#PCDATA)>
        <!ELEMENT PODNASLOV (#PCDATA)>
        <!ELEMENT UVOD (#PCDATA)>
        <!ELEMENT TEKSTCLANKA (#PCDATA)>
        <!ELEMENT NAPOMENE (#PCDATA)>
         <!ATTLIST CLANAK AUTOR CDATA #REQUIRED>
        <!ATTLIST CLANAK UREDNIK CDATA #IMPLIED>
        <!ATTLIST CLANAK DATUM CDATA #IMPLIED>
        <!ATTLIST CLANAK IZDANJE CDATA #IMPLIED>
        <!ENTITY NOVINSKICLANAK "Citaliste">
        <!ENTITY IZDAVAC "CET">
]>
```

**DTD za katalog proizvoda**

```
<!DOCTYPE KATALOG
[       <!ENTITY AUTOR "Ime Autora">
        <!ENTITY FIRMA "Prodavnica Delova d.o.o.">
        <!ENTITY EMAIL "prodavnica@co.yu">
        <!ELEMENT KATALOG (PROIZVOD+)>
        <!ELEMENT PROIZVOD
        (SPECIFIKACIJE+,OPCIJE?,CENA+,NAPOMENE?)>
        <!ATTLIST PROIZVOD
        NAME CDATA #IMPLIED
        KATEGORIJA (Alat|Deo|Potrosni materijal) "Alat"
        BROJARTIKLA CDATA #IMPLIED
        PROIZVODJAC (Proizvodjac1| Proizvodjac2| Proizvodjac3) " Proizvodjac1 "
        LAGER (Na lageru|Poruceno|van programa) "Na lageru">
        <!ELEMENT SPECIFIKACIJE (#PCDATA)>
        <!ATTLIST SPECIFIKACIJE
        MASA CDATA #IMPLIED
        JACINA CDATA #IMPLIED>
        <!ELEMENT OPCIJE (#PCDATA)>
        <!ATTLIST OPCIJE
        OBRADA (Metal|Polirano|Mat) "Mat"
        ADAPTER (Uracunat|Opcionalno|Nema) "Nema"
        VERZIJA (Profesional|Hobi|Nema) "Profesional">
        <!ELEMENT CENA (#PCDATA)>
        <!ATTLIST CENA
        MALOPRODAJNA CDATA #IMPLIED
        VELEPRODAJNA CDATA #IMPLIED
        ULICA CDATA #IMPLIED
        ISPORUKA CDATA #IMPLIED>
        <!ELEMENT NAPOMENE (#PCDATA)>
]>
```

Pripremio *Lučić Aleksandar, MCP*

Škola XML-a 3/3.a

U prethodna dva članka smo upoznali osnove XML-a i DTD-a. U trećem delu ćemo se pozabaviti Microsoftovom implementacijom XML standarda. Microsoft je XML implementirao kroz XML DOM (XML Document Object Model – XML objektni model dokumenta). Bitno je napomenuti da je XML samo preporučeni standard. Drugim rečima, to znači da XML ne pripada nikome od proizvođača softvera već se proizvođači trude da ga podrže u svom softveru. Razmatranje XML DOM-a kao Microsoftove implemetacije XML standarda ima svoje razloge jer se ipak radi o proizvođaču koji trenutno dominira u mnogim segmentima softverskog tržišta tako da se investicija u upoznavanje XML DOM-a isplati. Nadalje, XML DOM ima svoje mesto i u budućem Visual Studio.NET-u tako da je poznavanje istog ako ne neophodno onda barem korisno. Onima koji se prvi put sreću sa XML DOM-om čitava stvar deluje prilično komplikovano ali ako se samo malo potrudite onda rezultati definitivno dolaze veoma brzo. U ovom članku ćemo obraditi XML DOM, njegove metode, a svojstva i događaje u narednom članku. Masa primera će biti u VB razvojnom okruženju koje najtoplije preporučujem za upoznavanje sa XML DOM-om jer pored Intellisense-a koji stalno prikazuje svojstva i metode i na taj način ubrzava učenje, omogućuje i relativno lako pronalaženje grešaka. Normalno, DOM nije ograničen samo na VB. Možete ga koristiti na sličan način i u VB skriptu ili Java skriptu tako da će biti primera i u ovim skriptovima koje možete koristiti u nekim vašim Internet aplikacijama koje se izvršavaju u web pretraživaču (IE5 je minimalna verzija).

## XML DOM – o čemu se radi

XML DOM je kao što je to već pomenuto Microsoftova implementacija XML-a i njegovih srodnih tehnologija (na primer XSL-a). Kako ćemo se ovde baviti uglavnom XML-om kroz XML DOM, bilo koja od četiri trenutno raspoložive verzije će biti dovoljna. Međutim ako planirate da koristite srodne tehnologije (na primer, XSL) obratite pažnju na verziju. Microsoft tek sa verzijom 3.0 u potpunosti podržava XSL. Slično je i sa ostalim srodnim tehnologijama tako da će najbolje biti da koristite verziju 3.0 ako je imate a ako je nemate onda će bilo koja biti dobra jer se ovde ipak radi o osnovnim stvarima. XML DOM je implementiran kroz ActiveX tehnologiju i fizički se radi o msxml3.dll-u (druge verzije se razlikuju u broju). Reč je jednostavno o COM komponenti. XML DOM kreira XML dokument od postojećeg XML fajla u kome možete raditi sa XML čvorovima. Za one upućenije čitava stvar dosta liči na bazu podataka zbog hijerarhije koja se može uočiti ali napominjem još jednom – to nije generalna ideja XML-a.
XMLDOM ima objekte opisane u sledećem delu.

## XMLDOM objekti

Osnovni **XMLDOM** objekti su:

**XMLDOMDocument** objekat – Predstavlja najviši čvor u hijerarhiji XMLDOM-a.

**XMLDOMNode** objekat – Predstavlja jedan čvor u dokumentu. Ovo je osnovni interfejs u XMLDOM objektnom modelu.
**XMLDOMNodeList** objekat – Lista čvorova. Podržava iteraciju i indeksirani pristup kojekciji XMLDOMNode objekata.
**XMLDOMNamedNodeMap** objekat - Obezbeđuje iteracije i pristup po imenu kolekciji atributa.
**XMLDOMParseError** objekat - Daje broj linije, poziciju karaktera i tekstualni opis poslednje greške.
**XMLHttpRequest** objekat - Klijentima obezbeđuje podršku za komunikaciju sa HTTP serverom.
**XTLRuntime** objekat - Ovaj objekat implementira metode koje se koriste u XSL fajlovima.

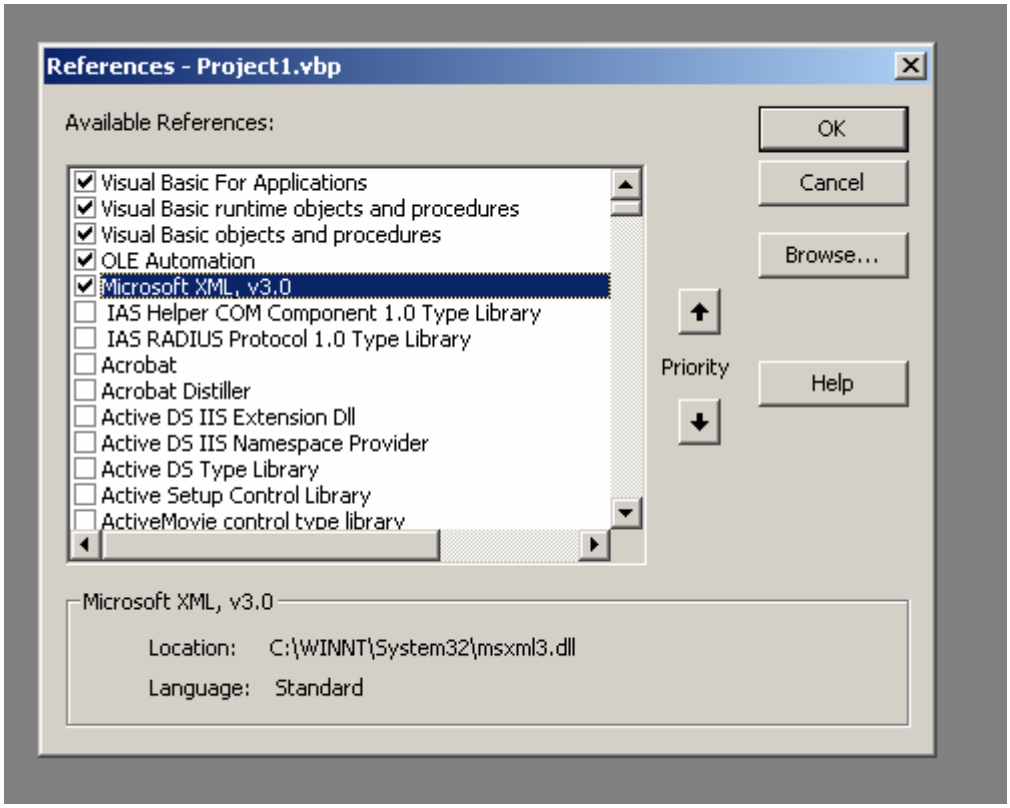
dok sledeći objekti predstavljaju dalju implementaciju XMLDOM interfejsa:

**IXMLDOMAttribute** – Predstavlja attribute objekat.
**IXMLDOMCDATASection** – Predstavlja CDDATA (character data - podaci karakteri) tekst tako da se on ne interpretira kao jezik za obeležavanje.
**IXMLDOMCharacterData** – Daje metode za manipulaciju tekstom koje koriste neki od ostalih objekata.
**IXMLDOMComment** - Predstavlja sadržaj XML komentara.
**IXMLDOMDocumentFragment** – Predstavlja fragment dokumenta.
**IXMLDOMDocumentType** – Sadrži informacije o deklaraciji tipa dokumenta.
**IXMLDOMElement** - Predstavlja element objekat.
**IXMLDOMEntity** – Predstavlja parsirani ili neparsirani entitet u XML dokumentu.
**IXMLDOMEntityReference** – Predstavlja referencu eniteta.
**IXMLDOMImplementation** - Pruža metode koji su nezavisne za datu instancu objektnog modela dokumenta..
**IXMLDOMNotation** – Sadrži notaciju koja je deklarisana u DTD-u ili šemi.
**IXMLDOMProcessingInstruction** – Predstavlja instrukcije za obradu koje definiše XML.
**IXMLDOMText** - Predstavlja tekstualni sadržaj elementa ili komentara.

Sve pomenute objekte kao i njihova svojstva, metode i događaje možete videti u Object Browser-u Visual Basica. Takođe, pomoću njega možete dobiti informacije o globalnim konstantama.

## Kreiranje objekta i učitavanje XML dokumenta

Ako radite u **Visual Basic** okruženju dodajte referencu na XML DOM da biste kreirali XML dokument:




a zatim možete napisati nešto nalik:

`Dim xmlDoc As New DOMDocument30` – za vezivanje preko *vtable* adrese

ili na primer:

```
Dim xmlDoc As Object
Set xmlDoc = CreateObject("Microsoft.XMLDOM") – za vezivanje preko DispID adrese
```

ili

```
Dim xmlDoc As Object
Set xmlDoc = CreateObject("Microsoft.FreeThreadedXMLDOM ") – za vezivanje u verziji sa slobodnom niti.
```

dok je u skript jezicima možete odmah kreirati objekat koji predstavlja XML dokument:

**JavaScript**:

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
```

ili

```
var xmlFTDoc = new ActiveXObject("Microsoft.FreeThreadedXMLDOM");
```

ili

**VBScript**:

```
Set xmlDoc = CreateObject("Microsoft.XMLDOM")
```

ili

```
Set xmlFTDoc = CreateObject("Microsoft.FreeThreadedXMLDOM")
```

U primerima je samo jedna linija koda je dovoljna za kreiranje objekta. Koji način ćete koristiti zavisi od toga da li vam treba slobodna ili deljena nit. U obe varijante objekat će se isto ponašati ali morate znati da se čvorovi dokumenta ne mogu kombinovati ako su kreirani različitim modelima niti. U verziji sa deljenom niti su bolje performanse jer XML parser ne mora da vodi računa o istovremenim pristupima među nitima.

Kada kreirate objekat još uvek ne možete raditi sa njim. Potrebno je da učitate xml fajl koji će formirati strukturu objekta. To se postiže pomoću `load`() metode xml objekta.

Na primer za xml objekat `xmlDoc` potrebno je da napišete:

`xmlDoc.load(C:\NekaPutanja\NekiXmlFajl.xml)` - za učitavanje iz lokala

`xmlDoc.load(http://lokacija_na_webu/NekiXmlFajl.xml)` - za učitavanje sa web lokacije.

Ovde ćemo napraviti malu digresiju. Naime, iako se svojstva, metode i događaji xml objekta zovu isto bez obzira na skript jezike ili VB, ima malih razlika u sintaksi. U VB-u se `Load()` metoda piše sa velikim L – `Load()` dok je u JavaScriptu situacija obrnuta – `load()`. Kako je **JavaScript** osetljiv na velika i mala slova svi primeri će biti pisani za njega. **Visual Basic** oko ovoga neće praviti probleme (sam će sve dovesti u red).

Znači trenutno imamo XML dokument koji je zapravo učitan xml fajl. Samo učitavanje se može izvršiti na dva načina: sinhrono i asihrono. Asihrono učitavanje (inače inicijalni način učitavanja) ili dovlačenja sa nekog URL-a znači da se kontrola izvršavanja vraća onome koji je pozvao metodu odmah po pozivu ove metode. Međutim samo učitavanje zavisi od mnogo parametara (da li je xml fajl u lokalu ili na nekoj udaljenoj lokaciji, veličina fajla, prioritetu procesa (pošto se radi o COM komponenti) itd.) tako da postoji i sinhrono učitavanje koje je zapravo asihrono učitavanje sa vrednošću `async` svojstva xml objekta `false`. Tada se kontrola vraća tek nakon potpunog učitavanja fajla. Drugim rečima ako je `async` svojstvo postavljeno na `false` onda će objekat biti dostupan tek nakon potpunog učitavanja xml fajla što može potrajati u slučaju veoma velikog fajla. Inicijalna vrednost async svojstva je true pa je potrebno postaviti je na false za asinhroni način učitavanja ako odmah nakon ovoga pozivate objekat. Za slučaj da radite sinhrono imate svojstvo `readyState` koje vam daje trenutno stanje u vezi sa učitavanjem. Više detalja u vezi sa njim imate u delu članka koji opisuje događaje.

## Rad sa XML objektom

Do sada smo kreirali xml objekat i učitali xml fajl. Za dalje objašnjavanje nam treba neki primer tako da ćemo se u većem delu teksta baviti sa sledećim xml fajlom:

**********

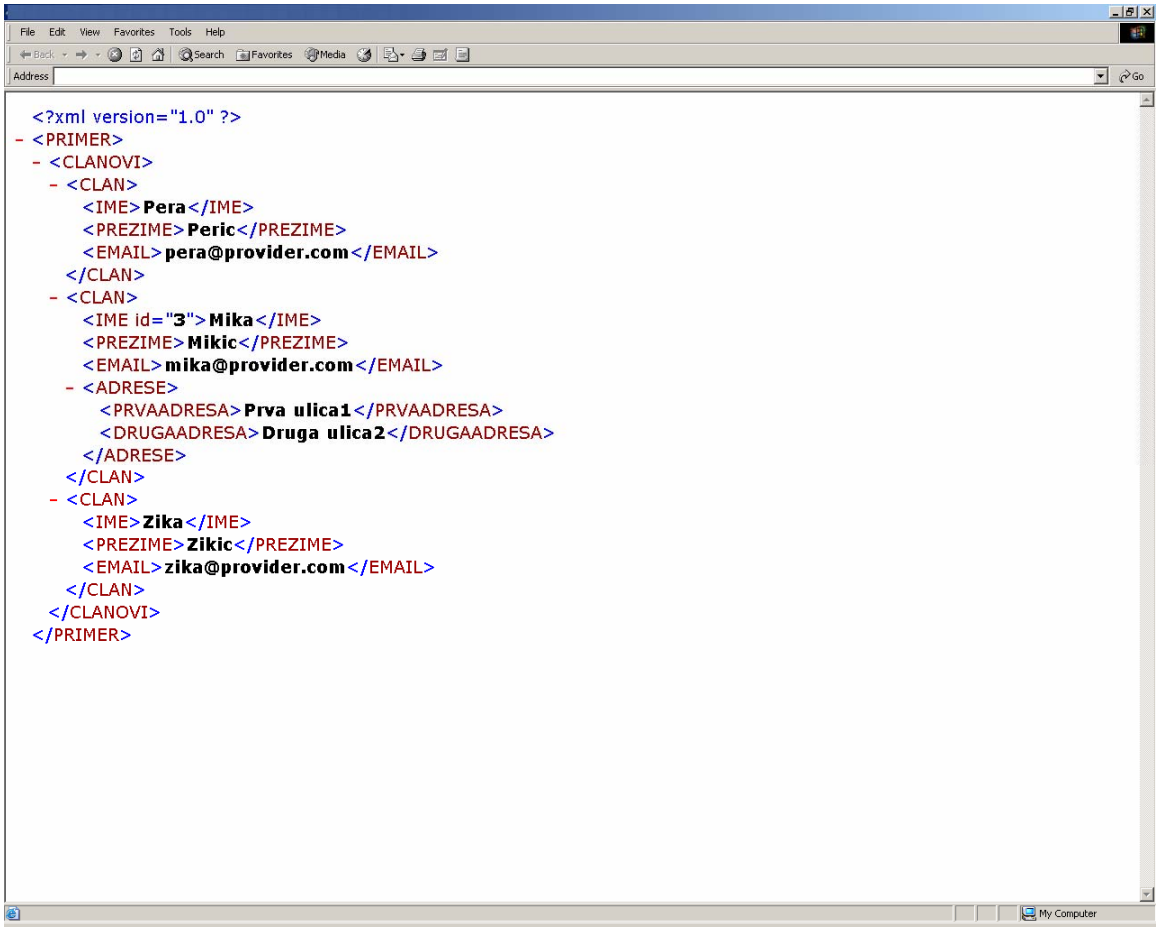
```
<?xml version="1.0"?>
<PRIMER>
     <CLANOVI>
          <CLAN>
               <IME>Pera</IME>
               <PREZIME>Peric</PREZIME>
               <EMAIL>pera@provider.com</EMAIL>
          </CLAN>
          <CLAN>
               <IME id="3">Mika</IME>
               <PREZIME>Mikic</PREZIME>
               <EMAIL>mika@provider.com</EMAIL>
               <ADRESE>
                    <PRVAADRESA>Prva ulica1</PRVAADRESA>
                    <DRUGAADRESA>Druga ulica2</DRUGAADRESA>
               </ADRESE>
          </CLAN>
          <CLAN>
               <IME>Zika</IME>
               <PREZIME>Zikic</PREZIME>
               <EMAIL>zika@provider.com</EMAIL>
          </CLAN>
     </CLANOVI>
</PRIMER>
```

**********

Ako bolje pogledate videćete da xml fajl nije apsolutno simetričan. Prvi i treći član u listi članova su identični dok je drugi član različit. On je proširen sa dva taga <PRVAADRESA> i <DRUGAADRESA> a tag <IME> ima atribut id=3.

Kada se ovaj fajl isparsira dobićete u web pretraživaču sledeće:

Slika 2 izgled parsiranog fajla u pretraživaču




Xml objekat ima svojstvo koje se zove `xml`. Ono nam daje sadržaj xml fajla što po sadržaju u potpunost odgovara prethodno navedenom primeru:

```
<?xml version="1.0"?>
<PRIMER>
	<CLANOVI>
		<CLAN>
			<IME>Pera</IME>
			<PREZIME>Peric</PREZIME>
			<EMAIL>pera@provider.com</EMAIL>
		</CLAN>
		<CLAN>
			<IME id="3">Mika</IME>
			<PREZIME>Mikic</PREZIME>
			<EMAIL>mika@provider.com</EMAIL>
			<ADRESE>
				<PRVAADRESA>Prva ulica1</PRVAADRESA>
				<DRUGAADRESA>Druga ulica2</DRUGAADRESA>
			</ADRESE>
		</CLAN>
		<CLAN>
			<IME>Zika</IME>
			<PREZIME>Zikic</PREZIME>
			<EMAIL>zika@provider.com</EMAIL>
```

```
            </CLAN>
      </CLANOVI>
</PRIMER>
```

On takođe ima i svojstvo `text` koje daje sadržaj celog teksta u xml objektu:

```
Pera Peric pera@provider.com Mika Mikic mika@provider.com Prva ulica1
Druga ulica2 Zika Zikic zika@provider.com
```

Razlika je očigledna. Svojstvo `xml` daje sadržaj sa tagovima dok `text` svojstvo ima samo tekstualni sadržaj bez tagova i atributa xml objekta. Ova dva svojstva su najčešće korišćena.

Sledi spisak svih važnijih svojstava i primeri sa objašnjenjima. Međutim, kako Microsoftova implementacija pokriva veliku oblast moguće primene ovde ćemo se koncentrirati na **XMLDOM Document** objekat a neke naprednije stvari ćemo morati da ostavimo za neku drugu priliku zbog obima članka (samo ćemo napomenuti da postoje i daćemo kratak opis bez nekog naročitog ulaženja u detalje). To ipak ne umanjuje ono što je ovde predstavljeno – ona vam je sasvim dovoljna da lako počnete da radite sa XML-om a kasnije kako se u praksi budu javljali novi, specifični zahtevi i nova pitanja biće vam mnogo lakše da nađete odgovore na njih jer ćete biti u poziciji da postojeće znanje nadgrađujete sa nekoliko svojstava, metoda ili događaja novih objekata.

## *Svojstva xml objekta*

### `async` svojstvo

Kao što je već pomenuto ovo svojstvo određuje način učitavanja fajla – da li je to sinhrono učitavanje ili asihrono. Vrednosti su `true` (inicijalno postavljena vrednost) ili `false`.
Na primer:

```
xmlDoc.async=false
```

### `attributes` svojstvo

Ovo svojstvo sadrži indeksiranu listu atributa za navedeni čvor. Za naš primer, da bismo dobili vrednost atributa id koji je dat u primeru trebamo napisati sledeće:

```
xmlDoc.documentElement.childNodes(0).childNodes(1).childNodes(0).attrib
utes(0).text
```

### `baseName` svojstvo

Ovo svojstvo je u tesnoj vezi sa pojmom *namespace* (opseg važenja u nekom opštem slučaju). Na primer kada u xml dokumentu opisujete dve različite stvari – automobilski točak i točak od bicikla koje imaju isto ime – točak, potreban je način da razlikujete ove dve stvari u zavisnosti od konteksta – da li se radi o automobilu ili se radi o biciklu. Tada

možete uvesti namespace (što predstavlja na izvestan način rečnik koji govori parseru o mogućim kontekstima – automobil ili bicikl) :

```
<TOCAK xmlns:auto="urn:autodelovi:tocak"
       xmlns:bicikl="urn:delovibicikla:tocak">
        <auto:TOCAK>autotocak</auto:TOCAK>
        <bicikl:TOCAK>bicikltocak</bicikl:TOCAK>
 </TOCAK>
```

U navedenom primeru postoje dva namespace-a: `autodelovi` i `delovibicikla`. Kada je reč o točku za bicikl onda se na njega referišemo sa `<bicikl:TOCAK>` a kada je reč o automobilskom točku onda se na njega referišemo sa `<auto:TOCAK>`. Da bismo dobili ime taga u oba slučaja koristimo `baseName` svojstvo koje nam daje `TOCAK`.
Na primer pošto imamo dva elementa i oba su TOCAK za ime prvog elementa trebamo napisati:

```
ImePromenljiveNPR = xmlDoc.documentElement.childNodes(0).baseName
```

a da bismo dobili ime namespace-a treba nam vrednost `prefix` svojstva:

```
ImePromenljiveNPR = xmlDoc.documentElement.childNodes(0).prefix
```

Daje `auto`

```
ImePromenljiveNPR = xmlDoc.documentElement.childNodes(1).prefix
```

Daje `bicikl`

dok nam svojsto `namespaceURI` daje `urn:autodelovi:tocak` za prvi element a za drugi element `urn:delovibicikla:tocak`.

Sva tri svojstva su po tipu string i samo su za čitanje.

## `childNodes` svojstvo

Ovo svojstvo daje listu čvorova koji imaju podčvorove. Na primer ako za primer sa članovima napišete:

```
objListaCvorova = xmlDoc.documentElement.childNodes
```

dobićete listu čvorova (koja je objekat) koji sadrži tri elementa sa kojima možete dalje raditi pomoću svojstava i metoda objekta **`IXMLDOMNodeList`**:

**`item(integer) metoda`** – metoda koja daje navedeni (integer) element liste. Npr. ako napišete:
`xmlDoc.documentElement.childNodes(0).childNodes.item(1).xml`

u podvučenom delu koda kreirate listu i preko svojstva `childNodes` dobijate xml sadržaj drugog elementa u listi:

```
<CLAN>
    <IME id="3">Mika</IME>
    <PREZIME>Mikic</PREZIME>
    <EMAIL>mika@provider.com</EMAIL>
    <ADRESE>
        <PRVAADRESA>Prva ulica1</PRVAADRESA>
        <DRUGAADRESA>Druga ulica2</DRUGAADRESA>
    </ADRESE>
</CLAN>
```

Kao što možete videti iz priloženog XMLDOM je maksimalno fleksibilan i osim XMLDOMDocument objekta sadrži i druge objekte navedene u poglavlju **XMLDOM Objekti**.

**`length` svojstvo** – daje broj čvorova u kolekciji. Na primer:

```
xmlDoc.documentElement.childNodes(0).childNodes.length
```

daje za navedeni primer 3, odnosno broj elemenata u listi.

**`nextNode` metoda** – daje sledeći čvor u kolekciji. Na primer:

```
Set ListaObjekta = xmlDoc.getElementsByTagName("CLAN")
    For i = 0 To (ListaObjekta.length - 1)
        Set objCvor = ListaObjekta.nextNode
        MsgBox objCvor.Text
    Next
```

će dati redom sadržaj svih čvorova u listi.
**`reset` metoda** – ova metoda resetuje iteraciju kroz kolekciju čvorova u listi. Na primer:

```
Set ListaObjekata = xmlDoc.getElementsByTagName("CLAN")
For i = 0 To (ListaObjekata.length - 1)
  Set objCvor = ListaObjekata.nextNode
  MsgBox objCvor.Text
Next

ListaObjekata.Reset
Set objCvor = ListaObjekata.nextNode
MsgBox objCvor.Text
```

## `dataType` svojstvo

Ovo svojstvo određuje tip podataka za dati čvor. Na primer:

```
TipPodataka = oXMLDOMNode.dataType
objXMLDOMNode.dataType = TipPodataka
```

Tip podataka za dati čvor je u vezi sa `nodeType` svojstvom XMLDOMNode objekta. Ono je ili može biti:

```
NODE_ATTRIBUTE
NODE_CDATA_SECTION
NODE_COMMENT
NODE_DOCUMENT
NODE_DOCUMENT_FRAGMENT
NODE_DOCUMENT_TYPE
NODE_ENTITY
NODE_NOTATION
NODE_PROCESSING_INSTRUCTION
NODE_TEXT
NODE_ELEMENT
NODE_ENTITY_REFERENCE
```

## `definition` svojstvo

Ovo svojstvo daje definiciju čvora u DTD šemi. U pitanju je objekat koji predstavlja čvor na koji se referencira definicija. Može biti nešto od sledećeg:

```
NODE_ENTITY_REFERENCE
NODE_ENTITY
NODE_ATTRIBUTE
NODE_ELEMENT
NODE_CDATA_SECTION
NODE_COMMENT
NODE_DOCUMENT
NODE_DOCUMENT_FRAGMENT
NODE_DOCUMENT_TYPE
NODE_NOTATION
NODE_PROCESSING_INSTRUCTION
NODE_TEXT
```

## `doctype` svojstvo

Sadrži čvor (objekat) tipa dokumenta koji određuje DTD za dokument. Tako na primer: XML dokument (primer.xml) sadrži

```
<?xml version="1.0"?>
<!DOCTYPE root SYSTEM "primer.dtd" [
<!ELEMENT root (description,dsig)>
<!ATTLIST root photoid ENTITY #REQUIRED>
<!NOTATION gif SYSTEM "http://www.microsoft.com">
```

```
<!ENTITY slika SYSTEM "nekaslika.gif" NDATA gif>
]>
<root photoid="slika">
    <description>
        Ovo je ostatak XML DOM dokumenta...
        <foo>&bar;</foo>
    </description>
    <dsig value="172631"/>
</root>
```

DTD dokument (primer.dtd) sadrži:

```
<!-- ovaj DTD opisuje XML tagove -->

<!ELEMENT description (#PCDATA | foo)*>
<!ELEMENT foo (#PCDATA)>
<!ELEMENT dsig (#PCDATA)>
<!ATTLIST dsig value CDATA #IMPLIED>
<!ATTLIST dsig crypt CDATA "128">
<!ENTITY bar SYSTEM "primer.ent">
```

Spolji entitet (primer.ent) sadrži :

```
<!--spoljasnji_DOM_entitet-->
Ovo je spoljni entitet ucitan iz primer.dtd-a. To je
uradjeno tako sto je deklarisan entitet po imenu bar sa
SYSTEM literalom koji pokazuje na fajl koji sadrzi ovaj
tekst.
```

Onda će sledeći VB kod:

```
    Dim xmlDoc As New DOMDocument30
    xmlDoc.async = False
    xmlDoc.Load ("C:\primer.xml")
    MsgBox (xmlDoc.doctype.Name)
```

u MessageBox-u prikazati - root- što u stvari predstavlja ime čvora u xml dokumentu. Možete probati da `name` svojstvo zamenite sa `xml` svojstvom da biste videli rezultat koji možda očiglednije ilustruje ovo svojstvo.

## `documentElement` svojstvo

Ovo svojstvo daje čvor (objekat) koji je najviši u hijerarhiji XML dokumenta. Ovo svojstvo vraća sve što se nalazi ispod definicije XML doumenta –

(<?xml version="1.0"?>).

## firstChild svojstvo

Sadrži prvi potomak u hijerarhiji XML dokumenta. Za XML fajl sa sledećim sadržajem:

```
<?xml version="1.0"?>
<PRIMER>

            <CLAN>
                  <IME id="3">Mika</IME>
                  <PREZIME>Mikic</PREZIME>
                  <EMAIL>mika@provider.com</EMAIL>
                  <ADRESE>
                        <PRVAADRESA>Prva ulica1</PRVAADRESA>
                        <DRUGAADRESA>Druga ulica2</DRUGAADRESA>
                  </ADRESE>
            </CLAN>
            <CLAN>
                  <IME>Zika</IME>
                  <PREZIME>Zikic</PREZIME>
                  <EMAIL>zika@provider.com</EMAIL>
            </CLAN>
</PRIMER>
```

ovo svojstvo kroz sledeći VB kod:

```
  Dim xmlDoc As New DOMDocument30
     xmlDoc.async = False
     xmlDoc.Load ("C:\primer.xml")
     MsgBox (xmlDoc.documentElement.firstChild.xml)
```

daje sledeći xml:

```
<CLAN>
      <IME>Pera</IME>
      <PREZIME>Peric</PREZIME>
      <EMAIL>pera@provider.com</EMAIL>
</CLAN>
```

## implementation svojstvo

Ovo svojstvo vraća **XMLDOMImplementation** objekat za dati dokument.

## lastChild svojstvo

Vraća poslednji čvor u listi čvorova. Za prethodni primer korz sledeći VB kod:

```
Dim xmlDoc As New DOMDocument30
xmlDoc.async = False
xmlDoc.Load ("C:\primer.xml")
MsgBox (xmlDoc.documentElement.lastChild.xml)
```

ovo svojstvo vraća:

```
<CLAN>
      <IME>Zika</IME>
      <PREZIME>Zikic</PREZIME>
      <EMAIL>zika@provider.com</EMAIL>
</CLAN>
```

## `namespaceURI` svojstvo

Vraća URI za namespace. Ovo svojstvo je već objašnjeno u delu članka gde se objašnjavaju `baseName` i `prefix` svojstva.

## `nextSibling` svojstvo

Sadrži sledeći čvor na istom nivou hijerarhije. Za sledeći xml dokument:

```
<?xml version="1.0"?>
<PRIMER>
      <CLANOVI>
            <CLAN>
                  <IME>Pera</IME>
                  <PREZIME>Peric</PREZIME>
                  <EMAIL>pera@provider.com</EMAIL>
            </CLAN>
            <CLAN>
                  <IME id="3">Mika</IME>
                  <PREZIME>Mikic</PREZIME>
                  <EMAIL>mika@provider.com</EMAIL>
                  <ADRESE>
                        <PRVAADRESA>Prva ulica1</PRVAADRESA>
                        <DRUGAADRESA>Druga ulica2</DRUGAADRESA>
                  </ADRESE>
            </CLAN>
            <CLAN>
                  <IME>Zika</IME>
                  <PREZIME>Zikic</PREZIME>
                  <EMAIL>zika@provider.com</EMAIL>
            </CLAN>
      </CLANOVI>
</PRIMER>
```

sledeći VB kod:

```
     Dim xmlDoc As New DOMDocument30
    xmlDoc.async = False
    xmlDoc.Load ("C:\primer.xml")
    MsgBox _
(xmlDoc.documentElement.childNodes(0).childNodes(0).nextSibling.xml)
```

daje sledeći xml:

```
<CLAN>
      <IME id="3">Mika</IME>
      <PREZIME>Mikic</PREZIME>
      <EMAIL>mika@provider.com</EMAIL>
      <ADRESE>
            <PRVAADRESA>Prva ulica1</PRVAADRESA>
            <DRUGAADRESA>Druga ulica2</DRUGAADRESA>
      </ADRESE>
</CLAN>
```

## nodeName svojstvo

Vraca ime za naznačen čvor. Tako VB kod za prethodni XML dokument:

```
    Dim xmlDoc As New DOMDocument30
    xmlDoc.async = False
    xmlDoc.Load ("C:\primer.xml")
    MsgBox _
(xmlDoc.documentElement.childNodes(0).childNodes(0).nodeName)
```

daje:

```
CLAN
```

## nodeType svojstvo

Određuje XML DOM tip čvora, koji određuje ispravne vrednosti i da li čvor može imati potomke. Za prethodni xml dokument sledeći VB kod:

```
    Dim xmlDoc As New DOMDocument30
    xmlDoc.async = False
    xmlDoc.Load ("C:\primer.xml")
    MsgBox (xmlDoc.documentElement.nodeType)
```

daje 1.

Ovo je lista mogućih vrednosti:

```
NODE_ELEMENT (1)
NODE_ATTRIBUTE (2)
NODE_TEXT (3)
NODE_CDATA_SECTION (4)
NODE_ENTITY_REFERENCE (5)
NODE_ENTITY (6)
NODE_PROCESSING_INSTRUCTION (7)
NODE_COMMENT (8)
NODE_DOCUMENT (9)
NODE_DOCUMENT_TYPE (10)
NODE_DOCUMENT_FRAGMENT (11)
NODE_NOTATION (12)
```

## `nodeTypedValue` svojstvo

Sadrži vrednost čvora izraženu u njegovom definisanom tipu podataka. Moguće vrednosti su:

```
NODE_ATTRIBUTE
NODE_CDATA_SECTION
NODE_COMMENT
NODE_DOCUMENT
NODE_DOCUMENT_FRAGMENT
NODE_DOCUMENT_TYPE
NODE_ENTITY
NODE_NOTATION
NODE_PROCESSING_INSTRUCTION
NODE_ELEMENT
NODE_ENTITY_REFERENCE
NODE_TEXT
```

## `nodeTypeString` svojstvo

Ovo svojstvo vraća tip čvora u tekstualnom formatu. Moguće vrednosti su:

```
NODE_ELEMENT (1)
NODE_ATTRIBUTE (2)
NODE_TEXT (3)
NODE_CDATA_SECTION (4)
NODE_ENTITY_REFERENCE (5)
NODE_ENTITY (6)
NODE_PROCESSING_INSTRUCTION (7)
NODE_COMMENT (8)
```

```
NODE_DOCUMENT (9)
NODE_DOCUMENT_TYPE (10)
NODE_DOCUMENT_FRAGMENT (11)
NODE_NOTATION (12)
```

## `nodeValue` svojstvo

Sadrži tekst koji je združen sa čvorom. Može biti nešto od sledećeg:

```
NODE_ATTRIBUTE
NODE_CDATA_SECTION
NODE_COMMENT
NODE_DOCUMENT
NODE_DOCUMENT_TYPE
NODE_DOCUMENT_FRAGMENT
NODE_ELEMENT
NODE_ENTITY
NODE_ENTITY_REFERENCE
NODE_NOTATION
NODE_PROCESSING_INSTRUCTION
NODE_TEXT
```

## `ondataavailable` svojstvo

Ovo svojstvo određuje *eventhandler* (funkciju ili proceduru) za `ondataavailable` događaj.

## `onreadystatechange` svojstvo

Ovo svojstvo određuje *eventhandler* (funkciju ili proceduru) koja se poziva pri promeni `readyState` svojstva.

## `ontransformnode` svojstvo

Ovo svojstvo određuje *eventhandler* (funkciju ili proceduru) za `ontransformnode` događaj.

## `ownerDocument` svojstvo

Vraća koren dokumenta koji sadrži dati čvor.

## `parentNode` svojstvo

Sadrži čvor koji se nalazi iznad u hijerahiji (za čvorove koji mogu imati nadčvorove). Objekat po strukturi.

## `parsed` svojstvo

Sadrži true ako je čvor i svi podčvorovi parsiran i instanciran ili false ako neki od čvorova tek treba da bude parsiran.

## `parseError` svojstvo

Vraća **XMLDOMParseError** objekat koji sadrži  informacije o poslednjoj grešci pri parsiranju.

## `prefix` svojstvo

Vraća namespace prefiks.

## `preserveWhiteSpace` svojstvo

Sadrži true ako inicijalno procesiranje čuva whitespace (prazan prostor) i false u suprotnom.

## `previousSibling` svojstvo

Sadrži prethodni čvor na istom nivou hijerarhije. Objekat po strukturi.

## `readyState` svojstvo

Naznačuje trenutno stanje XML dokumenta.

## `resolveExternals` svojstvo

Naznačuje da li spoljašnje definicije (namespace, DTD spoljašnji podsetovi, i spoljašnje reference entiteta) trebaju da budu razrešene prilikom parsiranja a nezavisno od validacije.

## `specified` svojstvo

Naznačuje da li je čvor eksplicitno naznačen ili je izveden iz inicijalne vrednosti u DTD dokumentu ili šemi.

## `text` svojstvo

Sadrži tekst čvora i njegovih podčvorova.

## `url` svojstvo

Vraća URL poslednjeg učitanog XML dokumenta.

## `validateOnParse` svojstvo

Naznačuje da li parser treba da validira dokument prilikom parsiranja.

## `xml` svojstvo

Sadrži XML čvora i svih njegovih podčvorova.

U sledećem članku će biti opisane metode i događaji za XMLDOM.