ПРОГРАМИРАЊЕ ВЕБ АПЛИКАЦИЈА

Nastavana jedinica: 9
XML i JSON

POJAM XML-A

- E**X**stensible **M**arkup **L**anguage
- Jezik za serijalizaciju podataka
- Nije izvršni jezik
- Sastoji se od elemenata i atributa

POJAM XML-A

- **XML dokument** predstavlja podatke koji su tekstualno formatirani u skladu sa (strogim) pravilima.
- XML dokument tj. tekst sa podacima može da bude
  - smešten u datoteku na disku
  - kao poruka koja se šalje HTTP protokolom
  - kao niz znakova, tj. string u programskom jeziku
  - kao objekat u bazi podataka
  - na bilo koji drugi način koji omogućava korišćenje tekstualnih podataka
- Zbog svog formata, omogućava razmenu podataka i između nekompatibilnih sistema
- **Nezavisan je od korišćenih hardverskih i softverskih platformi.**

SVOJSTVA XML-A

- XML je danas postao *de-facto* **standard** za opis sadržaja i strukture (tekstualnih i multimedijalnih) dokumenata i razmenu dokumenata na Web-u.

Markup omogućava

- specijalno značenje podataka
- koristi se **tag** za predstavljanje markup-a

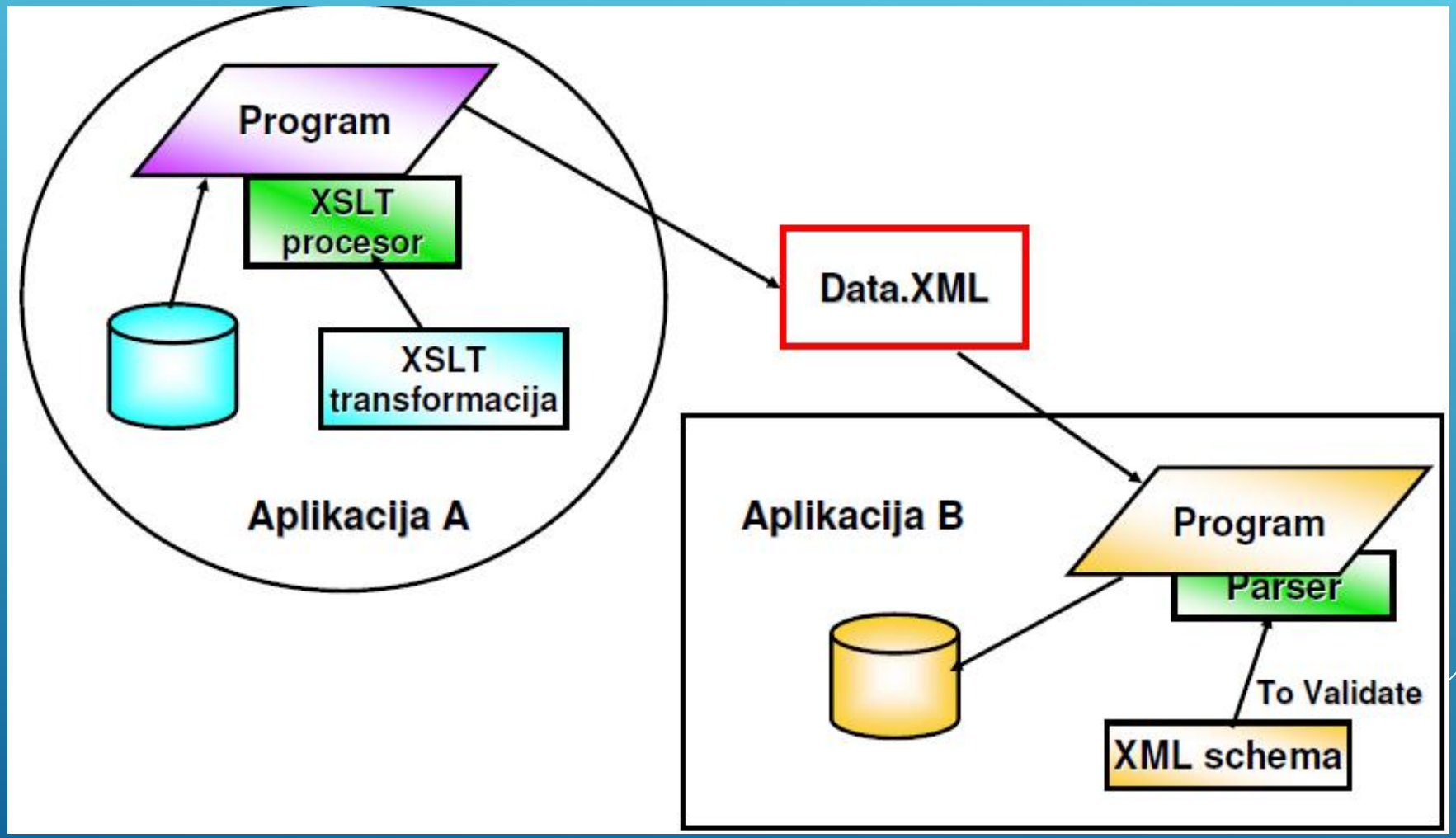
Extensible

- proširliv jezik, dozvoljava definisanje novih tagova
- predstavlja meta jezik koji omogućava definisanje drugih markup jezika

SVOJSTVA XML-A

- XML je **samoopisujući,** platformski **nezavisan** i u tekstualnom formatu.
- XML omogućava **razdvajanje struktuiranog sadržaja dokumenta od njegove prezentacije** (*Style Sheet*).
- XML je **projektovan za distribuirano okruženje.**
- XML je veoma **pogodan kao format za razmenu podataka između heterogenih aplikacija na Web-u.**
- XML kao format **je dovoljno formalan za mašinsko procesiranje i dovoljno razumljiv za korisnike**

PRIMER

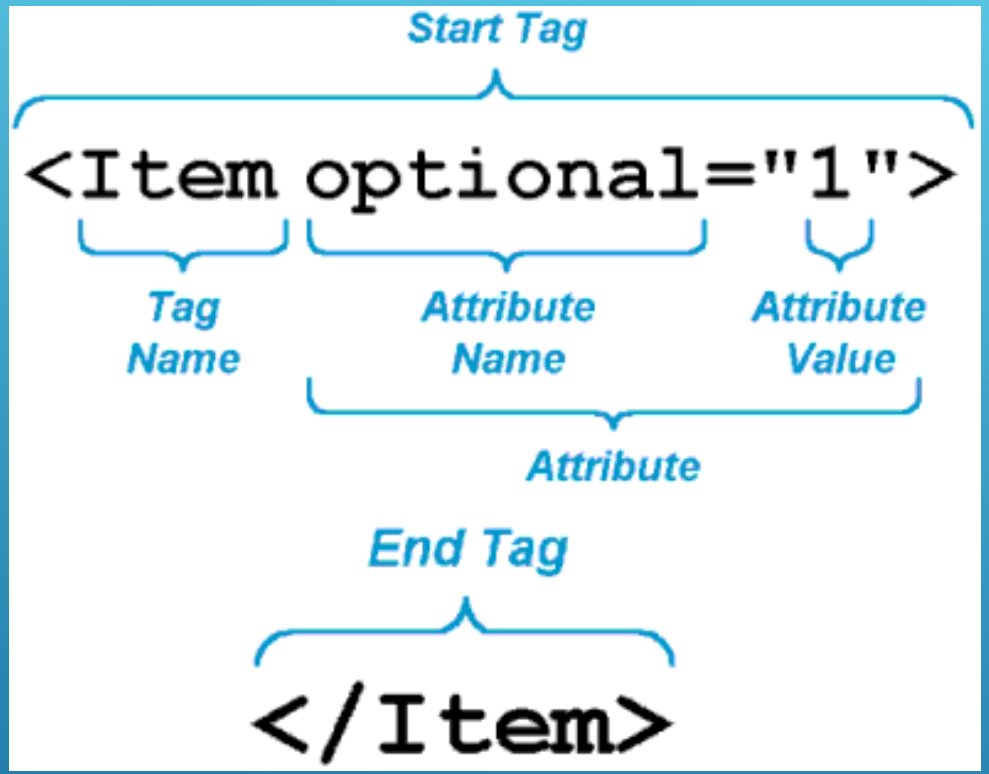

Program
XSLT procesor
XSLT transformacija
Aplikacija A
Data.XML
Aplikacija B
Program
Parser
To Validate
XML schema

SINTAKSA XML-A

- Sastoji se od elemenata i atributa
- Isti princip funkcionisanja kao i html
- Case sensitive
- Nisu dozvoljeni specijalni karakteri

&lt;	<
&gt;	>
&amp;	&
&apos;	‘
&quot;	“

XML ELEMENT ITEM



- XML dokument se sastoji iz teksta organizovanog uz pomoć tagova u elemente

XML ELEMENTI

- Elementi su osnovni blokovi XML-a

<pozdrav> Hello XML! </pozdrav>

- ***Složeni / kontejner*** element čini par tag-ova (početni/otvarajući i krajnji/zatvarajući tag) sa sadržajem

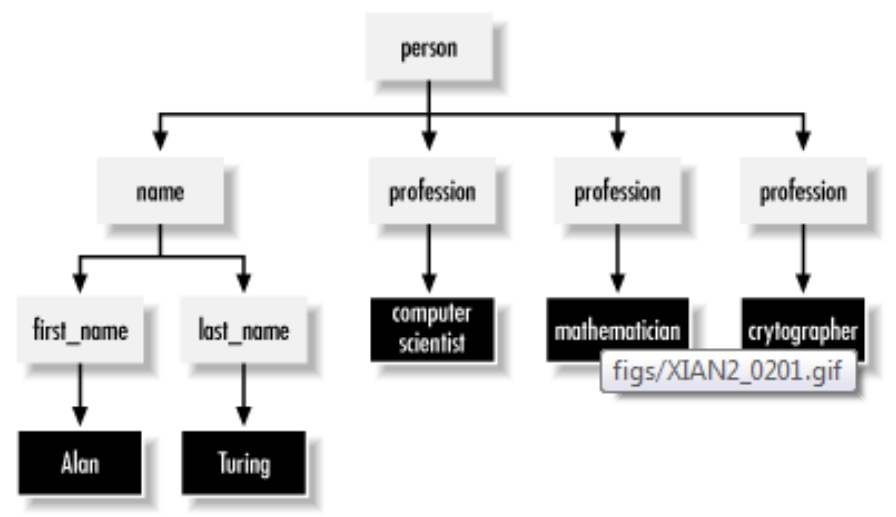
- ***Prost/prazan*** *element obično se za krajnji tag koristi* skraćenica **/>** (samozatvarajući tag)

<poruka/>

<pozdrav tekst = “Hello XML” />

ROOT – KORENI ELEMENT

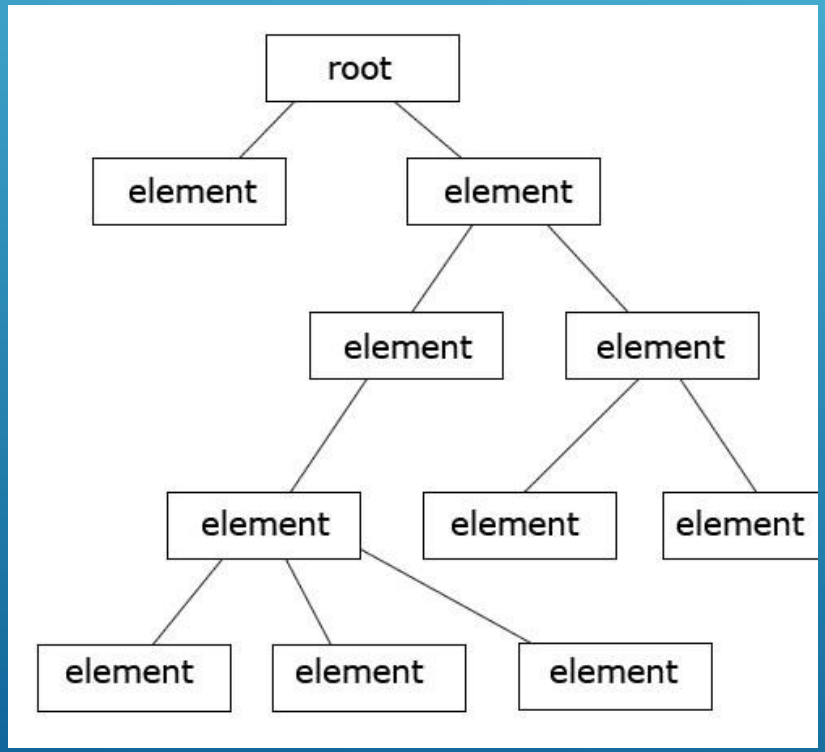
- XML dokument sadrži jedan element koji nema roditelja.
- To je prvi element i sadrži sve druge elemente.
- Taj element nazivamo korenskim (engl. *root*).




```
<person>
  <name>
    <first_name>Alan</first_name>
    <last_name>Turing</last_name>
  </name>
  <profession>computer scientist</profession>
  <profession>mathematician</profession>
  <profession>cryptographer</profession>
</person>
```

MEŠOVIT SADRŽAJ

- XML se može koristiti i za narativne dokumente slobodnijeg oblika, kao što su poslovni izveštaji, članci, eseji, priče...



ZNACI NAVODA

- Vrednosti atributa moraju uvek biti unutar znaka navoda. Moguće je koristiti **jednostruke ili dvostruke** znake navoda:

```
<ime="Krcun"> ili:

<ime='Krcun'>
```

- Dupli znaci navoda su češći. Nekada je neophodno koristiti jednostruke kao u sledećem primeru:

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

PRIMERI

- <Person name='Martin' age='33' />
- <age base='16' units='years' >20</age>
- <age base="10" units="years" >32</age>

ŠTA KORISTITI - ELEMENT ILI ATRIBUT?

Primer 1.

<ime>Krcun</ime>

<nesto ime="Krcun">

Primer 2a.

<partner tip="nabavljač">

<ime>Pera</ime>

<prezime>Perić</prezime>

</partner>

Primer 2b.

<partner> nabavljač </partner>

<ime>Pera</ime>

<prezime>Perić</prezime>

Podaci se mogu skladištiti u elementima ili u atributima.

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 (OGRANIČENJA) PRILIKOM KORIŠĆENJA ATRIBUTA

1. Atributi ne mogu sadržati višestruke vrednosti (elementi mogu)
2. Atributi nisu lako proširivi
3. Atributi ne opisuju strukturu
4. Atributima se teže manipuliše u programskom kodu
5. Vrednosti atributa se teško testiraju u odnosu na DTD (eng. Document Type Definition) – definicija tipa dokumenta

KOMENTARI

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

- Koristi se za dodatne informacije radi čitljivosti. Sadržaj u komentaru ne može imati **--**, sve ostale markup karaktere može.
- Aplikacije za parsiranje mogu a ne moraju da prosledjuju informacije o komentarima (misli se na sam sadržaj komentara).

- Primeri:

```
<!-- This is a comment about how to open ( <![CDATA[ ) and close ( ]]> ) CDATA sections -->

<!-- I really like having elements called <fred> in my markup languages -->

<!-- Comments can contain all sorts of character literals including &, <, >, ' and". -->
```

ODELJAK CDATA

- XML dokumenti imaju strogo definisana pravila. Ukoliko je potrebno da dokument sadrži običan tekst i specijalne zankove, dokument postaje teško čitljiv. U takvim slučajevima može se koristiti odeljak CDATA koji obezbedjuje da se sadržaj tretira kao sirovi tekst. Na primer.

```
<p>ovde ide neki tekst </p>
<pre>
<![CDATA[
    <svg xmlns="http://www.w3.org/2000/svg"
    width="12cm" height="10cm">
    <ellipse rx="110" ry="130" />
    <rect x="4cm" y="1cm" width="3cm" height="6cm" />
]]>
</pre>
```

ODELJAK CDATA

Није безбедно | websrv3.viser.edu.rs/xml/rss-vesti.php

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


```
▼<rss version="2.0">
  ▼<channel>
    ▼<title>
       Висока школа електротехнике и рачунарства струковних студија - Вести
     </title>
     <link>http://www.viser.edu.rs/</link>
     <lastBuildDate>2019-12-09T23:57:16+01:00</lastBuildDate>
     <language>sr</language>

    ▼<item>
       <title>[Стручна пракса] - Стручна пракса - Смер НРТ</title>
       <link>http://www.viser.edu.rs/vesti.php?id=8442</link>
      ▼<description>
        ▼<![CDATA[
           <p>Обавештавају се сви студенти који пријављују испит из предмета Стручна пракса на смеру НРТ код проф. Јелене Митић да су обавезни да се пријаве на курс "
           <strong>Стручна пракса НРТ</strong>" на Moodle-у систему за даљинско учење, како би благовремено примили обавештења везана за испит.</p><br><p>Након положеног
           испита студенти су обавезни да се одјаве са курса.</p><br><p>Шифра за приступ курсу је "strucnapraksa".</p><br><p>Процедура за полагање испита Стручна пракса за
           смер НРТ следећа:<br />1. Припрема Word документа са детаљима Стручне праксе коју је студент обављао.<br />2. Слање документа предметном сараднику Бошку Богојевићу
           на адресу електронске поште <a href="mailto:bbosko@viser.edu.rs">bbosko@viser.edu.rs</a> у термину заказаном за одговарајући испитни рок.<br />3. Опционо, у
           зависности од броја кандидата - Избор времена полагања (обавља се на Moodle-у).<br />4. Припрема кратке PowerPoint презентације (не дуже од 6 минута).<br
           />5. Припремити диск са нарезаним документом и презентацијом.<br />6. На испит понети диск и Дневник стручне праксеу штампаној форми у једном примерку.
           </p><br><p>Студенти који нису запослени морају имати упут за стручну праксу. Студенти који су запослени на испит могу донети уверење о запослењу (није им потребан
           упут за стручну праксу).</p><br><p>Предметни сарадник<br />Бошко Богојевић</p><br>
         ]]>
       </description>
       <pubDate>2013-07-03T11:35:22+02:00</pubDate>
     </item>
    ▼<item>
      ▼<title>
         [Дизајн звучне слике] - Процедура за реализацију Дипломског рада
       </title>
       <link>http://www.viser.edu.rs/vesti.php?id=3707</link>
      ▼<description>
        ▼<![CDATA[
           <p style="text-align: justify;">Дипломски рад се састоји из ДВА дела:</p><br><p style="text-align: justify;">1. Практични део</p><br><p style="text-align:
           justify;">2. Теоријски део</p><br><p style="text-align: justify;"> </p><br><p style="text-align: justify;">1. Практични део Дипломског испита може бити дизајн
```

STRUKTURA XML-A

- Svaki XML dokument mora da sadrži XML **deklaraciju**, tj. ***instrukciju obrade kojom se*** dokument identifikuje kao XML dokument. Ovo je prva konstrukcija u dokumentu.
- Osnovni oblik XML deklaracije: **<?xml version =“1.0”?>**
- Opcioni oblik XML deklaracije:

<?xml version =“1.0” encoding= “UTF-8”?>

<?xml version =“1.0” encoding= “UTF-16”?>

- **Opšti oblik XML deklaracije:**

<?xml **version**='1.0’ **encoding**='character encoding' **standalone**='yes|no'?>

PRIMERI

- `<?xml version='1.0' ?>`
- `<?xml version='1.0' encoding='US-ASCII' ?>`
- `<?xml version='1.0' encoding='US-ASCII' standalone='yes' ?>`
- `<?xml version='1.0' encoding='UTF-8' ?>`
- `<?xml version='1.0' encoding='UTF-16' ?>`
- `<?xml version='1.0' encoding='ISO-10646-UCS-2' ?>`
- `<?xml version='1.0' encoding='ISO-8859-1' ?>`
- `<?xml version='1.0' encoding='Shift-JIS' ?>`

ATRIBUT VERSION

- Atribut *version* treba da ima vrednost 1.0. U veoma retkim slučajevima može da dobije vrednost 1.1. Pošto zadavanje verzije 1.1 ograničava upotrebu na manji broj analizatora, ne bi trebalo postavljati ovu verziju bez istinskog razloga.

- Ako ne govorite burmanski, mongolski, kambodžanski, amharski ili divehi, ako ne koristite netekstualne kontrolne znakove (vertikalni tabulator, nova stranica, zvonce) nemate razloga da koristite verziju 1.1

ATRIBUT ENCONDING

- XML je dokument od čistog teksta.
- Svi XML dokumenti su podrazumevano kodirani sa UTF-8 kodovima promenljive dužine u skupu Unicode.

ATRIBUT STANDALONE

- Ako ovaj atribut ima vrednost "*no*" onda aplikacija može učitati spoljni DTD preko koga se utvrđuju vrednosti i značenje delova dokumenta.
- Dokument koji nema DTD može imati ovaj atribut postavljen na vrednost "*yes*".
- Ako je izostavljen, pretpostavlja se da ima vrednost "*no*"

PRAZNINE U XMLU

- 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

(to sa HTML-om nije slučaj)

DA BI DOCUMENT BIO DOBRO-FORMIRAN XML DOCUMENT TREBA DA:

1. Postoji XML **deklaracija**.
2. Dokument sadrži jedan i **samo jedan koreni element** u kome su ugnježdeni svi ostali elementi i njihovi sadržaji.
3. Svi elementi i atributi u dokumentu moraju da budu **sintaksno ispravni**.
4. Elementi moraju imati **završni tag** (<…> … </…>). Jedini izuzetak predstavlja *empty* tag koji nema ni sadržaj ni telo, a označava se <…/>.
5. Elementi moraju biti **ugnježdeni**.
6. Imena tagova u XML-u su case-sensitive (zavise od velikih i malih slova). **Pri dodeljivanju imena se moraju poštovati određena pravila**.
7. Sve **vrednosti atributa** moraju biti u okviru **navodnika**.

DOCUMENT TYPE DEFINITION

- Da bi se uvela ograničenja tj opis strukture podataka i time dobile nove mogućnosti. Opis strukture je javan i potpuno razumljiv.
- Automatizacija provere ispravnosti dokumenta.
- Višestruka upotreba definisanih delova dokumenta.
- DTD je dodatni dokument koji može biti pridružen nekom XML dokumentu i pruža gore navedene osobine.

DEFINICIJA DTD?

- Skraćenica od naziva: ***D****ocument* ***T****ype* ***D****efinition*

- DTD definiše sledeće:
  1. elemente koji se pojavljuju u dokumentu
  2. broj elemenata u dokumentu
  3. način ugnježdavanja
  4. skup dozvoljenih, zahtevanih i podrazumevanih atributa

ŠTA DTD MOŽE DA DEKLARIŠE?

Tip elementa

- Opisuje tip elementa, kao i tipove podataka koje on sadrži.

Tip atributa

- Opisuje tip atributa, kao i tipove podataka koje on može da sadrži.

Listu atributa

- Elementi imaju atribute navedene u listi. Liste omogućavaju grupisanje svih srodnih atributa elementa.

ŠTA DTD **NE MOŽE** DA DEKLARIŠE?

- Koji je korenski element.
- Koliko primeraka elemenata pojedinih vrsta ima u dokumentu.
- Kako izgledaju znakovni podaci unutar elemenata.
- Semantiku, tj. značenje elemenata; na primer, da li određeni element sadrži datum ili ime neke osobe.

PRIMER DTD FAJLA ZA POZNATI PRIMER

```
<person>
  <name>
    <first_name>Alan</first_name>
    <last_name>Turing</last_name>
  </name>
  <profession>computer scientist</profession>
  <profession>mathematician</profession>
  <profession>cryptographer</profession>
</person>
```

```
<!ELEMENT person (name, profession*)>
<!ELEMENT name (first_name, last_name)>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT last_name  (#PCDATA)>
<!ELEMENT profession (#PCDATA)>
```

KAKO SE DTD PRIDRUŽUJE XML FAJLU?

- Kao poseban fajl u odnosu na XML
  - Može biti naknadno više korišćen
- Kao deo XML dokumenta
- Kombinacijom prethodna dva slučaja, tj. deo DTD-a je u posebnom dokumentu, a deo u XML dokumentu

SPOLJAŠNJI DTD

XML i DTD su zasebni dokumenti, prosleđuje se URI na kome je DTD fajl

hello.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE message SYSTEM "http://www.xxxxxxxx.rs/dtd/message.dtd">
<message>
     <greeting>Hello, World!</greeting>
     <farewell>Goodbye, World!</farewell>
</message>
```

http://www.xxxxxxxxx.rs/dtd/message.dtd

```
<!ELEMENT message (greeting,farewell)>
<!ELEMENT greeting (#PCDATA)>
<!ELEMENT farewell (#PCDATA)>
```

UGRAĐENI DTD

XML i DTD su u istom dokumentu

hello.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE message [
   <!ELEMENT message (greeting,farewell)>
   <!ELEMENT greeting (#PCDATA)>
   <!ELEMENT farewell (#PCDATA)>
]>
<message>
   <greeting>Hello, World!</greeting>
   <farewell>Goodbye, World!</farewell>
</message>
```

BLOKOVI KOJI SE DEFINIŠU PREKO DTDA

- Elementi
- Atributi
- Entiteti
  - Neki karakteri imaju specijalno značenje. (<,>,&..). Entiteti se ekspanduju kada se neki dokument parsira.
- PCDATA (engl. *Parsed Character Data)*
  - Tekst koji će biti procesiran od parsera. Tekst će biti analiziran kao markap celine.
- CDATA
  - *Character Data.*
  - Označava tekst koji neće biti parsiran. Tagovi u okviru ovog teksta neće biti tretirani kao markap.

DEKLARISANJE ELEMENATA

- **<!ELEMENT ime (sadrzaj)>**

  ime – ime elementa

  sadrzaj – podaci koje element može da sadrži

Primer:

DTD

```
<!ELEMENT greeting (#PCDATA)>
```

Validni XML kod

```
<greeting>Hello World!</greeting>
<greeting>
    <![CDATA[G'day!]]>
</greeting>
```

TIPOVI SADRŽAJA

Definisanje strukture

Definiše niz elemenata deklarisanih na sledeći način:

<!ELEMENT ime (struktura)>

Element je definisan imenom (**ime**) i strukturom elemenata koji su njegovi podelementi tj. članovi.

Tipovi članstva u strukturi elementa:

Sekvenca	<!ELEMENT ime (a,b)>
Izbor	<!ELEMENT ime **(a|b)**>
Jedan	<!ELEMENT ime **(a)**>
Jedan ili više	<!ELEMENT ime **(a) +**>
Nula ili više	<!ELEMENT ime **(a) ***>
Nula ili jedan	<!ELEMENT ime **(a) ?**>

PRIMER

DTD:

```
<!ELEMENT order (order-item+,delivery-address,order-date?)>
```

XML kod 2 deo:

```
<order>
   <order-item>item3</order-item>
   <order-item>item4</order-item>
   <delivery-address>123 State Street</delivery-address>
</order>
```

ili

```
<order>
   <order-item>item5</order-item>
   <order-item>item6</order-item>
   <delivery-address>123 State Street</delivery-address>
   <order-date>July 5, 2001</order-date>
</order>
```

LISTE ATRIBUTA

Koristi se za deklarisanje atributa elementa.

Dve moguće sintakse:

```
<!ATTLIST elemenat atribut tipAtr podrDek>
<!ATTLIST elemenat
   atribut tipAtr podrDek
                  ...
   atribut tipAtr podrDek >
```

podrDek je podrazumevana deklaracija atributa.

PRIMER

- `<!ATTLIST img source CDATA #REQUIRED>`

1. Element **img** ima atribut **source**.
2. Vrednost atributa su znakovni podaci (CDATA).
3. Svi primerci elementa *img* moraju imati neku vrednost za atribut *source*.

PRIMER

- Jedna deklaracija ATTLIST može deklarisati više atributa istog elementa. Na primer:
- **<!ATTLIST** img **source CDATA #REQUIRED**
  width CDATA #REQUIRED
  height CDATA #REQUIRED
  alt CDATA #IMPLIED >

- Atribut alt je neobavezan i može se izostaviti u nekim elementima.
- Ova deklaracija ima isti efekat i značenje kao 4 zasebne deklaracije ATTLIST, po jedna za svaki atribut.

NEDOSTACI

- Glavni nedostaci DTD-a su:
  - odsustvo tipizacije podataka (#PCDATA može biti bilo koji string)
  - sintaksa DTD nije usklađena sa sintaksom XML-a
  - postoje ograničenja koja se ne mogu lako izraziti DTD-om (npr. element x se može pojaviti od 4 do 17 puta)
- Mnoga ograničenja DTD-a uspešno prevazilazi XML Schema (XML šema).

XSD - XML SCHEMA DEFINITION

- **XML Schema** predstavlja (XML) dokument koji sadrži opis strukture i pravila koja čine validan XML dokument.
- XML dokument ako zadovoljava sva ograničenja zadata šemom, je validan za tu šemu.
- XML Schema dokumenti imaju ekstenziju **.xsd** - pišu se u XSD (*XML Schema Definition*) jeziku.

XML SCHEMA PODRŽAVA TIPOVE PODATAKA

- Jedna od najvažnijih osobina je podrška tipovima podataka.

- XSD omogućava lakšu primenu za:
  - opis dozvoljenog sadržaja tj tipova
  - validnost podataka
  - rad sa podacima iz baza
  - definisanje ograničenja za podatke
  - definisanje složenih uzoraka podataka
  - konverzije između tipova podataka

DTD I XML SCHEMA

DTD deklaracija elementa *količina*

```
<!ELEMENT količina (#PCDATA) >
```

Deklaracija elementa *količina* u XML Schema

```
<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
    <xsd:element name='količina' type="xsd:nonNegativeInteger'/>
</xsd:schema>
```

Na osnovu DTD deklaracije validan je XML kod:

```
<količina>3</količina>
<količina>-12</količina>
<količina>lots</količina>
```

Na osnovu XML Scheme validan je samo prvi primer.

XSD - XML SCHEMA DEFINITION - ŠEMA

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="drzava">
  <xs:complexType>
   <xs:sequence>
      <xs:element name="naziv" type="xs:string" />
      <xs:element name="glavniGrad" type="xs:string" />
      <xs:element name="opis" type="xs:string" />
   </xs:sequence>
  <xs:attribute name="oznakaDrzave" type="xs:string"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

XSD - XML SCHEMA DEFINITION - XML

```
<?xml version="1.0" encoding="UTF-8"?>
<drzava xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation= "drzavaSema.xsd"
   oznakaDrzave="sr">
   <naziv>Srbija</naziv>
   <glavniGrad>Beograd</glavniGrad>
   <opis>Opis Srbije.....</opis>
</drzava>
```

DEKLARACIJE ELEMENATA

Najčešći oblik elementa u šemi:

<xs:element name="ime" type="xs:tip">

Sadržaj elementa može biti:

- prost
- složen.

Prost sadržaj se sastoji samo od teksta bez ugnježdenih elemenata. U tabeli su navedeni najčešće korišćeni prosti tipovi definisani u W3C specifikaciji:

anyURI	URI identifikator resursa	duration	Vremenski period (relativno vreme)
Boolean	true ili false	integer	Ceo broj
dateTime	datum i vreme	string	Unicode znakovni niz

DEKLARACIJE ATRIBUTA

- **Atributi moraju biti deklarisani kao prosti tipovi.**
- Elementi koji sadrže atribute su složenog tipa.
- Atributi se deklarišu pomoću elementa:

<xs:attribute name="ime" type="xs:tip">

- Primer:
  - **<xs:attribute name="id" type="xs:integer"/> definisanje atributa**
  - <proizvod id="135">stolica</proizvod> - upotreba atributa

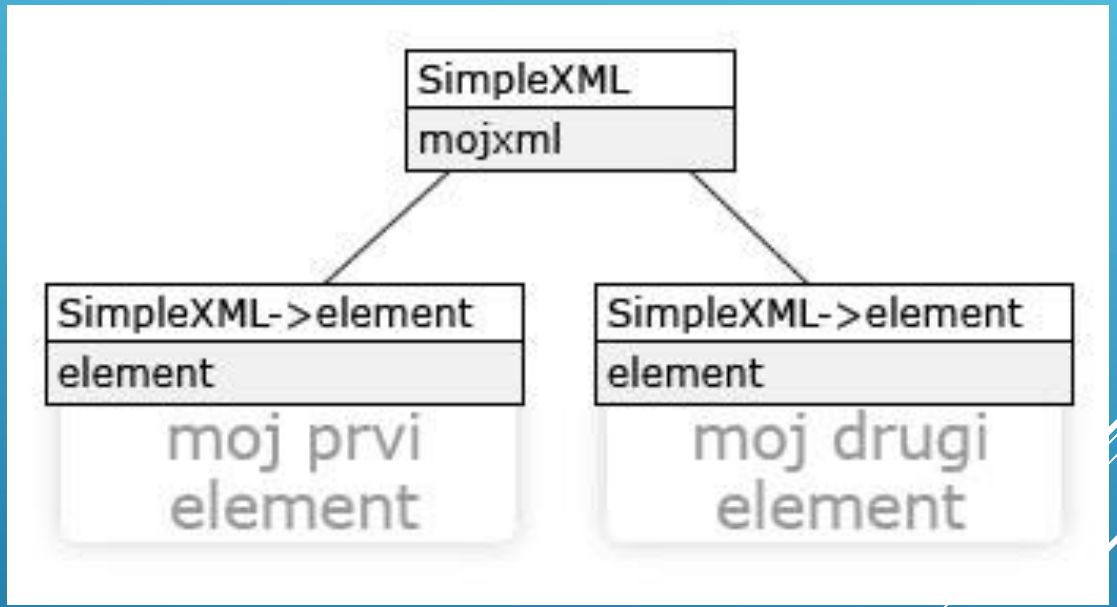
- Atributi su podrazumevano opcioni. Ako je obavezan: **use="required"**
- Primer:
  - <xs:attribute name="id" type="xs:integer" use="optional"/>
  - <xs:attribute name="id" type="xs:integer" use="required"/>

SIMPLE XML

- Klasa za rukovanje XML-om.
- Omogućava jednostavan način dobavljanja informacija o elementu.
- Pretvara XML dokument u strukturu podataka koja je laka za obradu u PHP programskom jeziku.

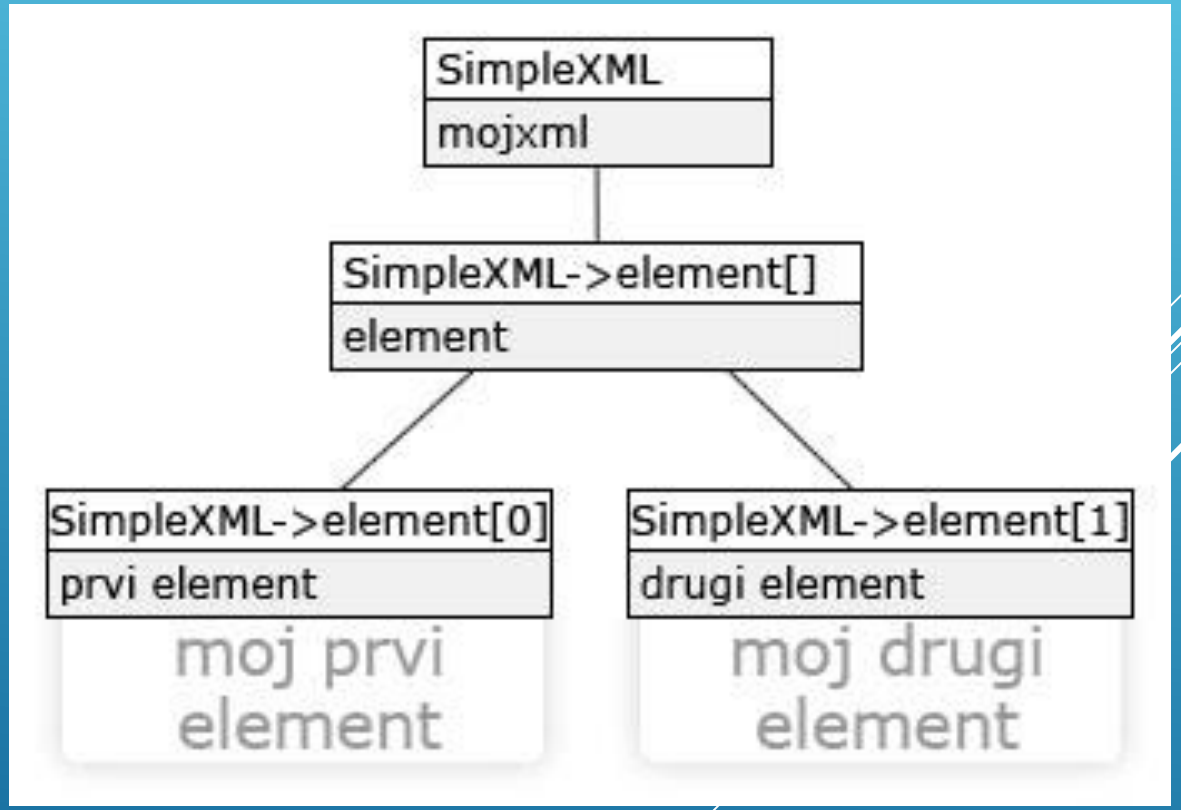
STRUKTURA SIMPLEXMLELEMENT-A

```
<mojxml>
    <element>
        moj prvi element
    </element>
    <element>
        moj drugi element
    </element>
</mojxml>
```



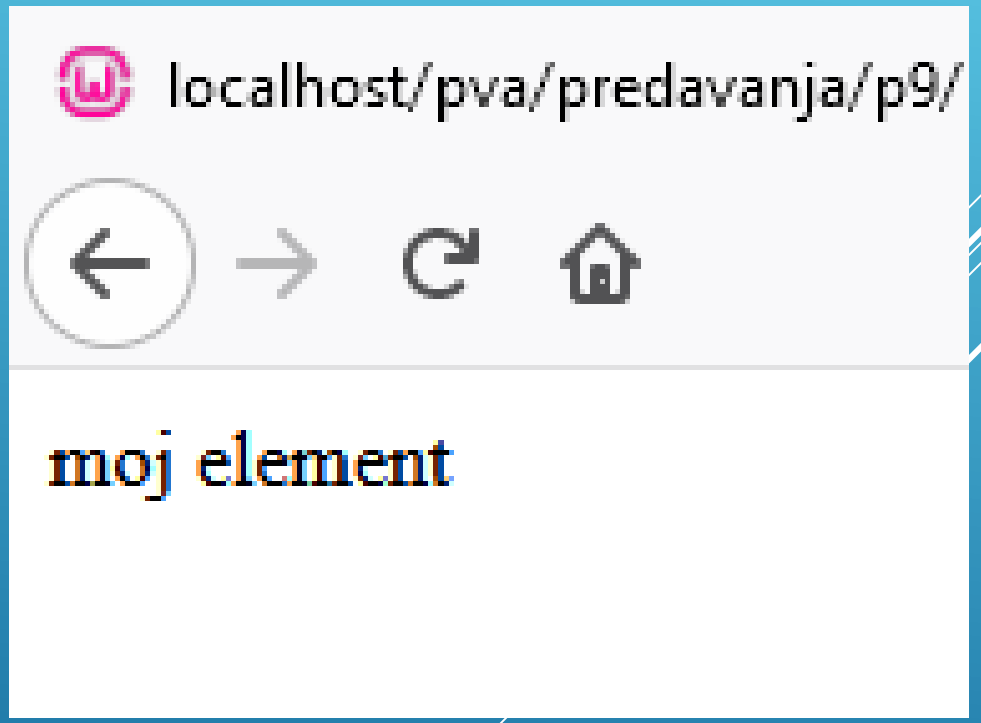
STRUKTURA SIMPLEXMLELEMENT-A

```
<mojxml>
    <element>
        moj prvi element
    </element>
    <element>
        moj drugi element
    </element>
</mojxml>
```



KREIRANJE SIMPLEXML OBJEKTA

```php
<?php
$xmlTekst = <<<XML
<mojxml>
    <element>
        moj element
    </element>
    <element>
        moj element 123
    </element>
</mojxml>
XML;

$xml = new SimpleXMLElement($xmlTekst);
echo $xml->element[0];
```

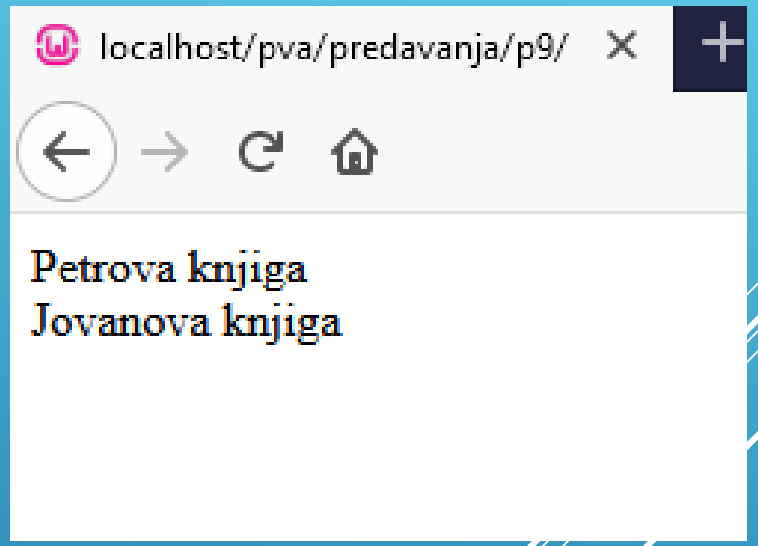


PREUZIMANJE DOKUMENTA IZ FAJLA

```php
<?php
$xmlTekst = file_get_contents("xmlFajl.xml");
$xml = new SimpleXMLElement($xmlTekst);

$xml = simplexml_load_file("xmlFajl.xml");

//simplexml_load_string(); (radi isto kao i fajl, samo što prihvata string)

$xml = simplexml_load_string($xmlTekst);
```

NAVIGACIJA KROZ SIMPLEXML DOKUMENT

```php
<?php
$xmlTekst = <<<XML
<izdanja>
    <knjiga>
        <naslov>Petrova knjiga</naslov>
        <autor>Petar Petrovic</autor>
    </knjiga>
    <knjiga>
        <naslov>Jovanova knjiga</naslov>
        <autor>Jovan Jovanovic</autor>
    </knjiga>
</izdanja>
XML;
$xml = new SimpleXMLElement($xmlTekst);

foreach($xml as $knjiga)
echo $knjiga->naslov."<br>";
?>
```

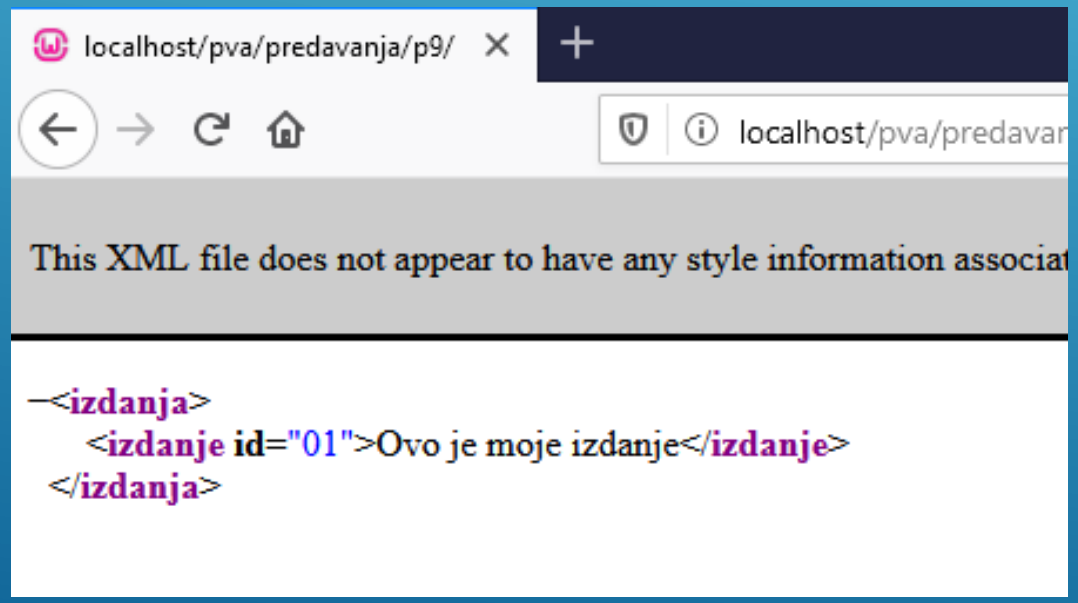


MODIFIKACIJA XML DOKUMENTA

- Tri stvari su nam dovoljne da kreiramo novi XML dokument
  1. *new SimpleXMLElement() – kreiranje instance*
  2. *addChild() – dodavanje podelementa*
  3. *addAttribute() – dodavanje atributa*

MODIFIKACIJA XML DOKUMENTA

```php
<?php
$xml = new SimpleXMLElement("<?xml version='1.0' encoding='utf-8'?><izdanja></izdanja>");
$izdanje = $xml->addChild("izdanje", "Ovo je moje izdanje");
$izdanje->addAttribute("id", "01");
header("Content-type: text/xml");
echo $xml->asXML();//$xml->asXML("mojFajl.xml");
?>
```



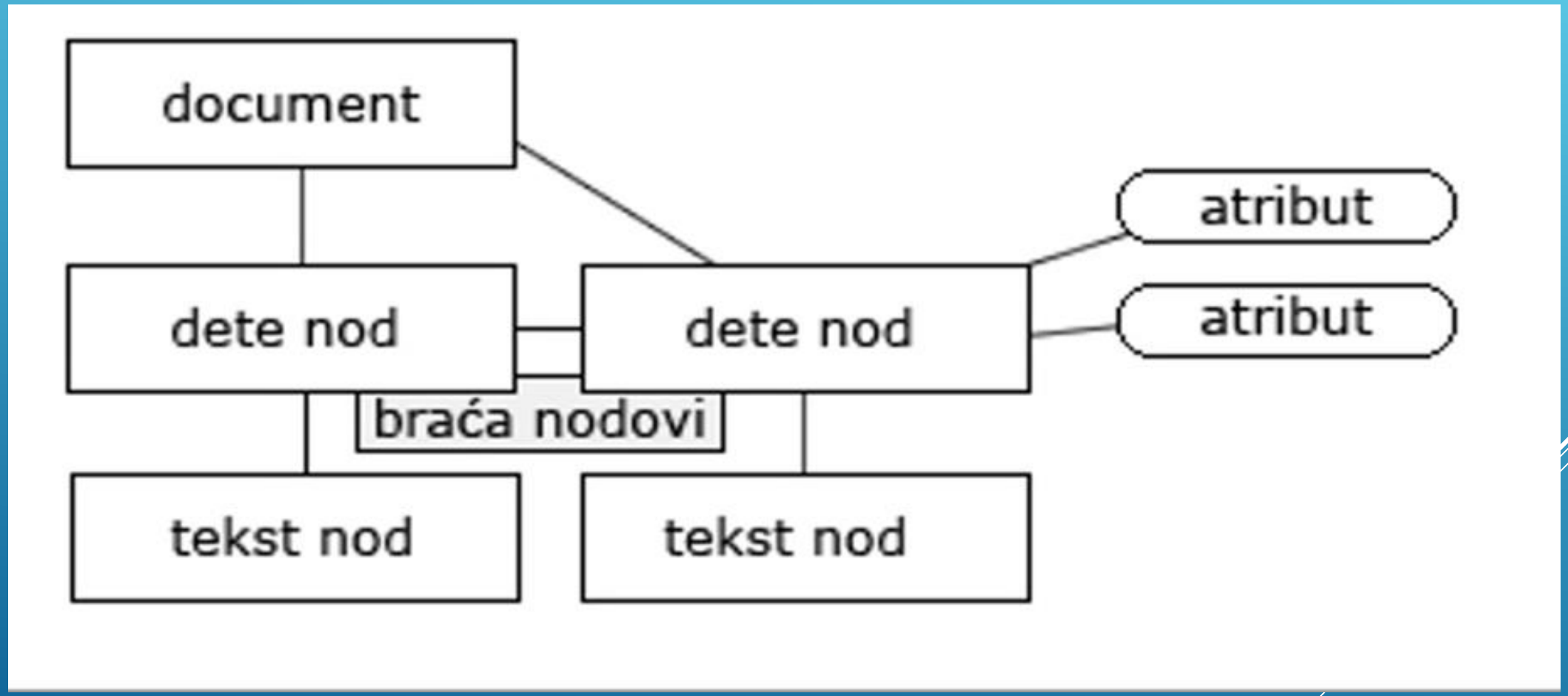
DOM – DOCUMENT OBJECT MODEL

- Kao i u JS-u i u PHP-u postoji DOM
- Konvencija za rukovanje XML dokumentima
- Veoma često se koristi za rukovanje objektima u JS

STRUKTURA DOM

- Ima strukturu drveta (tree)
- Koreni element je **document**
- Podelementi su deca nodovi (child nodes)
- Braća nodovi (siblings)
- Tekst nodovi
- Atributi

STRUKTURA DOM



KREIRANJE DOM OBJEKTA

- DOM je klasa i kreira se korišćenjem rezervisane reči **new**

```php
<?php
    $obj=new DOMDocument();
?>
```

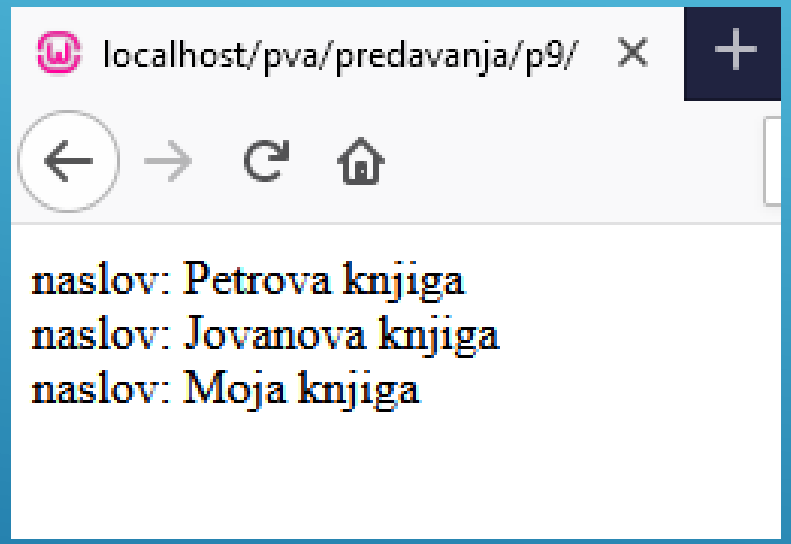
- Može se kreirati iz stringa

```php
<?php
    $obj=new DOMDocument();
    $obj->loadXML($xmlString);
?>
```

- Može se kreirati iz datoteke

```php
<?php
    $obj=new DOMDocument();
    $obj->load("datoteka.xml");
?>
```

NAVIGACIJA KROZ DOM

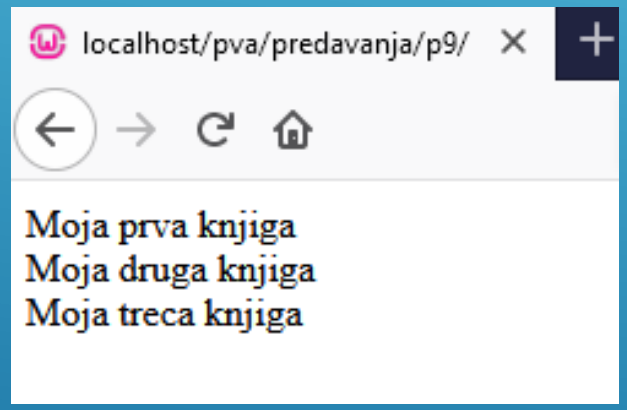



```php
<?php
//Prikaz svih izabranih elemenata
$xmlTekst=<<<XML
<?xml version="1.0"?>
<izdanja>
    <knjiga>
        <naslov>Petrova knjiga</naslov>
        <autor>Petar Petrovic</autor>
    </knjiga>
    <knjiga>
        <naslov>Jovanova knjiga</naslov>
        <autor>Jovan Jovanovic</autor>
    </knjiga>
    <knjiga>
        <naslov>Moja knjiga</naslov>
        <autor>ja</autor>
    </knjiga>
</izdanja>
XML;
$xmlDOM=new DOMDocument();
$xmlDOM->loadXML($xmlTekst);
$element =$xmlDOM->getElementsByTagName("naslov");
foreach($element as $nod)
    echo $nod->nodeName.": ".$nod->nodeValue."<br>";
?>
```

NAVIGACIJA KROZ DOM

- Obzirom da su nam u XML-u najvažniji elementi (tagovi), najčešća navigacije je po tagovima.
- Za to se koristi funkcija koja ima isti naziv i u JS-u **getElementsByTagName()** koja vraća sve elemente kao niz.
- Vrednosti elementa pristupamo koristeći metodu **nodeValue**, a imenu taga pristupamo koristeći **nodeName().**

DOM ATRIBUTI




```php
<?php
$xmlTekst=<<<XML
<?xml version="1.0"?>
<izdanja>
    <knjiga id="Moja prva knjiga">
        <naslov>Petrova knjiga</naslov>
        <autor>Petar Petrovic</autor>
    </knjiga>
    <knjiga id="Moja druga knjiga">
        <naslov>Jovanova knjiga</naslov>
        <autor>Jovan Jovanovic</autor>
    </knjiga>
    <knjiga id="Moja treca knjiga">
        <naslov>Moja knjiga</naslov>
        <autor>ja</autor>
    </knjiga>
</izdanja>
XML;
$xmlDOM=new DOMDocument();
$xmlDOM->loadXML($xmlTekst);
$nodovi=$xmlDOM->getElementsByTagName("knjiga");
foreach($nodovi as $nod)
    echo $nod->getAttribute("id")."<br>";
?>
```

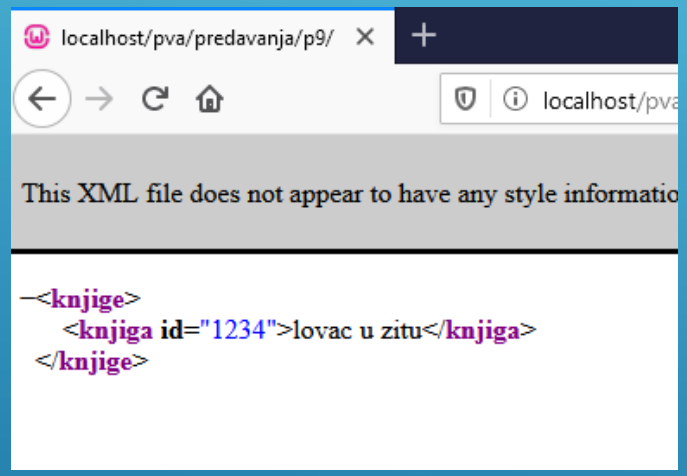
DOM ATRIBUTI

- Svaki DOM element može imati svoje atribute
- Atributima pristupamo tako što se referenciramo na element, pa pristupimo njegovom atributu koristeći metodu **getAttribute()**

MODIFIKOVANJE DOM DOKUMENTA

- Postoje 3 metode koje su nam potrebne za kreiranje i modifikovanje DOM dokumenta
  1. createElement() – kreiranje elementa
  2. appendChild() – dodavanje elementa
  3. createAttribute() – kreiranje atributa

MODIFIKOVANJE DOM DOKUMENTA




```php
<?php
$xmlDoc = new DOMDocument("1.0","utf-8");
$knjige = $xmlDoc->createElement("knjige");
$knjiga=$xmlDoc->createElement("knjiga","lovac u zitu");
$knjige->appendChild($knjiga);
$xmlDoc->appendChild($knjige);
$idKnjige=$xmlDoc->createAttribute("id");
$idKnjige->value="1234";
$knjiga->appendChild($idKnjige);
header("Content-type:text/xml");
echo $xmlDoc->saveXML();
?>
```

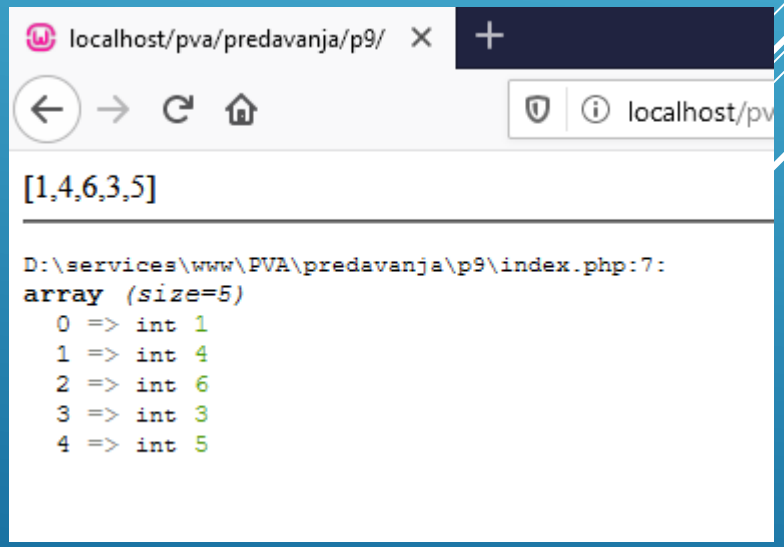
JAVA SCRIPT OBJECT NOTATION - JSON

- Obzirom da XML zauzima mnogo mesta za opis strukture danas je mnogo korišćeniji drugi način slanja podataka, a to je JSON
- JSON je, takođe, samoopisujući i svi programski jezici imaju mogućnost obrade JSON podatka
- JSON omogućava serijalizaciju objekata, klasa, nizova.....
- Najveća korist je mogućnost prosleđivanja objekta ili niza iz PHP u JS.

PHP JSON

- U programskom jeziku PHP postoje dve instrukcije koje rade sa JSON
  - json_encode() – kreiranje JSON podatka od klase, objekta ili niza
  - json_decode() – kreiranje objekta od JSON stringa

```php
<?php
$x=array(1, 4, 6, 3, 5);
$json=json_encode($x);
echo $json;
echo "<hr>";
$y=json_decode($json);
var_dump($y);
?>
```



PHP JSON

```php
<?php
$x=array("ime"=>"Пера", "prezime"=>"Перић", "visina"=>185, "tezina"=>100);
$json=json_encode($x, 256);
echo $json;
echo "<hr>";
$y=json_decode($json);
var_dump($y);
?>
```

JS JSON

- U programskom jeziku JS postoje dve instrukcije koje rade sa JSON
  - JSON.parse(str) – kreiranje objekta od JSON stringa
  - JSON.stringify(obj) – kreiranje JSON podatka od klase, objekta ili niza

JS JSON

```
<script>
    var obj = { name: "John", age: 30, city: "New York" };
    var myJSON = JSON.stringify(obj);
    console.log(myJSON);
    var x=JSON.parse(myJSON);
    console.log(x.name);
</script>
```

WEB SERVISI

- Skalabilne, multiplatformske biblioteke
- Slaba povezanost i udaljeno izvršavanje
- Podaci se lako transportuju

TIPOVI WEB SERVISA

- SOAP – Simple Object Access Protocol – koristi samo XML za request i response
- RPC – Remote Procedure Call – koristi samo XML za request i response
- REST - Representational State Transfer – može da koristi i JSON i XML za request i response

RESTFULL SERVISI - REST

- Representational State Transfer
- Koristi HTTP protokol
- Zahtev i odgovor ne koristi posebne xml wrapper-e
- PHP nema ugrađenu funkcionalnost za rest

REST I HTTP

- REST koncept koristi url-ove za reprezentaciju objekata i HTTP metode za rukovanje njima
  - GET za preuzimanje podataka
  - POST za kreiranje novih podataka
  - PUT za update podataka
  - DELETE za brisanje podataka.

REST I HTTP

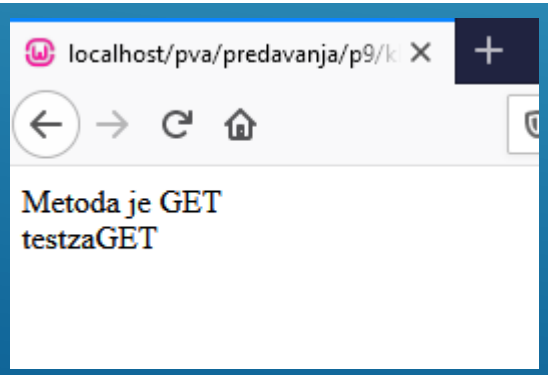
- Postoji klijentska strana koja šalje zahtev i serverska strana koja obrađuje zahtev
- Na serverskoj strani koristi se globalna promenljiva **$_SERVER** da bi se videlo kojom metodom stiže informacija

```php
<?php
//servis.php
$metoda=strtolower($_SERVER['REQUEST_METHOD']);
switch($metoda)
{
    case 'get';
        echo "Metoda je GET";
        break;
    case 'post';
        echo "Metoda je POST";
        break;
    case 'put';
        echo "Metoda je PUT";
        break;
    case 'delete';
        echo "Metoda je DELETE";
        break;
}
?>
```

REST I HTTP

- Na klijentskoj strani zahtev se mora slati na poseban način
- POST i GET metode nisu problematične jer HTTP ima funkcionalnost za njih

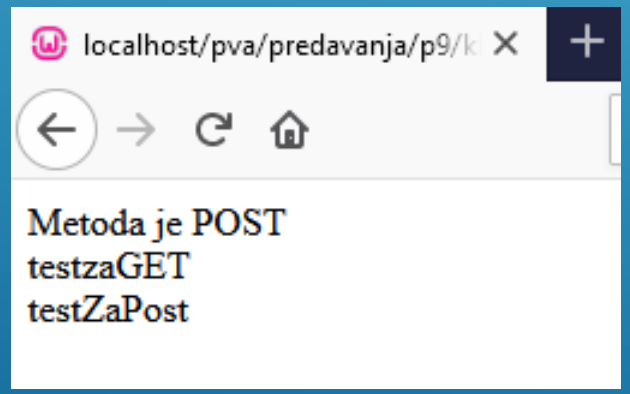
```php
<?php
$param=stream_context_create(array(
    "http"=>array(
            "method"=>"GET",

)));
$odgovor=file_get_contents("http://putanja/do/servisa/servis.php?b=testzaGET", false,
$param);
echo $odgovor;
?>
```




```php
<?php
//servis.php
$metoda=strtolower($_SERVER['REQUEST_METHOD']);
switch($metoda)
{
    case 'get';
        echo "Metoda je GET<br>";
        echo $_GET['b']."<br>";
        break;
    case 'post';
        echo "Metoda je POST";
        break;
    case 'put';
        echo "Metoda je PUT";
        break;
    case 'delete';
        echo "Metoda je DELETE";
        break;
}
?>
```

REST I HTTP

```php
<?php
$param=stream_context_create(array(
    "http"=>array(
            "method"=>"POST",
            "header"=>"Content-type: application/x-www-form-urlencoded",
            "content"=>"a=testZaPost"
)));
$odgovor=file_get_contents("http://putanja/do/servisa/servis.php?b=testzaGET", false, $param);
echo $odgovor;
?>
```




```php
<?php
//servis.php
$metoda=strtolower($_SERVER['REQUEST_METHOD']);
switch($metoda)
{
    case 'get';
        echo "Metoda je GET<br>";
        echo $_GET['b']."<br>";
        break;
    case 'post';
        echo "Metoda je POST<br>";
        echo $_GET['b']."<br>";
        echo $_POST['a']."<br>";
        break;
    case 'put';
        echo "Metoda je PUT";
        break;
    case 'delete';
        echo "Metoda je DELETE";
        break;
}
?>
```