

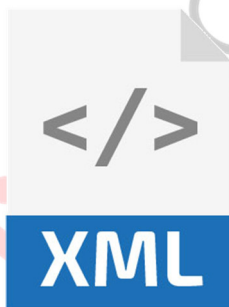
# Anatomija XML-a

Priča o radu sa podacima kojima rukuju web aplikacije započne upoznavanjem jednog posebnog jezika – XML-a. Reč je o jednom od najpoznatijih i najznačajnijih jezika koji se koriste za predstavljanje podataka u tekstualnom obliku na webu. Tome u prilog ide i činjenica da su iz XML-a izvedeni i brojni drugi jezici: XHTML, WSDL, GPX, RSS, RDF, OWL, SMIL, XSLT...

U modulu pred vama prvo će biti izneta osnovna sintaksna načela XML jezika, a zatim će biti prikazano njegovo korišćenje iz ugla frontend programiranja.

## Šta je XML?

XML je skraćenica za **eXtensible Markup Language**. Reč je o jeziku za obeležavanje, odnosno markiranje (*markup language*), koji je po svojim osobinama veoma sličan osnovnom jeziku za kreiranje strukture web stranica – jeziku HTML. Ipak, XML ne poseduje unapred utvrđen skup oznaka, već tvorac dokumenta ima mogućnost da samostalno definiše proizvoljne oznake za markiranje teksta.



Slika 1.1. eXtensible Markup Language

### Šta je jezik za markiranje?

Jezik za markiranje (*markup language*) definiše skup pravila za predstavljanje podataka u tekstualnom obliku. Tako jezici za obeležavanje omogućavaju kreiranje dokumenata unutar kojih je različite delove tekstualnog sadržaja moguće markirati specijalnim oznakama:

```
<message>
  <text>Hello World!</text>
</message>
```

U primeru je korišćenjem specijalnih oznaka `<message>...</message>` i `<text>...</text>` obavljeno markiranje teksta *Hello World!*. Za nekoga ko čita ovakav dokument, tekst *Hello World!* sada će imati posebno značenje. Čitalac će znati da je reč o tekstu poruke, baš zato što je takvim oznakama unutar dokumenta obavljeno markiranje teksta.

## XML nije programski jezik

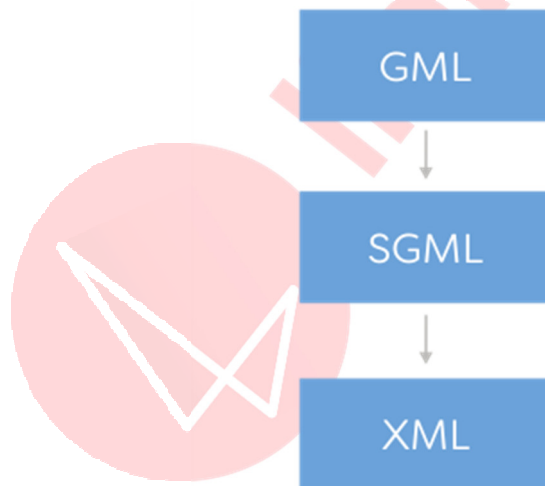
Bitno je već na početku naglasiti da, baš kao ni HTML i CSS, ni XML nije programski, već opisni jezik. Naime, programski jezici omogućavaju kreiranje kompjuterskih programa. Takvi programi predstavljaju instrukcije na osnovu kojih kompjuter obavlja određene zadatke. Stoga programski jezici omogućavaju da se logika za rešavanje nekog problema pretoči u izvorni kod kojim se upravlja hardverom računara.

Sa druge strane, jezici za obeležavanje se koriste za strukturiranje podatka. Njihovim korišćenjem je moguće podatke iz nekog programa jednostavno i brzo pretvoriti u strukturirani, tekstualni oblik, a zatim ih sačuvati ili proslediti nekom drugom programu.

Može se zaključiti da je XML jezik za obeležavanje koji omogućava da se podaci lako predstave u tekstualnom obliku, kako bi se sačuvali ili podelili. Sam proces konvertovanja podataka u format koji omogućava lako čuvanje i deljenje drugačije se naziva **serijalizacija** podataka.

## Razvoj XML-a

Istorija XML-a počinje još 60-ih godina prošlog veka, kada je u IBM-u u konstruisan prvi višenamenski jezik za serijalizaciju podataka – GML. Nastavak razvoja jezika za serijalizaciju rezultovao je nastankom jezika SGML, naslednika jezika GML. 1996. godine započet je rad na uprošćenoj verziji SGML-a. Rezultat procesa uprošćavanja bio je stvaranje XML-a, koji status W3C Recommendation dobija 1998. godine.



Slika 1.2. Razvojni put XML-a

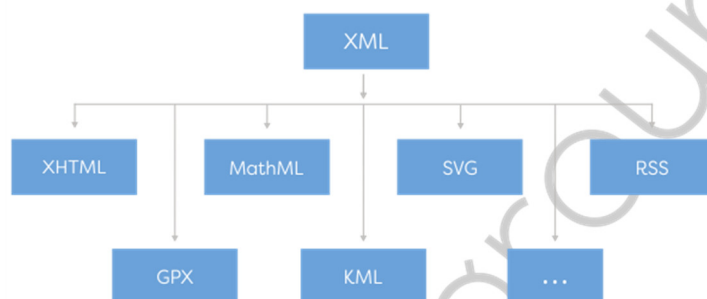
Do danas su kreirane dve verzije XML-a:

- XML 1.0
- XML 1.1

Obe verzije su tokom vremena prošle kroz nekoliko faza, tako da su danas aktuelni peto izdanje (*Fifth Edition*) verzije 1.0 i drugo izdanje (*Second Edition*) verzije 1.1.

Bitno je razumeti da verzija 1.1 nije naslednik verzije 1.0, već su tokom vremena obe verzije paralelno razvijane. Verzija 1.1 je uvedena kako bi se obradili neki vrlo specifični slučajevi korišćenja, pre svih oni koji podrazumevaju upotrebu nestandardnih Unicode karaktera. Danas je opšteprihvaćena verzija 1.0; većina web pregledača i parsera razume isključivo tu verziju jezika.

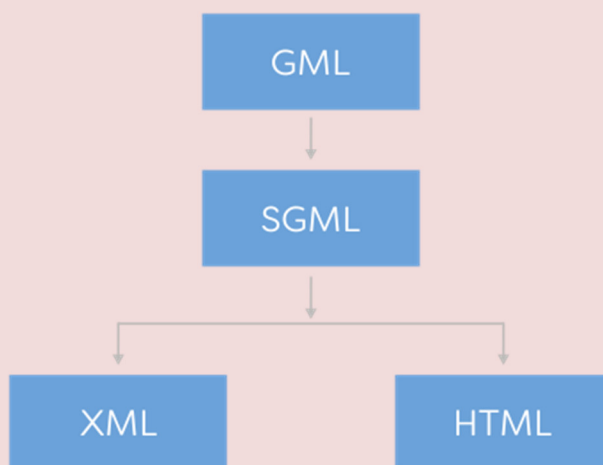
XML je poslužio za kreiranje brojnih drugih jezika koji se koriste za serijalizaciju podataka specifične namene. Neki od najpoznatijih jezika koji su izvedeni iz XML-a su: XHTML, MathML, SVG, RSS...



Slika 1.3. Jezici zasnovani na XML-u

## HTML i XML

Sličnost između jezika XML i HTML nije slučajna. Oba jezika izvedena su iz jezika SGML (slika 1.4).



Slika 1.4. XML i HTML zasnovani su na istom jeziku

## Zbog čega nam je potreban XML?

XML omogućava da se u tekstualnom formatu veoma lako prikažu podaci raznih tabela, adresara, konfiguracionih parametara, finansijskih transakcija, tehničkih crteža, navigacionih ruta... Ograničenja praktično nema i može se reći da se broj jezika zasnovanih na XML-u konstantno uvećava.

XML je standardizovan od strane organizacije World Wide Web Consortium (W3C), što osigurava da svi zainteresovani korisnici poštuju i koriste identičan skup pravila. Sve to na kraju omogućava da se XML koristi za čuvanje i razmenu podataka od strane programa bez obzira na to kojim jezikom ili skupom jezika su napisani. Tako je čuvanje podataka u XML formatu jedna od alternativa korišćenju baza podataka kao specijalizovanih sistema koji pored čuvanja, obezbeđuju i brojne druge, napredne osobine. U nekim situacijama, upotreba baza podataka može predstavljati suviše komplikovano i nepotrebno rešenje, pogotovo ukoliko je cilj sačuvati relativno jednostavne podatke. Primer mogu biti konfiguracioni fajlovi. Veoma česta je praksa da mnoge web, ali i desktop aplikacije za čuvanje podešavanja koriste upravo XML fajlove.

XML poseduje i veliki broj drugih namena. Njime, odnosno njegovom izvedenom verzijom – SVG-om, predstavlja se vektorska grafika. Dalje, navigacione rute, putanje i GPS podaci, koje koriste različite aplikacije za navigaciju i prikaz geografskih mapa, predstavljaju se korišćenjem još dva jezika bazirana na XML-u – GPX i KML.

Web sajtovi i aplikacije koje samostalno kreiramo mogu koristiti XML za obavljanje veoma širokog spektra poslova – od jednostavnog čuvanja podataka u XML fajlovima, preko razmene XML podataka sa HTTP serverima i drugim aplikacijama, pa sve do konzumiranja XML podataka raznih nezavisnih web servisa (vremenskih prognoza, berzanskih podataka...).

### Pitanje

XML je nastao iz jezika:

**a) SGML**

- a) HTML
- b) RSS
- c) XHTML

### Objašnjenje

*XML je nastao kao produkt uprošćavanja jezika SGML, 1998. godine. SGML je, opet, nastao uprošćavanjem jezika GML, koji je osnovni rodonačelnik jezika za markiranje. RSS i XHTML su varijante XML jezika, dok je HTML osnovni jezik za kreiranje strukture web stranica, nastao po ugledu na SGML.*

## Struktura XML dokumenta

Po svojoj strukturi, XML je vrlo sličan HTML-u. Njegovu okosnicu čine tagovi, elementi, atributi i podaci. Kompletan spisak različitih XML pojmova sa kojima ćemo se u nastavku ove lekcije upoznati je sledeći:

- XML deklaracija
- elementi
- atributi
- podaci
- CDATA sekcije
- prostori imena
- komentari
- instrukcije za procesiranje

### Referentni primer XML dokumenta

XML kod piše se unutar dokumenata sa ekstenzijom **.xml**. Stoga je prvi praktičan korak u ovoj lekciji kreiranje jednog fajla sa proizvoljnim nazivom i **.xml** ekstenzijom. Unutar takvog fajla je potrebno smestiti sledeći XML kod:

```
<recipe>
  <title>Grilled Cheese Sandwich</title>
  <ingredients>
    <ingredient qty="2">bread slice</ingredient>
    <ingredient>cheese slice</ingredient>
    <ingredient qty="2">margarine pat</ingredient>
  </ingredients>
</recipe>
```

Primer ilustruje XML dokument koji opisuje jedan recept. Odmah se može primetiti sličnost sa HTML jezikom, pošto se u prikazanom dokumentu koriste tagovi, atributi i sadržaj. Ipak, za razliku od HTML-a, kod koga se moraju koristiti unapred utvrđeni tagovi kao što su `<html>`, `<head>`, `<img>` i drugi, prilikom kreiranja upravo prikazanog XML dokumenta korišćeni su tagovi koje smo mi samostalno definisali, specijalno za opisivanje recepata: `<recipe>`, `<ingredients>`...

### Tagovi i elementi

Tag je osnovni pojam XML jezika koji se koristi za markiranje podataka. U XML-u postoje tri različite vrste tagova:

- otvarajući tagovi
- zatvarajući tagovi
- samozatvarajući tagovi

**Otvarajući tag** se kreira korišćenjem karaktera veće i manje (`<` `>`) između kojih se navodi naziv taga:

```
<ingredient>
```

**Zatvarajući tag** kreira se na identičan način kao i otvarajući, uz jednu malu razliku. Unutar zatvarajućeg taga, ispred naziva se postavlja karakter kosa crta (*slash*): `/`:

```
</ingredient>
```

**Samozatvarajući tag** se kreira navođenjem karaktera veće i manje, između kojih se navodi najpre naziv taga, a nakon naziva, razmak i karakter kosa crta /:

```
<ingredient />
```

### Pravila za imenovanje tagova

Imenovanje XML tagova podleže sledećim sintaksnim pravilima:

- nazivi tagova su osetljivi na mala i velika slova – tako su *ingredient* i *Ingredient* dva različita naziva;
- nazivi tagova mogu biti sastavljeni iz slova, brojeva i karaktera donja crta, srednja crta i tačka;
- nazivi tagova moraju započeti slovom ili karakterom donja crta;
- nazivi tagova ne smeju započeti tekстом *xml*, *XML*, *Xml*, odnosno bilo kojom kombinacijom velikih i malih slova u nizu *xml*;
- nazivi tagova ne smeju sadržati razmake.

Tagovi se koriste za kreiranje **XML elemenata**:

```
<ingredient>cheese slice</ingredient>
```

Prikazani kod ilustruje jedan XML element. Element je sačinjen iz otvarajućeg i zatvarajućeg taga, između kojih se nalazi sadržaj elementa (slika 1.5).

opening tag      data      closing tag

**<ingredient>cheese slice</ingredient>**

Slika 1.5. Struktura jednog XML elementa

Ukoliko XML element ne poseduje sadržaj, on se u takvoj situaciji može kreirati na dva načina. Prvi način jeste definisanje otvarajućeg i zatvarajućeg taga između kojih nema ničega:

```
<ingredient></ingredient>
```

Drugi način za kreiranje XML elementa bez sadržaja jeste korišćenje samozatvarajućeg taga:

```
<ingredient />
```

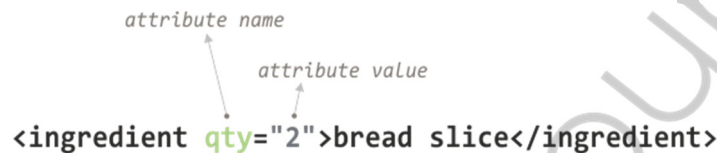
Oba prikazana primera proizvode identičan efekat – kreiraju jedan XML element koji nema sadržaj. Ukoliko se sada pitate koja je svrha XML elemenata bez sadržaja, odgovor je vrlo jednostavan: iako ovakvi XML elementi ne poseduju sadržaj direktno između tagova, oni mogu biti dekorisani jednim posebnim XML pojmom koji omogućava da se nad elementima definišu dodatni podaci. Reč je o atributima.

## Atributi

Atributi omogućavaju da se nad XML elementima definišu neki dodatni podaci. Svaki XML element na sebi može imati proizvoljan broj atributa. Ipak, atributi se uvek definišu isključivo na otvarajućim ili samozatvarajućim tagovima. Naš referentni primer, definisan nešto ranije, poseduje dva elementa sa XML atributima:

```
<ingredient qty="2">bread slice</ingredient>
<ingredient qty="2">margarine pat</ingredient>
```

Iz primera se može videti da se XML atributi definišu u obliku parova ključeva i vrednosti, odnosno naziva atributa i njihovih vrednosti, između kojih stoji karakter jednako (=).



`<ingredient qty="2">bread slice</ingredient>`

Slika 1.6. Struktura XML atributa

Vrednosti atributa moraju se postaviti ispod jednostrukih ili dvostrukih navodnika.

Tag `<ingredient>` sa slike 1.6. poseduje atribut `qty`, što je skraćenica od reči *quantity* (količina). Atributi obezbeđuju mehanizam za navođenje dodatnih informacija o elementima, pa tako atribut `qty` definiše količinu sastojka (*dve kriške hleba*).

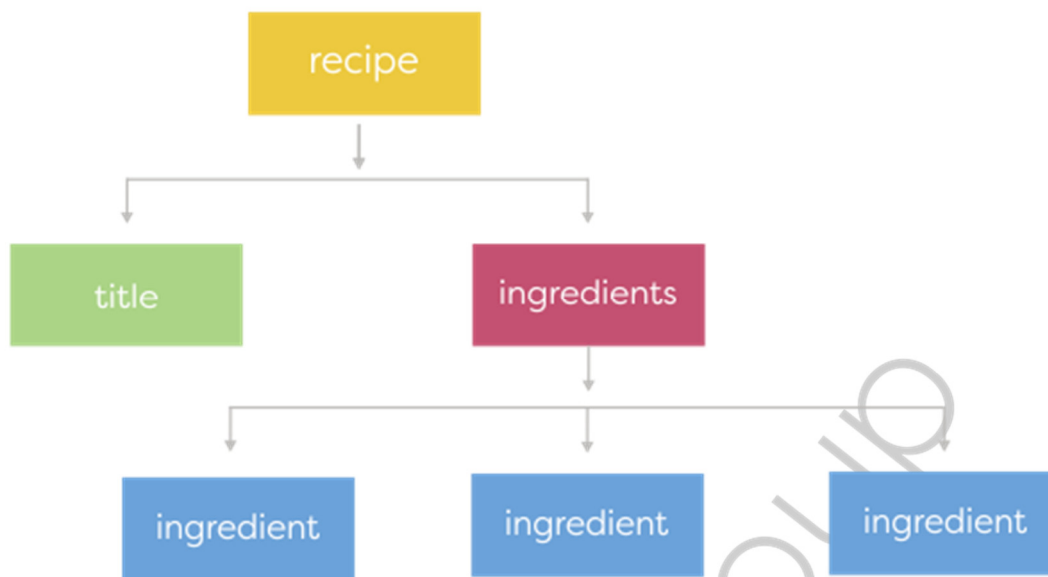
Na početku ovog poglavlja o atributima rečeno je da se atributi mogu naći i na elementima koji se kreiraju korišćenjem samozatvarajućeg taga:

```
<ingredient code="E333" />
```

Sada element `ingredient` ne poseduje podatke, ali zato ima jedan XML atribut koji ga dodatno opisuje. Reč je o atributu `code`, kojim se definiše kod sastojka.

## Stablo XML elemenata

Analizom referentnog primera može se primetiti još jedna značajna osobina XML jezika – pored jednostavnih tekstualnih podataka, kao što su *bread slice* ili *margarine pat*, jedan XML element kao svoj sadržaj može imati i druge XML elemente. U našem primeru, elementi `title` i `ingredients` smešteni su unutar elementa `recipe`. Dalje, unutar elementa `ingredients`, nalaze se tri nova elementa `ingredient`. Sve ovo praktično znači da XML omogućava smeštanje jednog XML elementa unutar nekog drugog, čime se formira stablo XML elemenata (slika 1.7).



Slika 1.7. Stablo XML elemenata

Slika 1.7. ilustruje stablo XML elemenata koje se dobija nešto ranije prikazanim referentnim primerom. Pregledom stabla XML elemenata može se uvideti nekoliko značajnih osobina ovog jezika:

- element koji se nalazi na vrhu XML stabla drugačije se naziva koreni element (*root element*); u primeru je to element `recipe`;
- XML dokumenti mogu imati samo jedan koreni element;
- svaki element osim korenog ima svog roditelja (*parent*); na primer, roditelj elemenata `title` i `ingredients` je `recipe`;
- jedan element može imati proizvoljan broj potomaka (*children*);
- XML element ne mora da ima potomke;
- roditelji i potomci međusobno grade jednu granu (*branch*);
- poslednji element u jednoj grani se naziva list (*leaf*); takvi su, na primer, `ingredient` elementi.

## XML deklaracija

XML dokumenti mogu započeti specijalnom oznakom koja informiše XML parser da je reč o XML dokumentu i definiše neke specifične instrukcije za procesiranje. Reč je o oznaci koja se naziva XML deklaracija, a negde se može čuti i naziv XML prolog. Odsustvo takve oznake na početku XML dokumenta za opis recepta koji je dat u ovoj lekciji govori o tome da XML deklaracija nije obavezna. Ipak, kada se navede, to se mora učiniti na samom početku dokumenta, pre bilo kog drugog XML koda.

Minimalni oblik XML deklaracije izgleda ovako:

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



## Instrukcije za procesiranje

XML deklaracija je jedan primer instrukcije za procesiranje. Reč je o instrukcijama koje su namenjene programima koji će obavljati čitanje sadržaja XML dokumenta (XML parsers). Instrukcije su obično specifične, odnosno razumljive samo aplikacijama koje će parsirati sadržaj takvih instrukcija.

Instrukcije za parsiranje započinju karakterima `<?`, a završavaju se karakterima `?>`. Opšti oblik instrukcije za procesiranje izgleda ovako:

- `<?target instructions?>`
- `target` – odnosi se na aplikaciju kojoj je instrukcija namenjena;
- `instructions` – predstavlja jednu ili više instrukcija.

Na osnovu uvida u opštu strukturu instrukcije za procesiranje može se zaključiti da se nešto ranije navedena XML deklaracija može rastumačiti ovako: reč je o instrukciji za procesiranje koja se upućuje aplikacijama koje čitanju XML kod i tom prilikom se takvim aplikacijama stavlja do znanja da se u dokumentu koristi 1.0 verzija jezika.

XML deklaracija nije jedina instrukcija za procesiranje. Još jedna instrukcija za procesiranje koja se često koristi izgleda ovako:

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

Reč je o instrukciji za procesiranje koja je namenjena programima za stilizaciju XML-a. Njom se takvim programima govori gde se nalazi stilizacija XML dokumenta. O stilizovanju XML dokumenata biće više reči u nastavku lekcije.

Analizom XML deklaracije može se zaključiti da ona poseduje jedan atribut – **version**. Reč je o obaveznom atributu. Drugim rečima, ukoliko se XML deklaracija navede, mora se definisati i atribut `version`. Atribut `version` se koristi za identifikovanje verzije XML-a koja se koristi u dokumentu.

Pored atributa `version`, XML deklaracija može imati i atribut **encoding**:

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

Atributom `encoding` definiše se način na koji se karakteri prevode u svoj binarni oblik. Jednostavnije rečeno, vrednost atributa `encoding` može uticati na to koje karaktere ćemo biti u mogućnosti da koristimo prilikom pisanja XML dokumenta. Tako nešto posebnu važnost ima kada se u dokumentu koriste neki specifični karakteri koji postoje u pojedinim jezicima (na primer Ä, ß, Š, Đ, Ç...). Srećom, podrazumevana vrednost atributa `encoding` je UTF-8, što praktično znači da je i bez definisanja ovog atributa, unutar XML dokumenta moguće upotrebiti gotovo sve specifične karaktere svih svetskih jezika.

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

Linija ilustruje XML deklaraciju kojom je `encoding` eksplicitno postavljen na vrednost `UTF-8`. Ipak, kao što je rečeno, tako nešto nije obavezno.

### Napomena

Ukoliko želite da vidite kako izgleda neadekvatno prikazivanje specifičnih karaktera, na početak XML dokumenta sa takvim karakterima postavite sledeću XML deklaraciju:

```
<?xml version="1.0" encoding="ASCII"?>
```

Na kraju, XML deklaracija može sadržati još jedan opcioni atribut. Reč je o atributu `standalone`. Ovaj atribut koristi se kada se sintaksa XML dokumenta zasniva na Document Type Definition (DTD) dokumentu. Reč je o pojmu koji će biti obrađen u narednoj lekciji, stoga ćemo sada samo reći da vrednost `yes`, atributa `standalone`, ukazuje na to da se DTD nalazi unutar XML fajla. Vrednost `no` govori da je DTD definisan izvan XML fajla, kao zaseban dokument.

## Komentari

XML dokumenti mogu sadržati i komentare. Komentarisani kod znači da parser neće uzeti u obzir taj deo koda. XML komentar se formira na sledeći način:

```
<!-- comment -->
```

Formiranje XML komentara podleže sledećim pravilima:

- XML komentari započinju oznakom `<!--` a završavaju se oznakom `-->`;
- komentari se mogu pojaviti bilo gde nakon XML deklaracije, osim unutar taga;
- jedan komentar se ne može smestiti unutar drugog komentara;
- tekst komentara ne može sadržati karakter srednja crta (-) ni dva uzastopna karaktera srednja crta (--);
- XML komentari se ne smatraju sadržajem.

Evo još jednog primera XML komentara:

```
<!-- myElement tag in use --><myElement atribut="my  
atribut">Content</myElement>
```

Komentar sa tekстом *myElement tag in use* sada je postavljen neposredno pre jednog XML elementa sa nazivom `myElement`.

## Reference na karaktere

Određeni karakteri za sam XML jezik imaju posebno, specijalno značenje. Takvi su, na primer, karakteri za kreiranje tagova (karakter veće i manje: `<` i `>`). Specijalni XML karakteri ne mogu se pojaviti kao sadržaj elemenata ili kao vrednost XML atributa. Na primer, nije moguće postaviti karakter `<` između otvarajućeg i zatvarajućeg taga, jer bi na taj način XML čitač pomislio da je reč o novom tagu.

Rešenje ovog problema jeste upotreba referentnih entiteta, odnosno specijalnih kodova koji predstavljaju reference stvarnih karaktera. Reference na karaktere se mogu podeliti na dve grupe:

- numeričke reference na karaktere kojima se karakteri predstavljaju preko koda koji im je dodeljen u Unicodeu; na primer, `&#0931` predstavlja grčko slovo sigma ( $\Sigma$ );
- unapred definisani entiteti koji predstavljaju zamenu za neke od najkorišćenijih specijalnih karaktera; ovakvih predefinisanih entiteta postoji pet i oni su prikazani tabelom 1.1.

Entitet	Karakter
<code>&amp;lt;</code>	<code>&lt;</code>
<code>&amp;gt;</code>	<code>&gt;</code>
<code>&amp;amp;</code>	<code>&amp;</code>
<code>&amp;apos;</code>	<code>'</code>
<code>&amp;quot;</code>	<code>"</code>

Tabela 1.1. XML entiteti

Problem prikaza specifičnih karaktera unutar XML dokumenata biće ilustrovan sledećim primerom:

```
<expression>6 < 8</expression>
```

Primer ilustruje jedan XML element (`expression`) kojim se predstavlja jedan jednostavan matematički izraz. Izrazom se govori da je broj 6 manji od broja 8. Za formiranje izraza je iskorišćen karakter manje (`<`), što će stvoriti grešku prilikom parsiranja ovakvog dokumenta (slika 1.8).

**This page contains the following errors:**

error on line 3 at column 16: StartTag: invalid element name

**Below is a rendering of the page up to the first error.**

Slika 1.8. Prikaz greške prilikom prikaza XML dokumenta u Chrome pregledaču

Slika 1.8. ilustruje poruku koja se dobija unutar browsera Google Chrome prilikom pokušaja pregleda XML koda kojim je predstavljen upravo prikazani jednostavni matematički izraz. Jasno je da je karakter *manje* neophodno zameniti nekim od specijalnih entiteta koji su upravo prikazani:

```
<expression>6 &#60; 8</expression>
```

Sada je karakter manje (`<`) zamenjen Unicode kodom, pa sintaksna greška više neće biti prisutna.

Prikazani problem može se rešiti na još jedan način:

```
<expression>6 &lt; 8</expression>
```

Umesto Unicodea, sada je upotrebljen unapred definisan entitet koji menja karakter manje (<).

### Napomena

Sintaksa pravila XML jezika kao nevalidne karaktere sadržaja definiše samo karakter manje < i ampersend &. To praktično znači da XML neće moći da bude parsiran samo u slučaju da se jedan od ova dva karaktera pojavi kao deo sadržaja atributa ili elementa. Ipak, i pored ovoga, dobra je praksa da se i ostali specijalni XML karakteri unutar sadržaja predstavljaju korišćenjem upravo prikazanih entiteta.

## CDATA sekcije

Upravo prikazani primer podrazumevao je definisanje jednog specifičnog karaktera koji se u svom izvornom obliku ne može naći kao sadržaj unutar XML dokumenata. Ali, zamislite situaciju u kojoj bi bilo potrebno definisati neki XML sadržaj sa znatno više karaktera koji u XML-u imaju specijalno značenje. Kako se ne bi obavljala pojedinačna zamena svakog takvog karaktera, XML obezbeđuje još jedan pristup kojim je unutar XML dokumenata moguće definisati sadržaj u svom izvornom obliku. Reč je o CDATA sekcijama.

CDATA je skraćenica za pojam *character data*. Reč je, zapravo, o sekciji koju je moguće kreirati unutar XML dokumenta tako da XML parser zna da nije potrebno da parsira sadržaj takve sekcije. Drugim rečima, unutar CDATA sekcije je moguće pisati bilo kakav sadržaj, uključujući i sve specijalne XML karaktere, bez ikakvih problema.

CDATA sekcija se kreira na sledeći način:

```
<![CDATA[
    content
]]>
```

Sve ono što se definiše kao sadržaj prikazane sekcije (`content`) može sadržati specijalne karaktere koji se inače ne mogu naći u izvornom obliku u XML dokumentu.

Kako biste na najbolji način razumeli u kojim situacijama CDATA sekcije mogu biti korisne, biće razmotren sledeći primer:

```
<expression>a < b && b > c && c < a</expression>
```

Prikazani XML element kao svoj sadržaj sada poseduje nekoliko karaktera koji imaju specijalno značenje u XML jeziku. Ukoliko bismo koristili pojedinačno referenciranje karaktera, kod bi izgledao ovako:

```
<expression>a &lt; b &amp;&amp; b &gt; c &amp;&amp; c &lt; a</expression>
```

Jasno je da definisanje reference za svaki od specijalnih karaktera može biti zamorno, a uz to stvara i vrlo nepregledan i nerazumljiv kod. Stoga, CDATA sekcija može pomoći u ovakvoj situaciji:

```
<expression>
  <![CDATA[
    a < b && b > c && c < a
  ]]>
</expression>
```

Zbog upotrebe CDATA sekcije, sada se svi specijalni XML karakteri mogu upotrebiti u svom izvornom obliku, što rezultuje znatno čitljivijim kodom.

## Prostori imena

Prilikom rada sa XML fajlovima, veoma česta praksa može biti da jedna aplikacija konzumira XML podatke iz različitih izvora. Ukoliko se unutar XML koda koji se dobija od različitih izvora upotrebe elementi sa identičnim nazivom, može doći do konflikata imena. Zamislite, na primer, da naša aplikacija konzumira XML podatke o dve različite vrste recepata: onim kulinarским, koje smo već videli, ali i podatke o receptima za pravljenje kozmetičkih preparata:

```
<recipe>
  <title>Grilled Cheese Sandwich</title>
  <ingredients>
    <ingredient qty="2">bread slice</ingredient>
    <ingredient>cheese slice</ingredient>
    <ingredient qty="2">margarine pat</ingredient>
  </ingredients>
</recipe>

<recipe>
  <title>Kombucha Skin Cream</title>
  <ingredients>
    <ingredient qty="15g">beeswax</ingredient>
    <ingredient qty="100ml">olive oil</ingredient>
    <ingredient qty="1tbsp">wheat germ</ingredient>
    <ingredient qty="10ml">calendula oil</ingredient>
    <ingredient qty="2tsp">kombucha tea</ingredient>
    <ingredient qty="1ml">lavender essential oil</ingredient>
  </ingredients>
</recipe>
```

Sada je prvo prikazan jedan kulinarški recept za pravljenje sendviča sa grilovanim sirom, a nakon njega i jedan recept za pravljenje kozmetičkog preparata – kombuča kreme za kožu. Svakako je reč o dva potpuno različita recepta. Ipak, to naša aplikacija neće moći da zna, s obzirom da su za markiranje podataka ovih recepata iskorišćeni elementi identičnih naziva.

U ovakvim situacijama može se upotrebiti pojam prostora imena. Prostori imena omogućavaju da se prevaziđu konflikti imena, u situacijama u kojima se koriste identični nazivi elemenata koji imaju različite namene.

Prostor imena se definiše kao atribut na otvarajućem tagu elementa i to u sledećoj formi:

```
xmlns:prefix="URI "
```

Sintaksa za definisanje prostora imena sačinjena je iz sledećih delova:

- **xmlns** – naziv atributa kojim se definiše prostor imena;
- **:** – karakter dve tačke (:) kojim se razdvajaju `xmlns` i prefiks prostora imena;
- **prefix** – prefiks koji će biti upotrebljavan za identifikaciju prostora imena;
- **URI** – naziv prostora imena, koji se najčešće definiše u formi URL adrese.

Upotreba prostora imena na primeru kulinarskog recepta može da izgleda ovako:

```
<c:recipe xmlns:c="http://www.my-site.com/cooking">
  <c:title>Grilled Cheese Sandwich</c:title>
  <c:ingredients>
    <c:ingredient qty="2">bread slice</c:ingredient>
    <c:ingredient>cheese slice</c:ingredient>
    <c:ingredient qty="2">margarine pat</c:ingredient>
  </c:ingredients>
</c:recipe>
```

Na korenom elementu kulinarskog recepta postavljen je atribut kojim se definiše prostor imena. Prostor imena ima prefiks *c*, čime se aludira na pojam *cooking*. Upravo zbog toga, nazivi svih tagova kulinarskog recepta započinju karakterom *c*, odnosno prefiksom prostora imena.

Po identičnom principu se može definisati i prostor imena za recept kozmetičkog proizvoda:

```
<b:recipe xmlns:b="http://www.my-site.com/beauty">
  <b:title>Kombucha Skin Cream</b:title>
  <b:ingredients>
    <b:ingredient qty="15g">beeswax</b:ingredient>
    <b:ingredient qty="100ml">olive oil</b:ingredient>
    <b:ingredient qty="1tbsp">wheat germ</b:ingredient>
    <b:ingredient qty="10ml">calendula oil</b:ingredient>
    <b:ingredient qty="2tsp">kombucha tea</b:ingredient>
    <b:ingredient qty="1ml">lavender essential oil</b:ingredient>
  </b:ingredients>
</b:recipe>
```

Za prefiks prostora imena recepta kozmetičkog preparata odabrano je slovo *b*, kao skraćenica za reč *beauty*.

Bitno je razumeti da nazivi prostora imena koji se definišu u formi URI-a ne moraju da predstavljaju adresu na kojoj se nalazi neki konkretan dokument. Praksa definisanja naziva u formi URI-a odabrana je prevashodno kako bi se postiglo definisanje jedinstvenih naziva prostora imena. Ipak, kao što je rečeno, nije obavezno da URI zaista i pokazuje na neki resurs. Sa druge strane, najčešća je praksa definisanje URI-a koji ukazuju na dodatni opis tagova iz prostora imena.

U upravo prikazanim primerima, prostori imena su definisani na samim elementima koji predstavljaju recepte. Ipak, česta je praksa da se prostori imena definišu na korenom elementu XML dokumenta:

```
<root xmlns:c="http://www.my-site.com/cooking"
      xmlns:b="http://www.my-site.com/beauty">

  <c:recipe>
    <c:title>Grilled Cheese Sandwich</c:title>
    <c:ingredients>
      <c:ingredient qty="2">bread slice</c:ingredient>
      <c:ingredient>cheese slice</c:ingredient>
      <c:ingredient qty="2">margarine pat</c:ingredient>
    </c:ingredients>
  </c:recipe>
```

```

<b:recipe>
  <b:title>Kombucha Skin Cream</b:title>
  <b:ingredients>
    <b:ingredient qty="15g">beeswax</b:ingredient>
    <b:ingredient qty="100ml">olive oil</b:ingredient>
    <b:ingredient qty="1tbsp">wheat germ</b:ingredient>
    <b:ingredient qty="10ml">calendula oil</b:ingredient>
    <b:ingredient qty="2tsp">kombucha tea</b:ingredient>
    <b:ingredient qty="1ml">lavender essential oil</b:ingredient>
  </b:ingredients>
</b:recipe>

</root>

```

Sada su definicije prostora imena prebačene na koreni element dokumenta. Korišćenje odgovarajućih prefiksa prostora imena na elementima je identično kao i u prethodnim primerima.

### Podrazumevani prostor imena

Kada se unutar jednog XML dokumenta koristi veći broj prostora imena, jedan se može definisati kao podrazumevani i na taj način se postiže čistiji izgled dokumenta. Na našem primeru, to može da izgleda ovako:

```

<root xmlns="http://www.my-site.com/cooking"
  xmlns:b="http://www.my-site.com/beauty">
  <recipe>
    <title>Grilled Cheese Sandwich</title>
    <ingredients>
      <ingredient qty="2">bread slice</ingredient>
      <ingredient>cheese slice</ingredient>
      <ingredient qty="2">margarine pat</ingredient>
    </ingredients>
  </recipe>
  <b:recipe>
    <b:title>Kombucha Skin Cream</b:title>
    <b:ingredients>
      <b:ingredient qty="15g">beeswax</b:ingredient>
      <b:ingredient qty="100ml">olive oil</b:ingredient>
      <b:ingredient qty="1tbsp">wheat germ</b:ingredient>
      <b:ingredient qty="10ml">calendula oil</b:ingredient>
      <b:ingredient qty="2tsp">kombucha tea</b:ingredient>
      <b:ingredient qty="1ml">lavender essential oil</b:ingredient>
    </b:ingredients>
  </b:recipe>
</root>

```

Sada je prvi (<http://www.my-site.com/cooking>) prostor imena pretvoren u podrazumevani prostor imena, tako što je atribut kojim se on definiše kreiran bez prefiksa, isključivo korišćenjem naziva `xmlns`. Stoga će svi elementi bez prefiksa pripadati ovakvom podrazumevanom prostoru imena. Upravo zbog toga se prilikom definisanja kulinarskog recepta sada ne koriste prefiksi.

## Kreiranje, izmena, prikaz i stilizovanje XML dokumenata

Već je rečeno da se XML kod piše unutar fajlova sa ekstenzijom .xml. Sadržaj takvih dokumenata zapravo je običan tekst. Stoga se XML dokumenti mogu pregledati i kreirati korišćenjem bilo kog programa koji rukuje tekstom – od Notepada pa sve do različitih kompleksnijih tekst editora – Notepad++, Visual Studio Code, Atom, Sublime Text...

Sadržaj XML fajlova primarno se kreira sa ciljem da opiše neke podatke i da bude korišćen od strane specijalnih programa koji parsiraju njihov sadržaj. Drugim rečima, XML, za razliku od HTML-a, nije primarno kreiran kako bi se na osnovu njegovog koda formirao neki prikaz. Stoga, ukoliko se XML dokument otvori unutar web pregledača, on se prikazuje u svom izvornom obliku, sa vidljivim tagovima, atributima i ostalim elementima (slika 1.9).

This XML file does not appear to have any style information associated with it. The document tree is shown below.

---

```
▼<root>
  ▼<recipe>
    <title>Grilled Cheese Sandwich</title>
    ▼<ingredients>
      <ingredient qty="2">bread slice</ingredient>
      <ingredient>cheese slice</ingredient>
      <ingredient qty="2">margarine pat</ingredient>
    </ingredients>
  </recipe>
</root>
```

Slika 1.9. Izgled XML dokumenta unutar web pregledača

Slika 1.9. ilustruje izgled našeg XML dokumenta unutar Chrome pregledača. Kao logično se nameće pitanje zbog čega pregledači na ovaj način prikazuju XML dokumenta? Odgovor je vrlo jednostavan.

XML elemente kreira sam tvorac dokumenta, tako da web pregledači podrazumevano nemaju način na koji bi mogli da razumeju značenje XML elemenata, za razliku od HTML elemenata, čija namena je unapred utvrđena. Upravo zbog toga, web pregledači XML dokumenta prikazuju kao na slici 1.9.

Ukoliko se javi potreba za drugačijom reprezentacijom XML dokumenata unutar web pregledača, moguće je koristiti CSS jezik, baš kao i prilikom stilizovanja HTML-a. CSS fajl sa eksternom stilizacijom je unutar XML dokumenta potrebno uključiti jednom instrukcijom za procesiranje, koja se postavlja na početak dokumenta, nakon eventualnog XML prologa:

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



Sadržaj eksternog CSS fajla koji je na ovaj način povezan sa XML dokumentom izgleda ovako:

```
root {
  font-family: sans-serif;
}

recipe {
  display      : block;
  margin       : 2em 1em;
  border       : 4px solid #93AEC9;
  padding      : 0 1em 1em;
  background-color: white;
}

recipe:before {
  display      : block;
  width        : 8em;
  font-weight  : bold;
  font-size    : 200%;
  content      : "Cooking Recipes";
  margin       : -.75em 0px .25em -.25em;
  padding      : .1em .25em;
  background-color: #93AEC9;
  color        : white;
}

title {
  display      : block;
  font-size    : 22px;
  font-weight  : bold;
}

ingredients {
  display      : block;
  padding-left : 36px;
}

ingredient {
  display: list-item;
}
```

Definisanjem ovakve stilizacije unutar `recipe.css` fajla, XML dokument unutar web pregledača dobija izgled kao na slici 1.10.



Slika 1.10. Novi izgled XML dokumenta unutar web pregledača, za koji je zaslužna eksterna CSS stilizacija

## Rezime

- XML je skraćenica od **eXtensible Markup Language**.
- XML je način za serijalizaciju, odnosno strukturiranje podataka, tj. način na koji se jednostavno i brzo mogu zapamtiti i proslediti podaci.
- Istorija XML-a počinje još 60-ih godina prošlog veka, kada je u IBM-u konstruisan prvi višenamenski jezik za serijalizaciju podataka.
- Okosnicu XML-a čine tagovi, elementi, atributi i podaci.
- Tag je osnovni pojam XML jezika koji se koristi za markiranje podataka.
- Tagovi se koriste za kreiranje XML elemenata.
- Atributi omogućavaju da se nad XML elementima definišu neki dodatni podaci.
- XML deklaracija je specijalna oznaka koja informiše XML parser da je reč o XML dokumentu.
- XML dokumenti mogu sadržati komentare; komentarisan kod znači da parser neće uzeti u obzir taj deo koda.
- Prostori imena se koriste da jednoznačno odrede XML element, tj. grupišu srodne elemente.
- XML dokumenti mogu se pregledati i kreirati korišćenjem bilo kog programa koji rukuje tekstom.
- XML je moguće stilizovati korišćenjem CSS-a.

