Otvoreno računarstvo

3b. Otvorenost zapisa podataka

Creative Commons

Otvoreno računarstvo

3b. Otvorenost zapisa podataka

- **Formati zapisa podataka, CSV**
- XML
- JSON

U kojem obliku...

- ... zapisati podatke...
- ... da izbjegnemo ovo?

The badly thought-out use of Microsoft's Excel software was the reason nearly 16,000 coronavirus cases went unreported in England.

And it appears that Public Health England (PHE) was to blame, rather than a third-party contractor.

- https://www.bbc.com/news/technology-54423988

Excel: Why using Microsoft's tool caused Covid-19 results to be lost

By Leo Kelion
Technology desk editor

7 days ago

Coronavirus pandemic

The badly thought-out use of Microsoft's Excel software was the reason nearly 16,000 coronavirus cases went unreported in England.

Zašto bi mi to bilo bitno?

- Formati zapisa podataka, makar se činili sličnima:
  - nisu predviđeni za iste namjene
  - nemaju ista svojstva
  - nemaju iste mogućnosti provjera dobrog oblikovanja i valjanosti podataka
  - nemaju iste pomoćne alate i specifikacije, koje donose dodatne mogućnosti
  - nemaju jednako rasprostranjenu podršku
- Odabir formata podataka može **dugoročno uvelike utjecati** na razvoj sustava

Koji zapisi dolaze u obzir?

- Naš kontekst – tekstualni zapisi podataka
- Formati koje razmatramo:
  - CSV – Comma Separated Values
  - XML – eXtensible Markup Language
    - i još nekoliko dodatnih specifikacija za
      - provjeru valjanosti – DTD i XML Schema
      - Pretragu i lociranje – Xpath
      - transformacije i vizualizacije – XSLT, XSL-FO
  - JSON – JavaScript Object Notation
    - i još jednu specifikaciju u nastajanju
      - za provjeru valjanosti i opis podataka – JSON Schema
- Dodatni formati koje ne razmatramo, ali su potencijalno jako korisni:
  - HDF5 – Hierarchical Data Format
  - NetCDF – Network Common Data Form

CSV

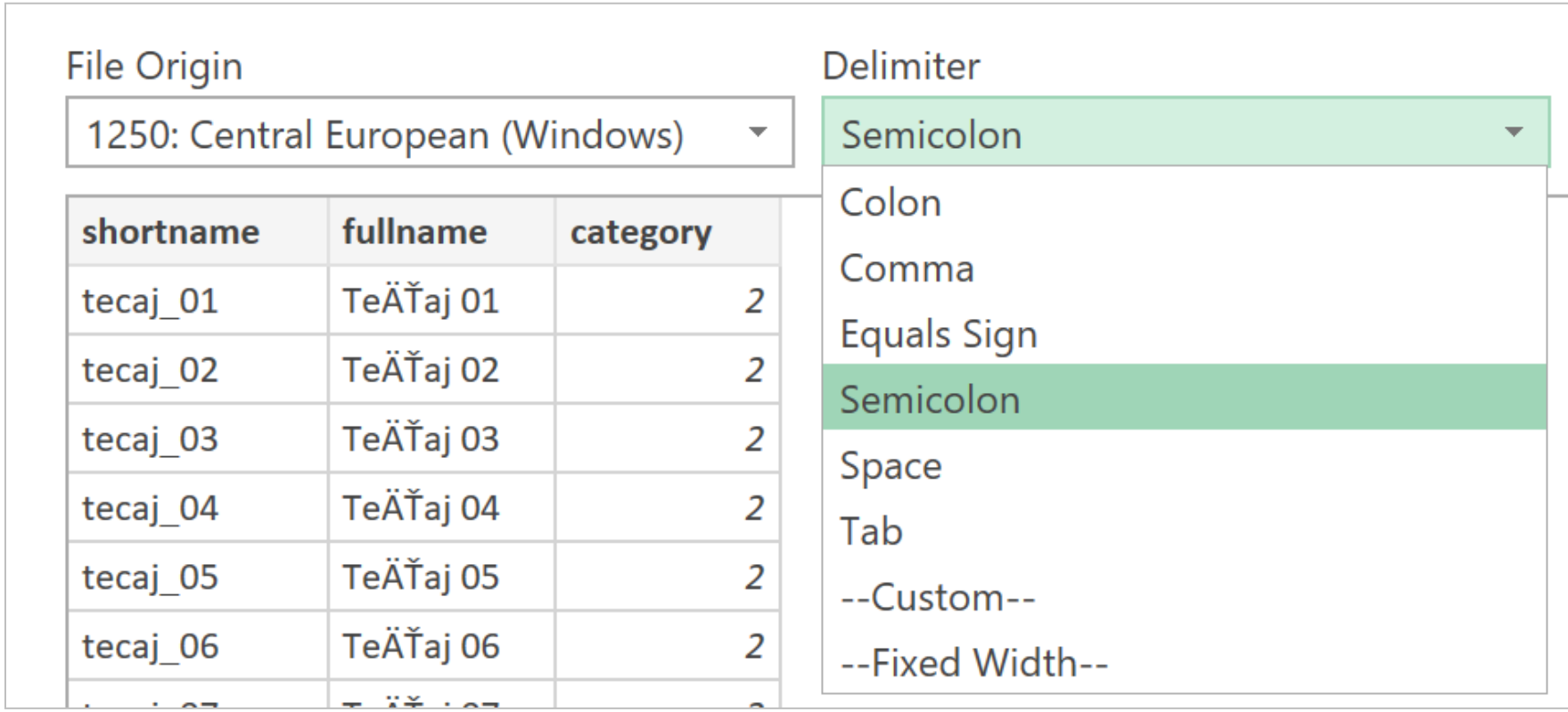
- Comma Separated Values
- „Standardiziran” (i nije baš) tek 2005: RFC 4180
  - MIME-type text/csv
- „... there is no formal specification in existence, which allows for a wide variety of interpretations of CSV files”
  - Svaki zapis u svome retku, retci odvojeni s CRLF
  - Posljednji redak može i ne mora imati CRLF
  - Opcionalni redak zaglavlja, s istim brojem stupaca
    - U MIME-tipu: opcionalni parametar „header”: present, absent
  - Polja odvojena zarezom (po specifikaciji, u praksi se često koriste i drugi znakovi)
  - Prazna mjesta se ne smiju ignorirati
  - Dodatna pravila o navodnicima

Na što treba paziti?

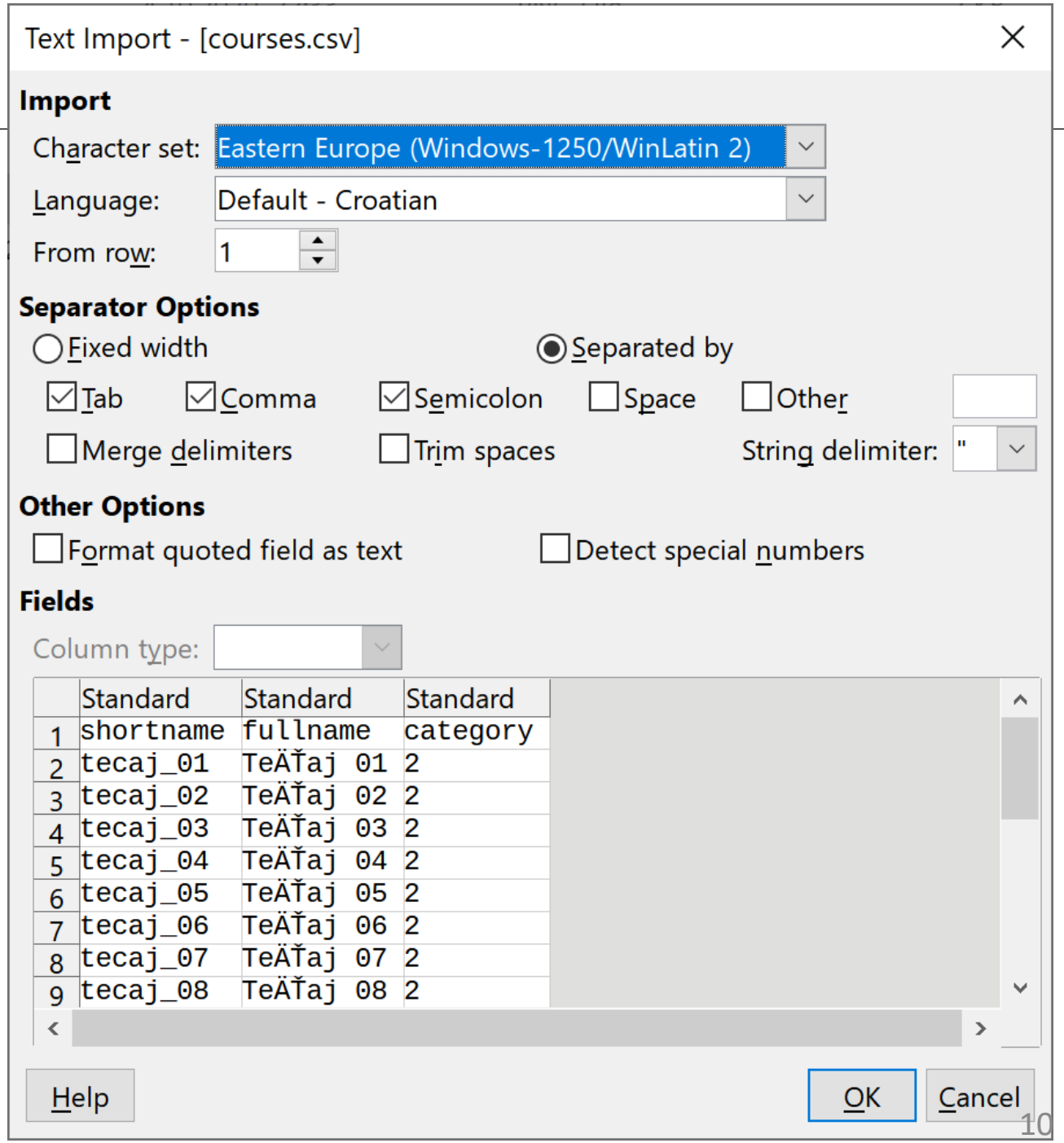
- Uvoz / Izvoz CSV-a
  - Oznaka novog retka?
  - Redak zaglavlja?
  - Delimiter?
  - Navodnici?
  - Kôdna stranica?

MS Office Excel

	A	B	C
1	shortname	fullname	category
2	tecaj_01	TeÄŤaj 01	2
3	tecaj_02	TeÄŤaj 02	2



LibreOffice Calc



Vrijedi uvijek – za svaki format!

- **Robustness principle** (*Postelov zakon*):

- "Be **liberal** in what you **accept**, and **conservative** in what you **send.**"
  - Internet protocol (RFC 760)
  - TCP protocol (RFC 793)
  - Requirements for Internet Hosts -- Communication Layers (RFC 1122)

CSV je kao tablica, bezveze...
Postoji li
nešto moćnije?

Otvoreno računarstvo

3b. Otvorenost zapisa podataka

- Formati zapisa podataka, CSV
- **XML**
- JSON

SGML – preteča XML-a, HTML-a (i ostalih ML-a)…

- **SGML** (*Standard Generalized Markup Language*)
  - ISO-8879 iz 1986.g.
    - 70-te: Goldfarb, Mosher, Lorie (IBM): GML (kasnije SGML)
  - Prva pojava meta jezika koji opisuje druge **jezike temeljene na oznakama** (*Markup Languages*)
  - Apstraktna sintaksa - može se konkretizirati na različite načine
    - npr.: definiranje šiljatih zagrada kao delimitera/odjeljnika između oznaka
- **Ključne komponente** označnih jezika:
  - elementi
  - atributi
  - tipovi podataka
  - DTD (*Document Type Definition*)

SGML - elementi

- Elementi definiraju "strukturu" dokumenta
- Sintaksa zapisa:
  - elementi omeđeni **šiljatim zagradama** < ... >
  - najčešće dolaze u parovima, npr. <u> i </u>
    - <u> početna oznaka (početak djelovanja imenovane oznake)
    - </u> završna oznaka (kraj djelovanja imenovane oznake)
  - sadrže dva osnovna dijela: **atribute** i **sadržaj**
  - tekst između početne i završne oznake - sadržaj oznake
    - Sadržaj oznake
    - neki nemaju sadržaj, npr.
 u HTML-u

SGML - atributi

- Atributi predstavljaju **svojstvo elementa**
- Parovi **ime=”vrijednost”**
- Pišu se unutar početne oznake elementa
- Primjer:
  -

SGML – tipovi podataka

- Tipovi služe za **definiranje tipa sadržaja**
- Podaci koji se ne parsiraju kao SGML (npr. CDATA, PCDATA)
  - Podaci o skriptama (izvođenje)
    - npr. u HTML-u: <script>
  - Podaci o stilu (prikazu)
    - npr. u HTML-u: <style>
- Posebni tipovi podataka (npr. NAME, ID, URI...)
  - imaju određena ograničenja ovisno o tipu

SGML – definicija tipa dokumenta

- DTD (Document Type Definition)
  - Definicija tipa dokumenta
- Sadrži **formalnu gramatiku**
- Validacija izraza
  - Primjer jednog DTD-a definiranog od strane W3C, za HTML 4.01:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

XML - eXtensible Markup Language

- Kao i HTML, temelji se na SGML-u
  - primjena u opisu, pohrani i razmjeni različitih podataka u tekstualnom obliku zapisa
- Mali broj jednostavnih, ali strogih pravila
- Orijentiran prema **strojnoj obradi**
  - uvjetuje **veće zahtjeve** pri radu sa sadržajem dokumenta (suprotno u odnosu na HTML)

XML - značajke

- Proširivost:
  - oznake se definiraju po potrebi
- Odjeljivanje podataka od prezentacije:
  - oznake **opisuju podatke**, **ne** njihov **izgled**
- Validacija:
  - stroga pravila - strojna provjera valjanosti i strukture
- Internacionalizacija:
  - XML izvorno koristi *Unicode* (UTF-8)
- Prenosivost:
  - XML-dokument je obična tekst datoteka
- Rasprostranjenost:
  - neovisan o platformi i dobro podržan u većini programskih jezika

XML – primjer dokumenta

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

<student jmbag="0036424454 5">
    <ime>Marko</ime>
    <prezime>Ferković - Enter</prezime>
    <institucija>
        <sveučilište>Sveučilište u Zagrebu</sveučilište>
        <fakultet id="0036">
            <ime>Fakultet elektrotehnike i računarstva</ime>
            <adresa>
                <ulica>Unska</ulica>
                <broj>3</broj>
                <pbr>HR-10000</pbr>
                <grad>Zagreb</grad>
            </adresa>
        </fakultet>
    </institucija>
</student>
```

XML – strukturiranje podataka

- XML ne propisuje oznake, već pravila definiranja
  - je li XML meta-jezik?
- Korisnici dokumenta XML tumače značenje oznaka
  - što npr. znači oznaka <ime>?
  - koje su sve oznake potrebne za opis podataka?
- Oznake definiraju hijerarhijske odnose podataka unutar dokumenta (odnos roditelj – dijete):

```
<student jmbag="0036424454S">
    <ime>Marko</ime>
    <prezime>Ferković - Enter</prezime>
</student>
```

XML – opis podataka

- Jezik za opis podataka:
  - konačan broj oznaka (rječnik)
  - struktura podataka (tvorba rečenica)
  - semantika (značenje riječi)
- Konačan broj oznaka i struktura
  - određuju se tipom dokumenta
  - tipovi se definiraju jezicima **DTD** ili **XML Schema**
- Semantika pojedinih oznaka definirana **implicitno**
  - mora biti dijeljena između svih entiteta koji koriste pojedini tip dokumenta (*shared semantics*)
- Eksplicitna definicija semantike – semantički Web

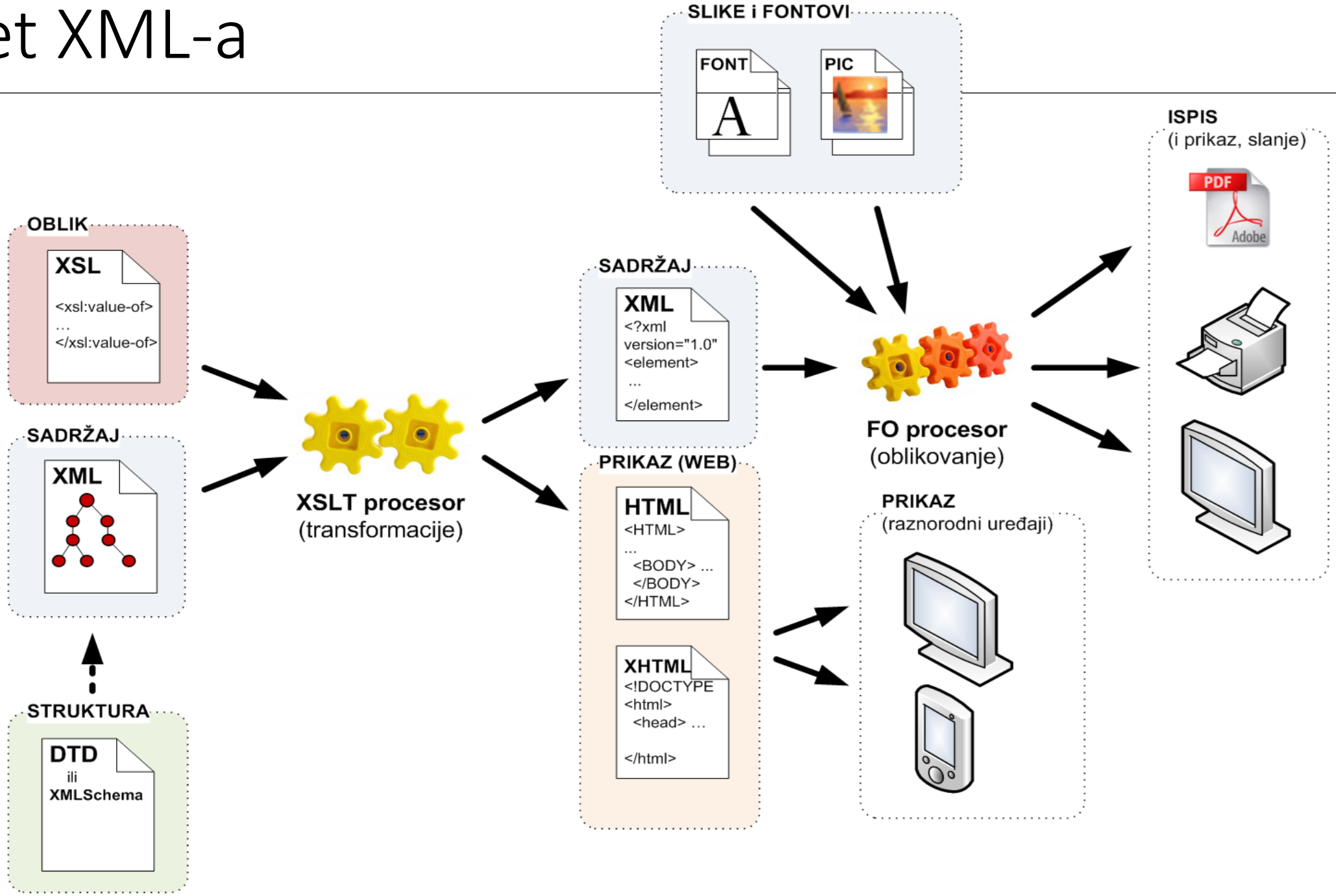
XML – opis podataka

- Normiranje jezika po domenama primjene
  - **horizontalne** domene
    - razmjena podataka između aplikacija u istoj domeni
      - npr. CAD sustavi, zdravstvo, e-business, logistika ...
    - normiranje onemogućuje ovisnost o proizvođaču aplikacije (*vendor lock-in*)
  - **vertikalne** domene
    - opis podataka prisutnih u svim domenama ili neovisnih o domenama primjene
      - npr. *XML stylesheets, SOAP* ...
  - razmjena podataka između **različitih domena**
    - npr. između zdravstva, osiguranja, MUP-a, biračkih popisa ...

Svijet XML-a

- Prateći jezici (preporuke i norme):
  - **DTD**, **XML Schema**, **XSL**, **XSLT**, **XPath**, XQuery, **DOM**, SAX, Namespaces, XLink, XPointer ...
  - XHTML, WAP ...
  - ODF, SMIL, **SVG**, CML, MathML, MusicML, OTA, ebXML, **OOXML** ...
  - **XML-RPC**, **SOAP**, WSDL ...

Svijet XML-a


SLIKE i FONTOVI
FONT
A
PIC
ISPIS
(i prikaz, slanje)
PDF
Adobe
OBLIK
XSL
<xsl:value-of>
...
</xsl:value-of>
SADRŽAJ
XML
SADRŽAJ
XML
<?xml
version="1.0"
<element>
...
</element>
XSLT procesor
(transformacije)
FO procesor
(oblikovanje)
PRIKAZ (WEB)
HTML
<HTML>
...
<BODY> ...
</BODY>
</HTML>
XHTML
<!DOCTYPE
<html>
<head> ...
</html>
PRIKAZ
(raznorodni uređaji)
STRUKTURA
DTD
ili
XMLSchema

XML – primjer sadržaja dokumenta

procesna naredba

komentar

element korijen (root)

element dijete (child)

prazan element

atribut

```
<?xml version=1.0"?>
<!-- komentar -->
<radnik>
    <ime> Marko </ime>
    <prazno/>
    <plaća valuta="kn">5000</plaća>
</radnik>
```

XML deklaracija

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

- Navođenje deklaracije na početku XML-dokumenta
  - preporučeno (ali nije nužno)
- Znak < mora biti prvi znak unutar teksta
- Atribut *version* mora biti naveden kao prvi atribut
  - vrijednost mora biti "1.0"
- Atribut *encoding* je neobavezan, vrijednost mora sadržavati oznaku kôdne stranice dokumenta
  - UTF-8 ...
- Atribut *standalone* je neobavezan
  - označava ovisnost dokumenta o vanjskim entitetima

Korijenski element

```
<?xml version=”1.0” encoding=”UTF-8” ?>
<doc>
<content readonly='true'>
       <chapter> ...  </chapter>
</content>
<content> ... </content>
</doc>
```

- Mora postojati isključivo jedan osnovni, korijenski element dokumenta
- Sadržaj dokumenta je unutar korijenskog elementa, u obliku elemenata, atributa, teksta, itd.
- Unutar korijenskog elementa moguće navođenje
  - komentara, procesnih naredbi, *whitespace*, CDATA blokova, entiteta itd.

Element

- **<ime_elementa>** sadržaj **</ime_elementa>**

- Element - osnovna gradivna komponenta XML-dokumenta, sastoji se od:
  - početne oznake (<ime_elementa>),
  - završne oznake (</ime_elementa>) i
  - sadržaja omeđenog oznakama.
- Elementi se imenuju po entitetima domene čiji se podaci opisuju npr.:
  - jmbag, ime, prezime, predmet, ocjena ...
    - iz domene poslovnih procesa na sveučilištu
- Imena elemenata su *case-sensitive*:
  - <jmbag> ≠ <JMBAG>

Imenovanje elemenata

- Valjana imena elementa:

  `([slovo]|[_])([0-9][slovo][znak])*`

- Razmak se ne smije koristiti u tvorbi imena
- Ime elementa ne smije započeti s *xml*
  - u imenu se može koristiti većina Unicode znakova
- Prvi znak imena odmah nakon oznake "**<**"
- Primjeri dobro oblikovanih imena elemenata:

  `<jmbag>, <first_name>, <first-name>, <šifra>, <_note>,`

  `<mbr >, <velika.seoba.naroda>`

- primjeri krivih imena elemenata:

  `<boja očiju>, <1.ples>, <  jmbag>, <-HT-R-A>`

Sadržaj elementa

- Sadržaj elementa:
  - sve između početne i završne oznake
- Vrste sadržaja:
  - Prazan

```
<ime_elementa></ime_elementa>

<ime_elementa />
```

  - Jednostavan

```
<ime_elementa>
Sadržaj u obliku
teksta bez oznaka...
</ime_elementa>
```

  - Elementi djeca

```
<vanjski_element>
        <unutarnji_element>
        ...
        </unutarnji_element>
        <unutarnji_element>
        ...
        </unutarnji_element>
</vanjski_element>
```

  - Miješani

```
<content>
Metodu su razvili
<name>Cholsky</name> i
<name>Rangapathra</name>.
</content>
```

Atributi

```
<ime_elementa ime_atributa=”vrijednost”>
        ...
</ime_elementa>

<prazan_element ime_atributa=”vrijednost” />
```

- Atribut mora pripadati elementu
- Element može imati jedan ili više atributa
- Atribut čini uređeni par *ime=”vrijednost”*

Pravila imenovanja atributa

- Pravila imenovanja atributa ista kao za elemente
- Unutar istog elementa atributi moraju imati različita imena
- Vrijednost atributa mora biti navedena unutar navodnika
  - jednostrukih ili dvostrukih
- Sadržaj atributa može se sastojati samo od teksta
- Razmak između riječi u sadržaju može se tretirati kao odjeljivanje dviju vrijednosti u nizu vrijednosti pojedinog atributa

```
<flag colors=”red white blue”/>
```

Kada elementi, a kada atributi?

- Sadržaj elementa
  - strukturirani podaci, jednostavno proširenje ili dodavanje višestrukih vrijednosti, jednostavniji za obradu
- Atribut pridijeljen elementu
  - podatak o elementu (meta podatak), primjer:

```
<lecture>
      <desc lang=”en”> ..... </desc>
</lecture>
```

- Preporuke:
  - kad god je moguće, za zapis podataka koristiti elemente
  - atribute koristiti za meta podatke i za definiranje logičke strukture podataka

- *When to use elements vs attributes?*
  - ~~https://www.ibm.com/developerworks/library/x-eleatt/index.html~~ -> Web Archive

Naredbe obrade

```
<?ime_naredbe lista_atributa ?>
```

- *processing instruction*
- Naredbe aplikacijama za obradu XML-dokumenata
- Ime i atributi ovisni o tome kojoj aplikaciji je naredba namijenjena
- Ako aplikacija ne razumije naredbu, ignorira je
- Primjer:
  - naredba za primjenu *stylesheeta* za oblikovanje XML-dokumenta

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

Komentari

```
<!-- ovo je jedan redak komentara -->

<!-- komentar se proteže preko više redaka
...
-->
```

- Oznaka početka komentara **<!--**
- Oznaka kraja komentara **-->**
- Sadržaj omeđen početnom i krajnjom oznakom komentara parser ignorira
- Nema gniježđenja komentara
  - na prvu detekciju oznake kraja **-->** parser terminira komentar

CDATA blokovi

- `<![CDATA[ Ako je "x < y" & "z > y" onda je "z > x " ]]>`

- Parser ignorira CDATA blokove
  - nema prepoznavanja elemenata, ekspanzije entiteta ...
- Označavanje blokova koji sadrže veći broj nedozvoljenih znakova
  - npr. dijelovi programskog kôda, formule ...
- Podaci unutar bloka ne smiju sadržavati slijed znakova]]>

Bijeli znak (*Whitespace*)

- Različitost terminiranja retka teksta:
  - DOS & Windows: **CR/LF**, UNIX: **LF**, Mac: **CR**
- Preporučeno korištenje LF za terminiranje retka
- Kao bijeli znakovi tretiraju se skupno:
  - razmak (' '), tab ('\t'), CR(ASCII 13, '\r'), LF (ASCII 10, '\n')
- Dvije vrste bijelih znakova:
  - **važan**: dio sadržaja dokumenta, mora biti sačuvan nakon parsiranja
  - **nevažan**: primjena u formatiranju dokumenta, ne mora biti sačuvan nakon parsiranja
- Svi bijeli znakovi se, u općem slučaju, čuvaju tijekom parsiranja dokumenta

Entiteti

- Entitet:
  - dio XML-dokumenta
- XML-dokument sastoji se od jednog ili više entiteta
- Vrste entiteta:
  - **parsirani**: datoteka s xml sadržajem
  - **neparsirani**: binarna datoteka
- Ugrađeni i naknadno definirani entiteti
- Unutarnji i vanjski entiteti
  - **unutarnji**: definirani unutar tipa dokumenta
  - **vanjski**: zaseban resurs, mora se prilikom parsiranja dohvatiti (slično #include direktivama u C/C++)

Referenciranje entiteta

- Referenciranje entiteta unutar dokumenta:

```
&ime_entiteta;
```

- Referenca se u postupku parsiranja zamjenjuje sa sadržajem entiteta
  - dobiva se parsirani dokument
- Dodavanje znakova u dokument,
  - npr. Unicode znak π: π
- Korištenje naknadno definiranog entiteta π
  - entitet mora biti definiran u sklopu definicije tipa dokumenta

Ugrađeni entiteti

<	lt
>	gt
&	amp
'	apos
“	quot

Problem konflikta imena elemenata

- Imena nastaju neovisno, npr.
  - *eIndeks* i potvrda o liječničkom pregledu
- Problem kod spajanja
  - sadržaji iz neovisnih izvora
    - za opis različitih koncepata mogu se koristiti elementi jednakih imena,
      a različite semantike, sadržaja, strukture ...
- Potrebno dodatno označiti
  - o kojem se točno konceptu (elementu) radi

```
<eindex version='1.0'>
<student>
<id>0036435453</id>
<firstName>Marko</firstName>
<status>A</status>
</student>
<course>
<title>OpenComputing</title>
<semester>6</semester>
        <exam>
<date>2008-07-04</date>
<grade>C</grade> ...
</eindex>
```

```
<UMed>
   <exam>
      <id>A65ZGB00456345F</id>
         <date>
    <year>2008</year>
    <month>02</month>
    <day>17</day>
         </date>
      <status>45</status>..
   </exam>
</UMed>
```

Prostori imena

- Rješenje na razini sintakse:
  - korištenje prefiksa za razlikovanje različitih elemenata jednakog imena

    <p:id/> ≠ <q:id/>
  - p:id i q:id su valjana imena elemenata, koja parser tretira kao i ostala valjana imena elemenata
- Rješenje na logičkoj razini:
  - p i q su dva različita prostora imena (*namespaces*);
  - svi elementi s istim prefiksom pripadaju zasebnom prostoru imena
    - p:id pripada prostoru imena p
- Prefiksi se definiraju za svaki dokument zasebno
  - globalna definicija prefiksa ne bi riješila problem!

Jedinstveni identifikator prostora imena

- **URI** kao globalno jedinstveni identifikator resursa, u ovom slučaju prostora imena
  - http://www.w3.org/TR/html4/
  - http://www.w3.org/1999/XSL/Transform
  - http://www.fer.hr/eindex/ver1.0

  - temelji se na hijerarhijskoj podjeli odgovornosti imenovanja domena i upravljanju resursima npr.
    - ICANN*-CARNet-FER

  - stvaran resurs reprezentiran URI-jem **ne mora postojati**

*Internet Corporation for Assigned Names and Numbers

Deklariranje prostora imena

- Povezivanje prefiksa (lokalna informacija) i URI-ja (globalna informacija) atributom xmlns:

```
xmlns:namespace-prefix=”namespaceURI”
<ex:eindex xmlns:ex=”http://www.fer.hr/eindex/ver1.0”>
    …
</ex:eindex>
```

- Doseg prostora imena je element njegove deklaracije uključujući sadržaj
  - u primjeru, doseg ex je element eindex
- Podrazumijevani (eng. *default*) prostor imena:

```
xmlns=”namespaceURI”
<eindex xmlns=”http://www.fer.hr/eindex/ver1.0”>
    …
</eindex>
```

Primjer uporabe prostora imena

```
<ex:eindex xmlns:ex=”http://www.fer.hr/eindex/ver1.0” ex:version='1.0'>
    <ex:student>
<ex:studentID>0036435453</ex:id>
<ex:firstName>Marko</ex:firstName> ...
    </ex:student>
    <ex:courses>
<ex:course>
        <ex:title>OpenComputing</ex:title> ...
<ex:exam>
        <ex:date>2008-07-04</ex:date>
        <ex:grade>C</ex:grade> ...
    <ex:attachments>
<med:UMed xmlns:med=”http://www.hzzo-net.hr/hrmedSchema/2008-01>
        <med:exam>
                <med:id>A65ZGB00456345F</med:id>
                <med:date>
                        <med:year>2008</med:year>
                        <med:month>02</med:month>
                        <med:day>17</med:day>
                </med:date>
                <med:status>45F</med:status> ...
</med:UMed> ...
    </ex:attachments>
</ex:eindex>
```

Dobro oblikovani XML-dokument

- Dobro oblikovani (*well-formed*) XML-dokument zadovoljava sva prethodno navedena pravila
  - strukture dokumenta
  - imenovanja elemenata i atributa
  - gniježđenja elemenata ...

- Parseri odbijaju obradu loše oblikovanih XML-dokumenata

- Primjer dobro oblikovanog XML-dokumenta:

```
<ž/>
```

Ekvivalentnost XML-dokumenata

- **Logička ekvivalentnost**:
  - jednak sadržaj dokumenata

```
<student><jmbag>003643443</jmbag><ime>Marko</ime> <prezime>Ferković -
  Enter</prezime></student>
```

```
<student>
    <ime>Marko</ime>
    <prezime>Ferković - Enter</prezime>
    <jmbag>003643443</jmbag>
</student>
```

- **Fizička ekvivalentnost**:
  - datoteke jednake na razini okteta
    - npr. korištenje alata *diff* ne pokazuje razlike između datoteka
  - svođenje na kanonički oblik (struktura, poredak atributa, kodiranje ...) za usporedbu fizičke ekvivalentnosti

A što ću sad s
dobro oblikovanim
XML-om?
Kako tumačiti što u njemu piše?

Otvoreno računarstvo

3b. Otvorenost zapisa podataka

- Formati zapisa podataka, CSV
- **XML**
- JSON

Definicija tipa dokumenta

- XML-dokument sam po sebi
  - označava sadržaj i određuje hijerarhijsku strukturu
- Ako nemamo definirani tip dokumenta
  - oznake i hijerarhijska struktura (gniježđenje) su potpuno nedefinirane i mogu se koristiti potpuno proizvoljno
- Definicija tipa dokumenta mora odrediti:
  - oznake koje se mogu koristiti (konačan broj)
  - sve moguće hijerarhijske strukture koje se u dokumentu mogu pojaviti
    - ne mora ih biti konačan broj
- Tip dokumenta
  - eksplicitno **ne određuje značenje** oznaka i struktura, no podrazumijeva da to znanje dijele korisnici dokumenata

Valjanost XML-dokumenta

- Definicija tipa dokumenta efektivno definira jezik oznaka
  - riječi + tvorba rečenica + (implicitno) značenje
  - jezik za definiranje opisnih jezika
  - meta-jezik
- Ako je XML-dokumentu pridijeljen tip:
  - parser koristi definiciju tipa za provjeru rječnika dokumenta, vrijednosti elemenata i atributa te strukture dokumenta
- **XML-dokument koji zadovoljava provjeru tipa dokumenta je valjan** (*valid*)

- **Valjani XML-dokument mora biti dobro oblikovan**
- **Dobro oblikovan XML-dokument ne mora biti valjan**

Zašto provjera valjanosti?

- Zašto uopće koristiti provjeru valjanosti dokumenta?
  - “Provjera valjanosti je obična gnjavaža”
  - Definiranje dobrog tipa dokumenta, koji će biti koristan dulje vrijeme, zahtijeva puno vremena i razmišljanja

- ALI:

  - Tjera na pridržavanje norma, preporuka, dogovora
  - Sprječava velike probleme tijekom korištenja, u komunikaciji i suradnji različitih sustava, potencijalno iz različitih organizacija
  - Identificira krivca za probleme u komunikaciji

XML DTD

- DTD – Document Type Definition
- Skup deklaracija o oznakama koje definiraju tip XML-dokumenta, po određenim pravilima
- Može i ne mora biti deklariran
- Ako jest, (validirajući) parser prilikom učitavanja provjerava i dobru oblikovanost i valjanost, javlja pogreške
  - Nije svaki parser validirajući – npr. preglednici weba!

Gdje se nalazi DTD?

- Unutarnja definicija:

  <!DOCTYPE korijenski-element [DTD definicija]>
- Vanjska definicija:

  <!DOCTYPE korijenski-element SYSTEM URI>

  <!DOCTYPE korijenski-element PUBLIC FPI URI>

- Kombinirana definicija:

  <!DOCTYPE korijenski-element SYSTEM URI [DTD definicija]>

  <!DOCTYPE korijenski-element PUBLIC FPI URI [DTD definicija]>

  Korijenski-element
  - bilo koji element definiran unutar DTD-a

Gdje se nalazi DTD?

- **Unutarnja definicija**: DTD se nalazi unutar XML-dokumenta
  `<!DOCTYPE korijenski-element [DTD definicija]>`

- **Vanjska definicija:** DTD se nalazi u zasebnoj datoteci ili skupu datoteka
  - za provedbu provjere valjanosti, parser mora dohvatiti datoteku s definicijom

  - **Privatna** vanjska definicija – koristi se u **užem krugu** (osobe, organizacija), nije za javnu distribuciju

    `<!DOCTYPE korijenski-element SYSTEM URI>`

  - **Javna** vanjska definicija – namijenjena **javnom korištenju** (*de facto/de iure* norma?)

    `<!DOCTYPE korijenski-element PUBLIC FPI URI>ž`

FPI – Formal Public Identifier

norma//odgovoran//tip-dokumenta//jezik

norma:

-	(nije norma)
+	(odobrilo nestandardno tijelo)
ime-norme	

odgovoran: ime grupe odgovorne za održavanje DTD-a

tip-dokumenta: što definira, inačica dokumenta

jezik: ISO 639 (dva ili tri slova identificiraju jezik)

-//UNIZG-FER-ZARI-RASIP//eIndex Ver. 1.0//HR

-//W3C//DTD HTML 4.0 Transitional//EN

Usporedba deklaracija

- Unutarnja deklaracija DTD-a:
  - DTD uvijek dostupan, nije ga potrebno dodatno dohvaćati

- Vanjska deklaracija DTD-a:
  - manje datoteke dokumenta (parser može držati DTD-dokumente u priručnoj memoriji)
  - dijeljenje standardne definicije tipa između više korisnika

Što provjerava DTD?

- Građevne komponente XML-dokumenta (po DTD-u):
  - elementi
    - rječnik i struktura tipa dokumenta
  - atributi
    - dodatni podaci o elementima
  - entiteti
    - građevne komponente XML-dokumenta

DTD - elementi

- Elementi se deklariraju ključnom riječi ELEMENT

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

- Pet vrsta sadržaja:
  - Prazan
    - `<!ELEMENT ime-elementa EMPTY> -> <hLine>  </hLine> ili <hLine/>`
  - Jednostavan
    - `<!ELEMENT ime-elementa (#PCDATA)> -> <lastName>Ferković - Enter</lastName>`
  - Slobodan
    - `<!ELEMENT ime-elementa ANY>`

```
<misc><NSK><posudba><datum> ...</NSK></misc>
```

  - Elementi s djecom
    - `<!ELEMENT element-roditelj (elementi-djeca)>`

```
<html>
    <head> ... </head>
    <body> ... </body>
</html>
```

  - Miješani
    - `<!ELEMENT element(#PCDATA|element1|element2|..)*>`

```
<name>dr.sc.
<firstName>Marko</firstName><lastName>Ferković -
Enter></lastName></name>
```

Kardinalnost i kombiniranje elemenata djece

- Točno jedan element – ne navodi se indikator (prethodni primjer)
- Najmanje jedan element – indikator **+**

  <!ELEMENT roditelj (dijete+)>
- Nijedan ili više elemenata – indikator *****

  <!ELEMENT roditelj (dijete*)>
- Nijedan ili jedan element – indikator **?**

  <!ELEMENT roditelj (dijete?)>
- Operator slijeda elemenata '**,**'
  - u dokumentu moraju biti navedeni u definiranom slijedu

    <!ELEMENT roditelj (dijete1, dijete2, dijete3)>
- Operator alternative (XOR) '**|**'
  - ili jedan ili drugi element

    <!ELEMENT roditelj (dijete1 | dijete3)>
- Operator grupiranja **( )**
  - niz elemenata unutar zagrada tretira se kao jedan element kod primjene ostalih operacija i indikatora

    <!ELEMENT roditelj ((dijete1 | dijete2)+, dijete3)>

Atributi

```
<!ATTLIST element atribut tip default-vrijednost>

<!ATTLIST element atribut1 tip default-vrijednost
                  atribut2 tip default-vrijednost
                  ...
                  atributN tip default-vrijednost     >
```

```
<!ATTLIST course courseID CDATA “-”>

<course courseID=”479”> ... </course>
```

- Element čiji se atribut definira mora biti definiran
- Ime atributa mora biti jedinstveno za pojedini element

Tipovi atributa

```
<!ATTLIST element atribut tip default-vrijednost>
```

CDATA	neparsirani znakovni podaci
(en1|en2|en3...)	jedna od vrijednosti iz liste (enumeracija)
ID	jedinstveni identifikator
IDREF	vrijednost postojećeg jedinstvenog identifikatora
IDREFS	lista vrijednosti postojećih jedinstvenih identifikatora
NMTOKEN	valjano ime u XML-u
NMTOKENS	lista valjanih imena u XML-u
ENTITY	entitet
ENTITIES	lista entiteta

Podrazumijevane vrijednosti

```
<!ATTLIST element atribut tip default-vrijednost>
```

vrijednost	podrazumijevana vrijednost atributa
#REQUIRED	obavezna vrijednost atributa
#IMPLIED	neobavezna vrijednost atributa
#FIXED *vrijednost*	nepromjenljiva vrijednost

Nedostaci DTD-a

- Nema provjere valjanosti podataka kod jednostavnih sadržaja!
- Nužnost provjere nameće korištenje atributa umjesto elemenata za pohranu podataka!
- Slaba provjera tipova podataka kod atributa
- Nespretno riješeno definiranje kardinalnosti
- Nepostojeća podrška za prostore imena
- DTD nije XML - nije dobro oblikovan dokument

DTD ne zadovoljava moje potrebe za validacijom XML-a! Što ću sada?

XSD – XML Schema Definition

- W3C preporuka donesena 2001. godine
- Namjena kao i DTD, ispravlja brojne nedostatke kao što su:
  - definiranje različitih tipova sadržaja elemenata i atributa
  - jednostavnije i preciznije mogućnosti definiranja strukture
  - definiranje novih tipova i formata zapisa podataka
  - podrška prostorima imena
  - Schema je XML, DTD nije
  - ...

XSD – početne deklaracije

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  ...

</xsd:schema>
```

- XML Schema dokument mora biti dobro oblikovan i valjan.
- Korijenski element XSD dokumenta je **`schema`**.
- Dobra praksa:
  - deklarirati prostor imena za XSD elemente (**`xsd`**, **`xs`**, ...) kako ne bi došlo do konflikta s imenima elemenata, tipova, atributa itd. jezika kojeg se definira.

Od čega se sastoji XSD?

- Osnovne komponente definicije tipa dokumenta u DTD-u:

  elementi `<!ELEMENT name (content)>`

  atributi `<!ATTLIST element name type default-value>`

- XSD koristi tri osnovne komponente:

  elementi `<xsd:element name=”ime” type=”tip” ...>`

  atributi `<xsd:attribute name=”ime” type=”tip” ...>`

  tipovi `<complexType name=”ime” ...>` | `<simpleType name=”ime” ...>`

Tipovi u XSD-u

- Sadržaj elemenata i atributa u XSD definiran s njihovim tipom
  - tip je ravnopravna komponenta sheme!
  - u DTD-u jednostavni tipovi podržani samo kod atributa
- Općenito, tip je definiran vrstama podataka koje može sadržavati...
  - jednostavan sadržaj u obliku niza znakova
  - elemente (koji su pak određenog tipa...)
  - atribute (također određenog tipa...)

Tipovi u XSD-u

- XSD definira dvije vrste tipova:
  - jednostavne i složene
- Jednostavni tipovi:
  - jednostavan sadržaj, npr. string, broj, URI, datum ...
  - ne sadrže ni elemente ni atribute
  - provjera tipa sadržaja kod parsiranja
    - npr. element tipa integer ne može imati vrijednost “534ž45”
  - DTD: element s (#PCDATA) sadržajem, bez atributa
- Složeni tipovi:
  - sadrže elemente i/ili atribute
  - sličnost s ostalim vrstama sadržaja elemenata iz DTD-a

Definiranje tipova u XSD-u

- Ugrađeni jednostavni tipovi (>40)
- Definiranje novih jednostavnih tipova **ograničavanjem** (ugrađenih ili novo-definiranih) jednostavnih tipova
- Ne postoje ugrađeni složeni tipovi, moraju se definirati
- Novi složeni tipovi definiraju se:
  - dodavanjem atributa jednostavnim tipovima
  - navođenjem elemenata (i atributa) koji čine složeni tip
  - proširenjem ili ograničenjem definiranih složenih tipova

Primjer: Ograničavanje jednostavnih tipova podataka

- Ograničavanjem jednostavnog tipa podataka definira se novi jednostavni tip podataka
- Ograničavaju se vrijednosti koje novi tip podatka može sadržavati, u ovisnosti o tipu podataka koji se ograničava

```
<xsd:simpleType name="percentage”>
    <xsd:restriction base="xsd:integer">
        <xsd:minInclusive value="1"/>
        <xsd:maxInclusive value="100"/>
    </xsd:restriction>
</xsd:simpleType>
```

Primjer: deklaracija i korištenje tipova podataka

- Korištenje ugrađenih tipova podataka

```
<xsd:element name="ime" type="xsd:string"/>
        <ime>Marko Ferković - Enter</ime>
<xsd:element name=”studenata-u-grupi” type=”xsd:integer”/>
        <studenata-u-grupi>15</studenata-u-grupi>
<xsd:element name=“datum-upisa” type=”xsd:date”/>
        <datum-upisa>2006-07-22</datum-upisa>
```

- Korištenje naknadno definiranih jednostavnih tipova podataka

```
<xsd:element name=”ocjena” type=”grade”/>
        <ocjena>B</ocjena>

<xsd:element name=”uspješnost” type=”percentage”/>
        <uspješnost>76</uspješnost>
```

Primjer: atributi u XSD-u

- Deklariranje atributa u XSD dokumentu:

```
<xsd:attribute name="attr-name" type="type-name"/>
```

- Atributi su isključivo jednostavnog tipa
  - vrlo su slični jednostavnim elementima, sastavni su dio složenih tipova
-

DTD:

```
<!ATTLIST element attr-name type default-value>
```

XSD:

```
<xsd:attribute name="lang" type="xsd:string"/>
<xsd:attribute name="version" type="xsd:decimal"/>
<xsd:attribute name="kamata" type="pct"/>
```

Kada DTD, a kada XSD?

- DTD
  - za jednostavnije, kraće dokumente
    - bitna je provjera strukture, same vrijednosti elemenata i atributa nisu toliko bitne
    - mogu se uređivati “ručno”, korištenjem običnih uređivača teksta
- XML Schema (XSD)
  - za složene dokumente
    - bitna je provjera i strukture i tipova vrijednosti elemenata i atributa
    - definicija duža u odnosu na ekvivalentnu definiciju u DTD-u, prvenstveno namijenjeno uređivanju pomoću namjenskih alata (grafička sučelja, manipuliranje simbolima umjesto tekstom ...)

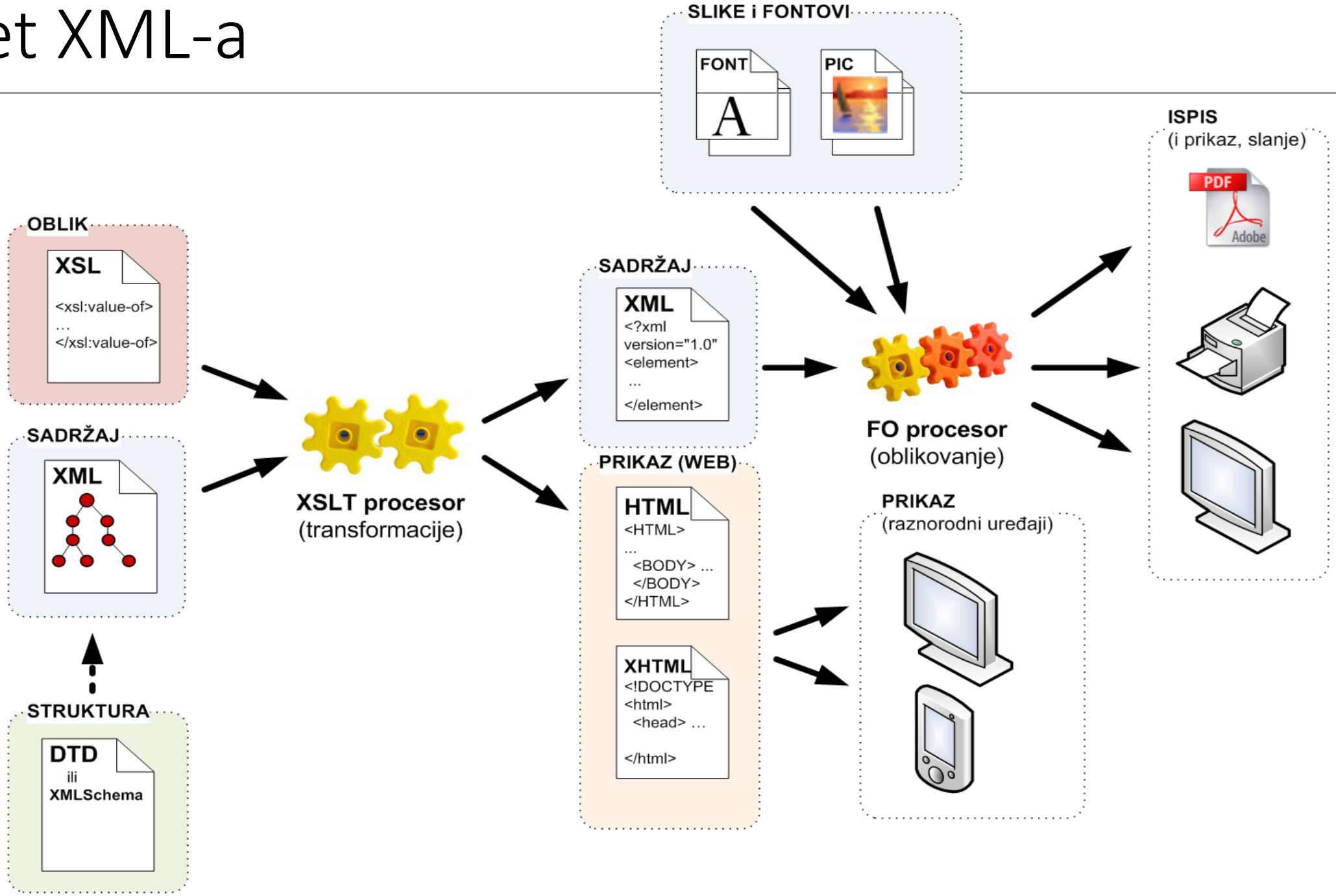
A što ću sad s
dobro oblikovanim, valjanim
XML-om?

Otvoreno računarstvo

3b. Otvorenost zapisa podataka

- Formati zapisa podataka, CSV
- **XML**
- JSON

Svijet XML-a


SLIKE i FONTOVI
FONT
A
PIC
ISPIS
(i prikaz, slanje)
PDF
Adobe
OBLIK
XSL
<xsl:value-of>
...
</xsl:value-of>
SADRŽAJ
XML
<?xml
version="1.0"
<element>
...
</element>
SADRŽAJ
XML
XSLT procesor
(transformacije)
FO procesor
(oblikovanje)
PRIKAZ (WEB)
HTML
<HTML>
...
<BODY> ...
</BODY>
</HTML>
XHTML
<!DOCTYPE
<html>
<head> ...
</html>
PRIKAZ
(raznorodni uređaji)
STRUKTURA
DTD
ili
XMLSchema

Alati za pregled i uređivanje XML-a

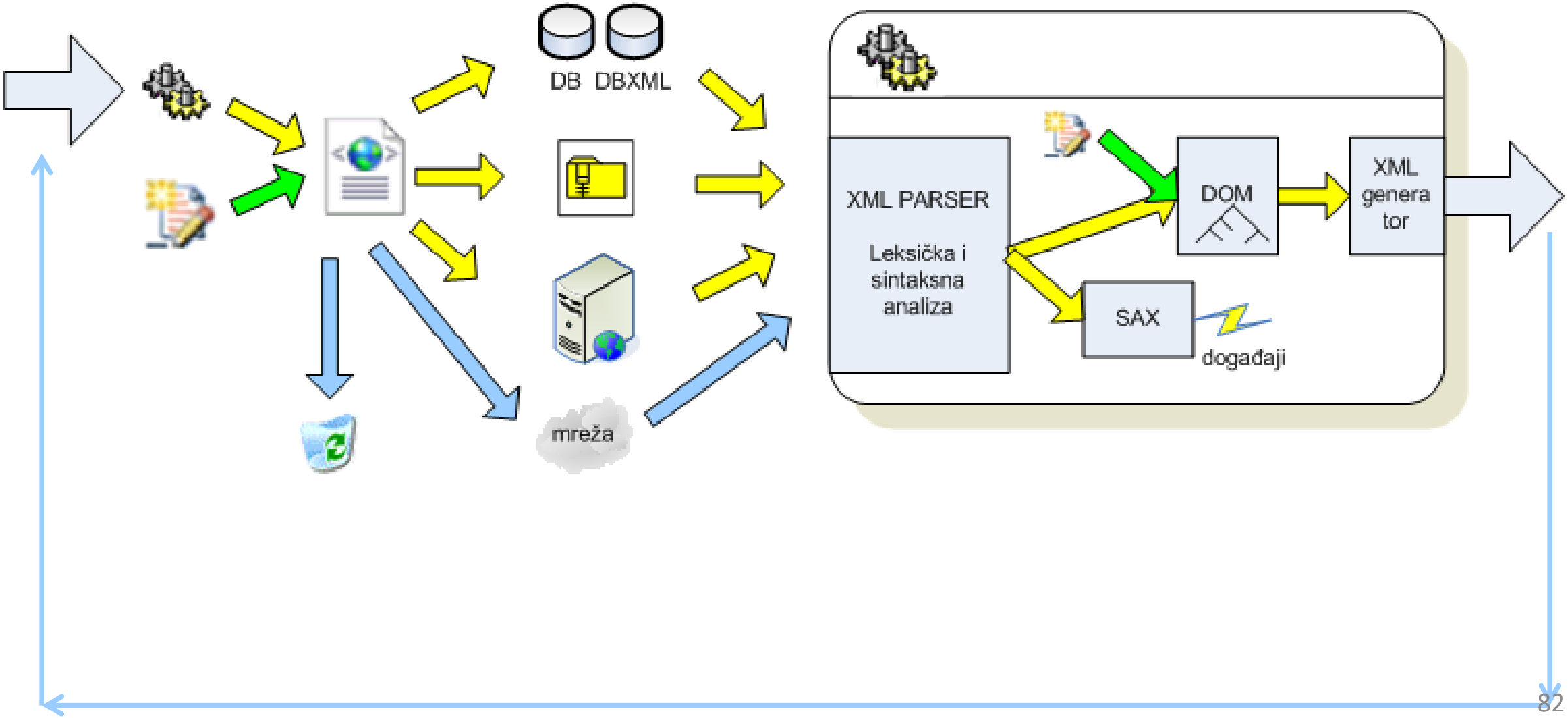
- Prikaz XML-dokumenata:
  - bilo koji uređivač teksta opće namjene
    - samo sadržaj
  - *Netscape/Firefox, Intenet Explorer, Opera, Safari*,...
    - sadržaj i struktura dokumenta, verifikacija, formatiranje, transformacije

- Alati za uređivanje:
  - uređivači teksta opće namjene (npr. *Notepad*, *Notepad++*, *vi*, *joe*)
  - integrirana razvojna okruženja (nativno ili uz pomoć dodatka)
  - prilagođeni uređivači (npr. *XMLNotepad*)
  - složeni alati za razvoj (npr. *XMLSpy*)

Pohrana podataka

- Skalabilnost
  - jednostavne datoteke (npr. konfiguracijske datoteke aplikacije)
  - pohrana složenih dokumenata, koje isključivo koriste aplikacije (npr. ova prezentacija)
- XML datoteke stvaraju
  - ljudi (u pravilu jednostavne)
  - aplikacije (od jednostavnih do vrlo složenih)
- XML datoteke gotovo isključivo koriste aplikacije

Životni vijek XML-dokumenta



Životni vijek XML-dokumenta

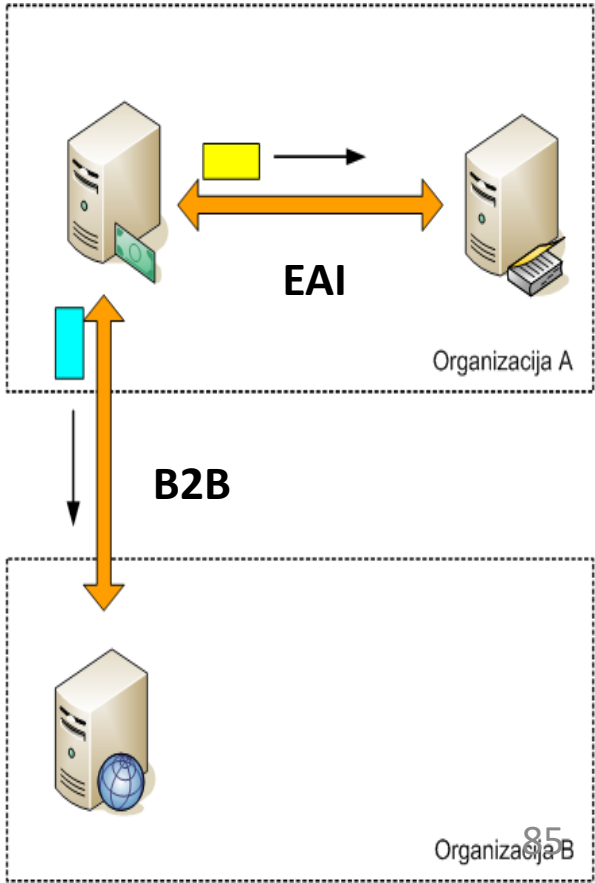
- XML-dokumenti po duljini životnog vijeka:
  - privremeni dokumenti (*transient*)
  - trajni dokumenti (*persistent*)
- **Privremeni** dokumenti se ne arhiviraju
  - ostaju u radnoj memoriji
  - ili putuju mrežom (u obliku poruka)
  - ili se privremeno zapisuju u datotečnom sustavu tek radi razmjene s drugim aplikacijama
- **Trajni** dokumenti arhiviraju se u datotečnom sustavu ili u bazi podataka
  - mogu biti dostupni i putem mreže (Web ...)

Arhiviranje XML-datoteka

- Izvorne XML-datoteke su tekstualne datoteke
  - mogu se koristiti alati za održavanje inačica (npr. *Subversion*)
- XML-datoteke mogu biti sažete
  - više srodnih datoteka može biti sažeto unutar iste arhive
    - npr. ova prezentacija: *XML.odp*
  - vrlo visok stupanj sažimanja XML datoteka
- Baze podataka:
  - *Native XML* baze – hijerarhijske baze podataka
  - relacijske baze podataka s mapiranjem
    XML ↔ relacijski model
- Pretraživanje dokumenata i baza dokumenata

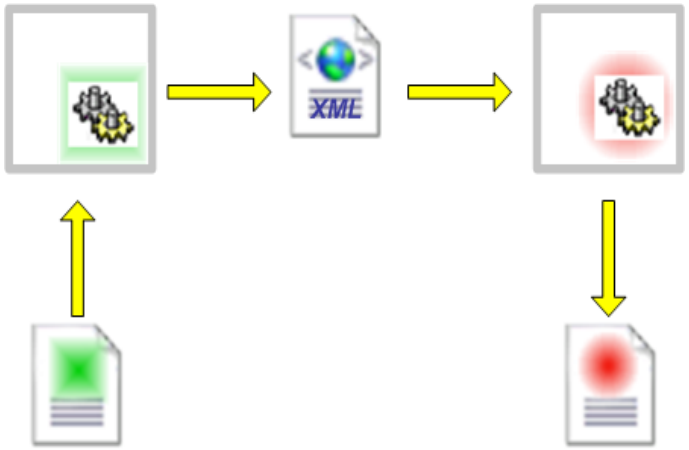
XML u integraciji aplikacija

- Različite aplikacije **unutar** organizacije moraju razmjenjivati podatke
  - npr. skladište i financije
  - Enterprise Application Integration - EAI
- Različiti sustavi **različitih** organizacija moraju razmjenjivati podatke:
  - npr. e-narudžbe, e-plaćanje, e-PDV
  - B2B, B2G
- Asinkrona komunikacija mrežom slanjem poruka u obliku **privremenih XML-dokumenata**

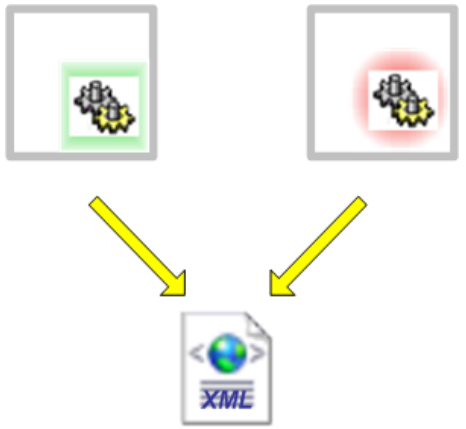


Prenosiva reprezentacija podataka

- Normirani format dokumenta
  - **neovisan** o aplikaciji, jeziku, platformi, načinu prijenosa i/ili arhiviranja
- Omogućuje **razmjenu** dokumenata između aplikacija kao međuformat
- Omogućuje **suradnju** aplikacija na istom dokumentu ili parcijalno uređivanje dijelova dokumenta



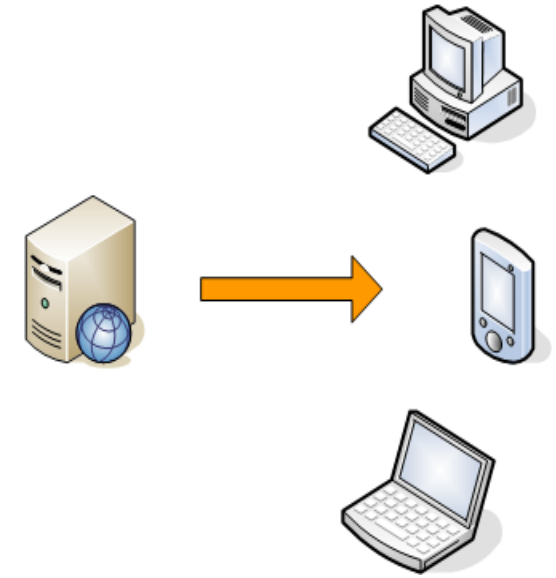
razmjena dokumenata



suradnja na istom dokumentu

Objavljivanje i prikaz dokumenta

- XML odvaja podatke i njihov prikaz
- Različiti prikazi XML-dokumenta:
  - po korisniku
  - po uređaju
  - po mediju
- Objavljivanje:
  - XML-dokumenta
    - **transformacija nakon** prijenosa na uređaj koji služi za prikaz sadržaja
  - dokumenta spremnog za prikaz
    - **transformacija prije** objave dokumenta ili prilikom dohvata (npr. transformacija XML → HTML+CSS)



Učitavanje XML-dokumenta u aplikaciju

- Parsiranje dokumenta
  - provjera dobre oblikovanosti (*well-formed*) dokumenta (obvezno)
  - provjera valjanosti (*valid*) dokumenta
    - validirajući parseri
  - dokument mora zadovoljiti prvu ili obje provjere
- Stvaranje unutarnje reprezentacije dokumenta
  - dvije vrste parsiranja:
    - Linijsko - SAX
      - ne stvara reprezentaciju
      - temeljen na događajima
    - Potpuno – DOM
      - stvara objektni model dokumenta
      - stvara stablastu strukturu

Uređivanje XML-dokumenta pomoću aplikacije

- Bez modela
  - stvaranje XML datoteke ispisivanjem teksta u datoteku korak po korak:
    - npr. korištenjem funkcije `fprintf()`
- Objektni model:
  - uređivanje sadržaja i strukture
  - stvaranje modela novog dokumenta
  - izvoz modela u formatu XML (datoteka, tok ...)

Stabla ili događaji?

- Korištenje XML-dokumenata kao strukture podataka u nekom programskom jeziku

- Dva osnovna načina rada s XML-dokumentima:
  - temeljen na **stablu**
    - potpuno parsiranje - DOM
    - mapira cijeli XML-dokument u objektnu/stablastu strukturu
  - temeljen na **događajima**
    - slijedno (linijsko) jednosmjerno parsiranje - SAX
    - parsiranje generira događaje koji se obrađuju u aplikaciji

Stabla ili događaji?

- Prednosti korištenja objektnog pristupa i stabla (DOM):
  - rukovanje čitavim XML-dokumentima tijekom obrade
  - mogućnost rada s objektnom strukturom stabla neovisnom o inicijalnom ulaznom XML-dokumentu

- Prednosti korištenja događaja i parsiranja (SAX):
  - nije potrebno stavljati cijeli XML-dokument u memoriju u obliku objektne strukture stabla
    - korisno kod velikih XML-dokumenata
  - kod potrebe za kreiranjem vlastitih memorijskih podatkovnih struktura
    - nepraktično sve prebaciti u DOM pa onda u novu strukturu

Document Object Model - DOM

- Samo podsjetnik – radili ste ga u predmetu WiM
  - Da, to je „isti taj” DOM, samo primijenjen na XML umjesto na HTML

- "*The W3C Document Object Model (DOM) is a **platform and language-neutral** interface that allows programs and scripts to dynamically **access** and **update** the content, structure, and style of a document.„*

http://www.w3.org/DOM/

XML DOM

- Objektni model XML-dokumenta
  - koristi objektnu paradigmu za reprezentaciju komponenata dokumenta
- Platformno i jezično neutralan
  - ne pogoduje specifičnom programskom jeziku ili operacijskom sustavu
  - postoje implementacije za važnije programske jezike
- DOM definira:
  - standardni **skup razreda** koji reprezentiraju tipove komponenata XML-dokumenta
  - standardne **funkcije** za rad s dokumentom
    - obilazak, pretraživanje, dodavanje, brisanje, mijenjanje vrijednosti ...

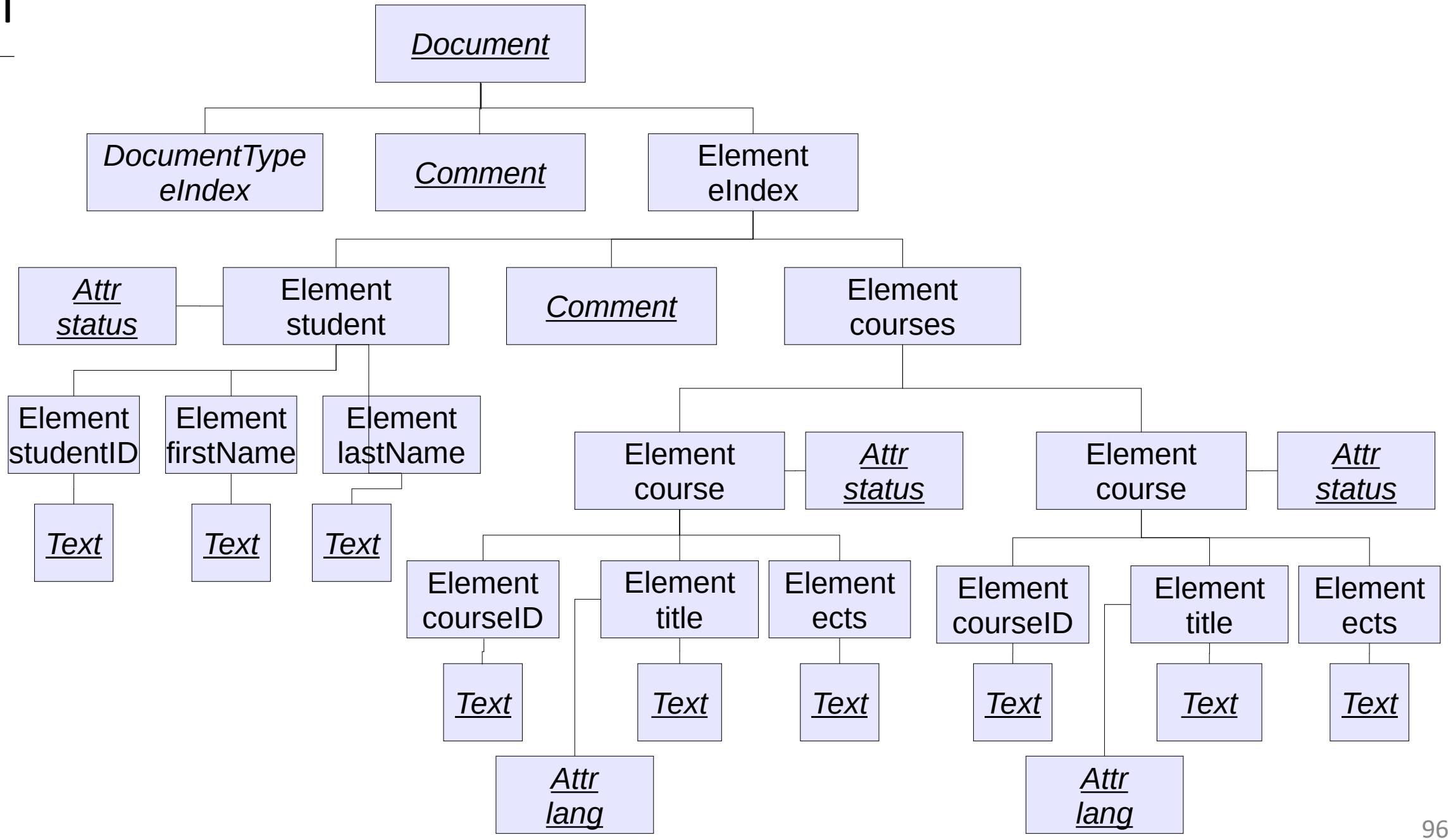
XML-dokument kao stablo

- Svaka komponenta je čvor stabla (tip *Node*)
- Komponente se kombiniraju u **uređeno stablo** koje reprezentira strukturu XML-dokumenta
- Između komponenata dokumenta vrijede odnosi istovjetni odnosima u uređenom stablu
- Odnosi čvorova i vrijednosti čvorova izraženi vrijednostima svojstava pojedinog čvora

Primjer XML-a i pripadajućeg DOM-stabla

```
<?xml version="1.0" ?>
<!DOCTYPE eindex SYSTEM "eIndex.dtd">
<!-- eIndex XML dokument -->
<eindex>
  <student status="exchange">
    <studentID>00362342433</studentID>
    <firstName>Ole Gunnar</firstName>
    <lastName>Solskjær</lastName>
    ...
  </student>
  <!-- popis predmeta -->
  <courses>
    <course status="module">
      <courseID>361</courseID>
  <title lang='hr'>Otvoreno računarstvo</title>
      <ects>4</ects>
      ...
    </course>
     <course status="orientation">
      <courseID>225</courseID>
      <title lang='hr'>Baze podataka</title>
      <ects>4</ects>
      ...
    </course>
  </courses>
</eindex>
```

DOM


Document
DocumentTypeeIndex
Comment
ElementeIndex
Attrstatus
Elementstudent
Comment
Elementcourses
ElementstudentID
ElementfirstName
ElementlastName
Text
Text
Text
Elementcourse
Attrstatus
Elementcourse
Attrstatus
ElementcourseID
Elementtitle
Elementects
ElementcourseID
Elementtitle
Elementects
Text
Text
Text
Text
Text
Text
Attrlang
Attrlang

Tipovi podataka specifični DOM-u

- Osnovni tip podatka: **Node**
- Tipovi podataka koji nasljeđuju Node:
  - komponente modela (Document, Element, Attr ...) nasljeđuju svojstva i metode
- NodeList
  - uređena lista elemenata tipa Node (poredak bitan)
  - elementu se pristupa preko pozicije u listi (indeksa)
  - koristi se za čuvanje liste elemenata djece
- NamedNodeMap
  - neuređen skup elemenata tipa Node
  - elementu se pristupa navođenjem njegova imena
  - koristi se za čuvanje liste atributa elementa

Svojstva tipa *Node*

strukturna

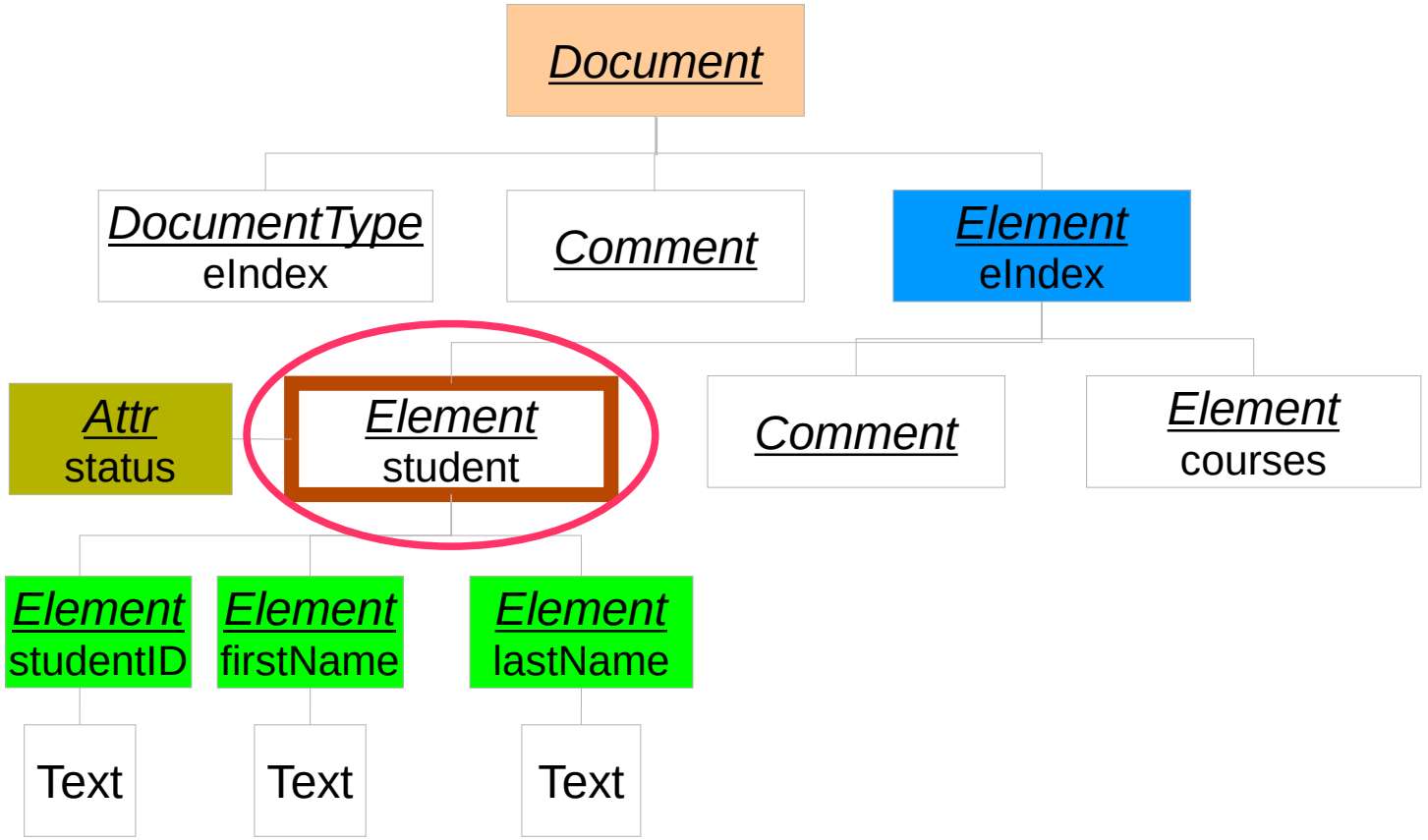
```
ownerDocument      : Document
parentNode         : Node
childNodes         : NodeList
firstChild         : Node
lastChild          : Node
previousSibling    : Node
nextSibling        : Node
attributes*        : NamedNodeMap
```

(- samo čvor Element)*

podatkovna

```
nodeName      : string
nodeType      : number
nodeValue     : string
```

Primjeri svojstava – čvor *student*



struktura

ownerDocument : *Document*

parentNode : eIndex

childNodes : *studentID, firstName, lastName*

firstChild : *studentID*

lastChild : lastName

previousSibling : -

nextSibling : *Comment*

attributes : *status*

podaci

nodeName : student

nodeType : 1

nodeValue : -

SAX

- SAX = Simple API for XML
  - aplikacijsko programsko sučelje parsera sa sekvencijalnim pristupom za XML
- Mehanizam čitanja XML-dokumenta
  - alternativa DOM-u
  - originalno izveden u programskom jeziku Java
    - jedan od prvih široko prihvaćenih API-ja u prog. jeziku Java
  - besplatan, u javnoj domeni, ne postoji norma
  - metode se pozivaju kada se tijekom parsiranja naiđe na:
    - jednostavan sadržaj (tekst)
    - element
    - naredbe obrade
    - komentar

Primjer rada SAX-a

```
<?xml version="1.0" ?>
   ...
<!-- eIndex XML dokument -->
<eindex>
  <student status="exchange">
<studentID>00362342433</studentID>
    <firstName>Ole Gunnar</firstName>
    <lastName>Solskjær</lastName>
    ...
  </student>
  <!-- popis predmeta -->
  <courses>
    <course status="module">
      <courseID>361</courseID>
      <title lang='hr'>Otvoreno
            računarstvo</title>
      <ects>4</ects>
      ...
    </course>
  </courses>
</eindex>
```

- Naišao na naredbu obrade ***xml*** s atributom ***version*** vrijednosti *"1.0"*
- Naišao na komentar
- Naišao na element ***eindex***
- Naišao na element ***student*** s atributom ***status*** vrijednosti *"exchange"*
- Naišao na element ***studentID***
- Naišao na sadržaj ***00362342433***
- .....

Kada koristiti SAX, a kada DOM?

- U načelu SAX je brži i koristi manje memorije
  - cijeli XML-dokument se ne učitava u memoriju
- SAX je jednostavniji i brži za pronalaženje i čitanje pojedinog podatka iz strukture
- DOM se koristi kod manipulacija sa cijelom strukturom (cijeli XML-dokument)
  - kod potrebe za objektnom reprezentacijom dokumenta DOM je prirodni odabir
- Ako se XML-dokument često ponovno koristi prikladno ga je imati u objektnom obliku – DOM
- Transformacije (XSLT) u načelu koriste DOM, jer im je potreban cijeli dokument
- **ALI U PRAKSI:**
  - **Danas je DOM standard, a SAX se koristi samo u iznimnim slučajevima.**

Super, otprilike znam
kako to interno radi...
Što sad?
Kada će podaci postati korisni?

Otvoreno računarstvo

3b. Otvorenost zapisa podataka

- Formati zapisa podataka, CSV
- **XML**
- JSON

XSL

- XSL ili e**X**tensible **S**tylesheet **L**anguage
- Skup preporuka za definiranje **transformacija** i **prezentacije** XML-dokumen(a)ta
- Sastoji se od 3 dijela:
  - **XPath**
    - jezik izraza za pristup i lociranje dijelova XML-dokumenta
  - **XSLT** (EXtensible Stylesheet Language Transformations)
    - jezik za transformacije
  - **XSL-FO** (XSL Formatting Objects)
    - rječnik za formatiranje XML-dokumenta za prezentaciju
- W3C o XSL: www.w3.org/Style/XSL

XPath

- XPath ili XML Path Language

- Pristup i lociranje (adresiranje) dijela XML-dokumenta
  - pronalaženje elemenata i atributa koji odgovaraju određenom kriteriju
  - traženje u bilo kojem smjeru (unaprijed, unatrag, u oba smjera)

- Npr. pronaći sva poglavlja knjige koja govore o XPath-u

Primjer XPath-a

```
<?xml version="1.0"?>
<knjiga>
   <autor>Marko</autor>
   <naslov>XPath</naslov>
    <cijena>1</cijena>
</knjiga>
```

ROOT - adresa "/"
Knjiga - adresa "/knjiga"
Autor - adresa "/knjiga/autor"
"**Marko**" - adresa "/knjiga/autor/text()"
Naslov - adresa "/knjiga/naslov"
"**XPath**" - adresa "/knjiga/naslov/text()"
Cijena - adresa "/knjiga/cijena"
"1" - adresa "/knjiga/cijena/text()"

Odabir čvorova

- Posebni znakovi

/	odabir od korijenskog čvora
//	odabir od trenutnog čvora bez obzira na položaj
.	odabir trenutnog čvora
..	odabir roditelja trenutnog čvora
@	odabir atributa
*	bilo koji element čvora
@*	bilo koji atribut čvora
|	odabir izraza prije ili iza ovog znaka

Primjer odabira čvorova

knjižnica	odabir sve djece elementa knjižnica
/knjižnica	odabir korijenskog elementa knjižnica
knjižnica/knjiga	odabir svih knjiga koji su djeca knjižnice
//knjiga	odabir svih knjiga bez obzira gdje su u dokumentu
knjižnica//knjiga	odabir svih elementa knjiga, bilo gdje, ali potomaka elementa knjižnica
//@jezik	odabir svih atributa jezik bilo gdje u dokumentu
/knjižnica/*	odabir sve djece elementa knjižnica
//*	odabir svih elemenata u dokumentu

Predikati

- Filtriranje - služe za odabir čvora sa specifičnom vrijednosti
- Uokvireni uglatim zagradama – [ ]
- Primjeri:
  - knjiga[1]
  - knjiga[last()]
  - knjiga[position()-1]
  - //naslov[@jezik='hr']

Trenutni kontekst

- Označava “gdje sam” trenutno
- Aktivni element u XPathu adresira korak
  - /root/…/predak/roditelj/ČVOR/dijete/potomak
- ČVOR je uvijek jednostruki čvor
  - može sadržavati samo jednog roditelja i jedan korijen
- Apsolutni put
  - /korak/korak/korak…
- Relativni put
  - korak/korak/korak…

Sintaksa koraka

- XPath lokacija puta sadrži jedan ili više koraka odijeljenim kosom crtom (“/”)
- Svaki korak sadrži:
  - os
    - relacija u stablu čvorova odabranog čvora u odnosu na trenutni kontekst (čvor)
  - čvor
    - čvor u odabranoj osi
  - predikat
    - filtar nad odabranim čvoro(vi)m(a)
- Sintaksa: **/os::čvor[predikat]/**

Primjeri korištenja osi

child::knjiga	Svi čvorovi tipa **knjiga** koji su djeca trenutnog čvora
attribute::jezik	Svi atributi tipa **jezik** trenutnog čvora
child::*	Sva djeca trenutnog čvora
child::text()	Svi tekstualni čvorovi djeca trenutnog čvora
child::node()	Svi čvorovi djeca trenutnog čvora
descendant::knjiga	Svi potomci tipa **knjiga** trenutnog čvora
ancestor::knjiga	Svi preci tipa **knjiga** trenutnog čvora
ancestor-or-self::knjiga	Svi preci tipa **knjiga** trenutnog čvora i on sam ako je **knjiga**
child::*/child::cijena	Svi unuci tipa **cijena** trenutnog čvora

XSLT

- E**X**tensible **S**tylesheet **L**anguage **T**ransformations (XSLT)
- Jezik temeljen na XML-u
- Služi za **transformacije XML-dokumenata** u druge XML-dokumente ili u neki drugi oblik
- Iz izvornog XML-dokumenta se na temelju transformacijskih pravila stvara novi dokument

XSLT - značajke

- Tipovi odredišnog dokumenta:
  - XML-dokument
  - HTML-dokument
  - čisti tekstualni dokument (*plain text*)
  - ...

- Mogućnosti:
  - dodavanje i prikaz elemenata i atributa
  - brisanje i sakrivanje nepoželjnih dijelova
  - uređivanje, oblikovanje i sortiranje podataka
  - testiranje i uvjetno izvršavanje pravila

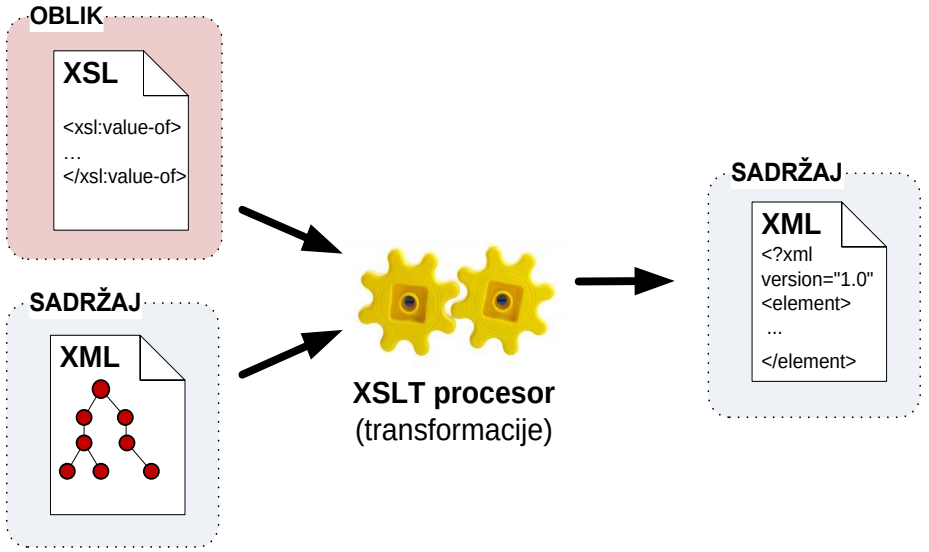
XSLT – model rada

- XSLT model uključuje:
  - XML izvorni dokument
  - XSLT predložak stila (*stylesheet*)
  - XSLT procesor (*processor, processing engine*)
  - odredišni dokument
    - rezultat transformacije

- XSLT **predložak stila** (*stylesheet*)
  - sadrži niz pravila i drugih naredbi pomoću kojih daje upute procesoru kako napraviti novi dokument
  - po strukturi to je također XML-dokument

XSLT – procesiranje

- Ulaz
  - XML izvorni dokument (npr. datoteka “.xml”)
  - XSLT predložak stila (npr. datoteka “.xsl”)
- Uzima se izvorni XML-dokument
  - Iz njega se gradi stablasta struktura
- Ponavlja se postupak:
  - putem XPath izraza se pronalaze odgovarajući čvorovi
  - nad pronađenim čvorovima se primjenjuju pravila
  - stvaraju se novi elementi (čvorovi) u odredišnom dokumentu
- Izlaz
  - proizvodi novi XML-dokument
    - može biti XML, HTML, XHTML …



XSLT procesor

- Može se nalaziti na:
  - poslužitelju (*server-side*)
  - klijentu (*client-side*)
- Može biti:
  - samostalna aplikacija
  - sadržan u pregledniku Weba
  - sadržan u aplikacijskom poslužitelju
  - sadržan u programskom okruženju (*framework*)
  - sadržan u operacijskom sustavu

Deklaracija predloška stila

- Korijenski element predloška stila
  - <xsl:stylesheet> ili <xsl:transform>
  - potrebna deklaracija inačice i prostora imena
    - primjer:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

- Spajanje izvornog XML-dokumenta i XSLT predloška stila
  - u izvornom XML-dokumentu povezivanje dolazi nakon deklaracije, a prije korijenskog dokumenta
    - primjer povezivanja:

```
<?xml-stylesheet type="text/xsl" href="predlozak.xsl"?>
```

  - otvaranjem spojenog XML-dokumenta s XSLT predloškom stila u pregledniku, transformacija će se izvršiti automatski
    - korištenjem XSLT procesora povezanog s preglednikom

Predlošci

- XSLT predložak stila (*stylesheet*) sadrži jedno ili više pravila, tzv. "**predložaka**" (*templates*)

- Svaki "predložak" sadrži pravila koja se izvršavaju kada se locira odgovarajući čvor

- "Predložak" koristi element <*xsl:template*> i atribut *match*
  - atribut *match* se koristi za pronalaženje XML podatka
    - upisuje se XPath izraz
    - najčešće korijenski čvor ("/")
  - vrijednost atributa *match* određuje pronalazak čvora

Osnovni elementi predloška stila

- Dohvat sadržaja elementa
  - <xsl:value-of>
  - <xsl:value-of select="knjižnica/knjiga/naslov">
- Pristup skupu elemenata
  - <xsl:for-each>

```
<xsl:for-each select="knjižnica/knjiga">
<xsl:for-each select="knjižnica/knjiga[autor='August Šenoa']">
```

- Uvjet
  - <xsl:if>

```
<xsl:if test="autor='August Šenoa'">
```

- Višestruki uvjet
  - <xsl:choose><xsl:choose>

```
<xsl:when test="autor='August Šenoa'"/>
```

- Poredak
  - <xsl:sort>

Primjer:

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

<knjižnica>
 <knjiga>
   <autor>Marko</autor>
   <naslov jezik="hr">
              Knjiga o XPathu
   </naslov>
    <cijena>1</cijena>
 </knjiga>
</knjižnica>
```

XSL-FO

- E**X**tensible **S**tylesheet **L**anguage **F**ormatting **O**bjects
  - XSL-FO
- Jezik temeljen na XML-u
- Služi za formatiranje XML-dokumenata u oblik **sa stranicama** koji se može ispisivati
  - dokument za ispis na papir
  - dokument za pregled na zaslonu u obliku stranica
- sadržaj se prelama po stranicama odredišnog dokumenta (prema XSL-FO pravilima)
- ne opisuje se definitivni izgled stranica, već relativni razmještaj područja s elementima sadržaja

XSL-FO – Struktura dokumenta

- Raspored stranice (*page layout*) definira značajke stranice
  - smjer toka teksta, veličinu stranice, margine, razlike parnih i neparnih stranica …
- Sadržajni dio sadrži sljedove tokova
  - svaki tok je pridružen rasporedu stranice
- Tok sadrži listu blokova koji sadrže tekstualne podatke i/ili oznake
- Blokovi se ponašaju slično kao kod CSS

XSL-FO Prednosti i nedostaci

- Prednosti:
  - jednostavnost transformacija korištenjem XSLT-a
  - sličnost s CSS-om
  - manji troškovi kod dobavljanja i održavanja
  - podrška za sve jezike
  - zrelost norme i manjak konkurencije
- Nedostaci:
  - ograničenja kod kompliciranog dizajna i tipografije
  - postoje bolji, komercijalni proizvodi
    - QuarkXPress, Adobe InDesign, MS Publishe...

Otvoreno računarstvo

3b. Otvorenost zapisa podataka

- Formati zapisa podataka, CSV
- XML
- **JSON**

Što nedostaje XML-u?

- „There are simply too many notes...” :-)

https://mygeekwisdom.com/2016/08/06/there-are-simply-too-many-notes/

Što nedostaje XML-u?

- Učinkovitost prijenosa podataka
  - dodaje dosta "nekorisnog" teksta na prenošene "korisne" podatke
  - parovi oznaka elemenata, imena atributa, instrukcije obrade, komentari ...

- Složenost rukovanja XML-om unutar programa
  - SAX – bez stvaranja strukture podataka
    - struktura nepotrebna ili sami stvaramo strukturu po želji
  - DOM – složena struktura stabla
    - parsiranje, poseban API za dohvat i manipulaciju stablom

- Tipovi podataka
  - nepostojeći (samostalni XML)
  - Za definiranje potrebni prejednostavni DTD ili presložena XML Schema (dodatni napor)

JSON

- JSON – **JavaScript Object Notation**
- Podskup jezika JavaScript
  - norme ECMA-404, ISO/IEC 21778:2017, RFC 8259
- Jednostavan, otvoren, platformno i jezično neovisan tekstni zapis podataka
- Zapis (polu-)strukturiranih podataka (kao i XML)
- Originalno puno manjih „apetita” od XML-a: samo **zapis podataka**

JSON – osnovne karakteristike

- **Razmjena** ili **zapis** jednostavnijih **struktura podataka**
- Mali broj predefiniranih tipova i struktura
- **Jednostavan** oblik zapisa strukture i podataka
  - jednostavnije parsiranje zapisa
  - jednostavnije mapiranje tipova i struktura u tipove i strukture ciljnog programskog jezika
- Format zapisa **UTF-8**
  - UTF-16 i UTF-32 dozvoljeni, ali ne i preporučeni
- MIME / Internet Media Type: **application/json**
- Nastavak imena datoteke: .json

JSON – osnovne karakteristike

- Naročito pogodan za **aplikacije weba**
  - asimetrija dostupnih resursa (klijent – **poslužitelj**)
- izravno mapiranje podataka i struktura JSON-a u **podatke** i **strukture JavaScripta**
  - JSON ⊂ JavaScript
- podrška ~~izvođenju~~ i **parsiranju** JSON zapisa na platformama koje podržavaju JavaScript (svi bitni preglednici weba)

JavaScript: JSON API

- Mogućnost izvođenja i parsiranja
  - ~~Izvođenje – funkcija eval()~~
  - **Problem sigurnosti** – injekcija JS koda u JSON datoteku

```
var myObject = eval('(' + myJSONtext + ')');
```

- Parsiranje – metoda *parse* objekta JSON
  - Siguran način (propušta samo **definiciju podataka**, ne i JavaScript programski kod)

```
var myObject = JSON.parse(myJSONtext);
```

- Serijalizacija – metoda *stringify* objekta JSON

```
var myJSONtext = JSON.stringify(myObject);
```

JSON – tipovi podataka

- JSON definira šest tipova podataka
- Osnovni tipovi
  - string
  - number
  - boolean
  - null
- Složeni tipovi
  - array
  - object

- Detalje tipova i sintaksu proučite sami!

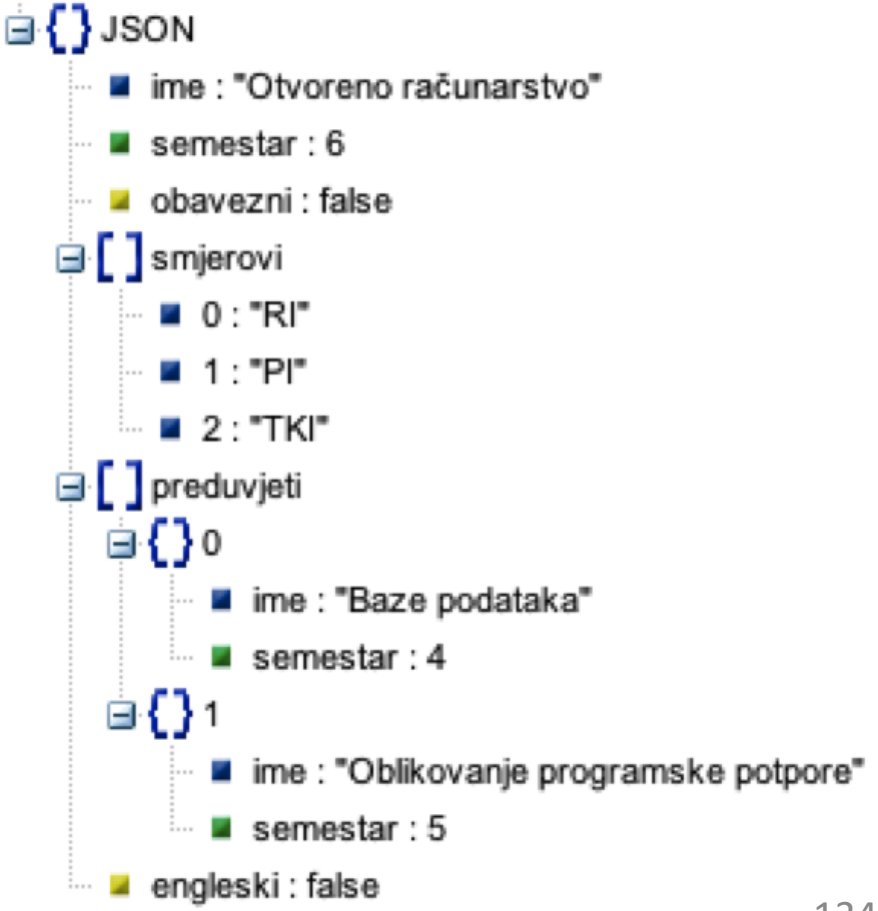
Primjer JSON-a – složeniji objekt

```
{
    "ime": "Otvoreno računarstvo",
    "semestar": 6,
    "obavezni": false,
    "smjerovi": [
        "RI",
        "PI",
        "TKI"
    ],
    "preduvjeti": [
        {
            "ime": "Baze podataka",
            "semestar": 4
        },
        {
            "ime": "Oblikovanje programske potpore",
            "semestar": 5
        }
    ],
    "engleski": false
}
```

Struktura JSON-dokumenta

- Valjani JSON-dokument
  - Objekt ili polje kao korijenski tip dokumenta (RFC 4627)
  - Bilo koji tip kao korijenski tip dokumenta (RFC 7159)

- JSON-dokument ima strukturu stabla (kao XML!)
  - osnovni tipovi: **listovi**
  - složeni tipovi: **grane**



JSON i XML

```
<predmet>
    <ime>Otvoreno računarstvo</ime>
    <semestar>6</semestar>
    <obvezni>false</obvezni>
    <smjerovi>
        <smjer>RI</smjer>
        <smjer>PI</smjer>
        <smjer>TKI</smjer>
    </smjerovi>
    <preduvjeti>
        <preduvjet>
            <ime>Baza podataka</ime>
            <semestar>4</semestar>
        </preduvjet>
        <preduvjet>
            <ime>Oblikovanje programske potpore</ime>
            <semestar>5</semestar>
        </preduvjet>
    </preduvjeti>
    <engleski>false</engleski>
</predmet>
```

Dobra oblikovanost i valjanost

- Dobru oblikovanost provjerava JSON parser
  - sintaksa
  - struktura složenih tipova podataka

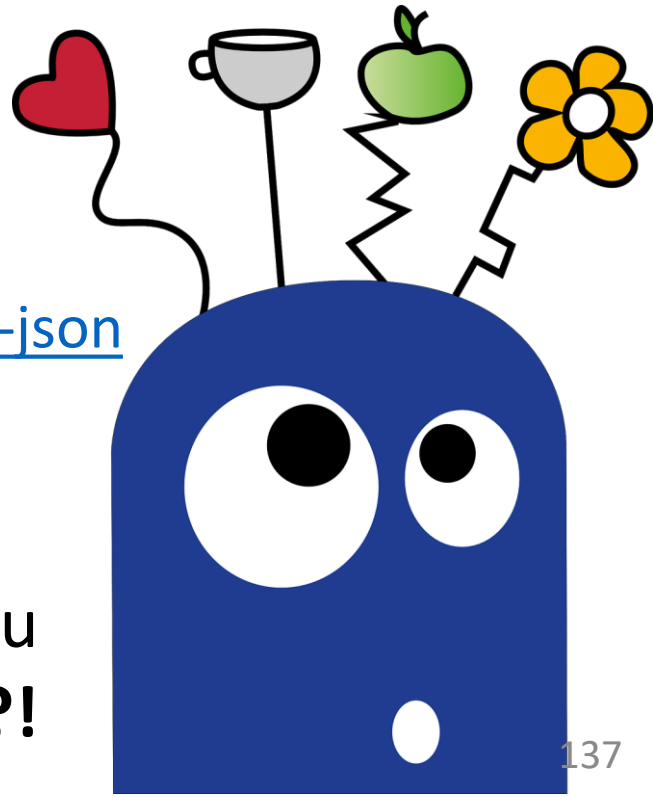
- JSON još nema normirane definicije sheme
  - **JSON Schema -** u *draft* fazi razvoja

- Valjanost mora provjeravati neka vanjska aplikacija
  - definiranje valjane strukture
  - definiranje valjanih vrijednosti

JSON „dodaci”

- Pokušaji (*više ili manje uspješni*) da JSON, izvorno zamišljen kao jednostavni zapis za razmjenu podataka, postane cjelovit „ekosustav” :-), kao što je XML

- JSON Schema
- JSONPath, JSPath, JSONata, jmespath, json-query, JSONiq...
  - https://www.npmtrends.com/JSONPath-vs-jmespath-vs-json-query-vs-jsonata-vs-jspath-vs-jsoniq
  - Ako vam nije dosta: https://stackoverflow.com/questions/1618038/xslt-equivalent-for-json
- JSON Pointer



Netko je na ovom predmetu spominjao **NORME?!**

(manje poznati) Nedostaci JSON-a

- Mali broj definiranih tipova -> nestandardna rješenja za kodiranje ostalih tipova
  - Primjer: zapis datuma i vremena!
  - Pametno koristiti neku normu, ISO 8601
  - Vraća ga i metoda *ToJSON* za objekt *Date*
- Ugnježđivanje kraćih binarnih zapisa
  - string + base64
- Podrazumijevana preciznost brojeva
  - Podržava li ciljna aplikacija IEEE 754?
- Korištenje \u0000 (NULL) u definiciji stringa
  - Kako će se ponašati ciljna aplikacija? Java, JavaScript? C?

XML – JSON mapiranja

- Ideja #1: Pragmatični pristup
  https://www.xml.com/pub/a/2006/05/31/converting-between-xml-and-json.html

	XML	JSON	Access
1	<e/>	"e": null	o.e
2	<e>text</e>	"e": "text"	o.e
3	<e name="value" />	"e":{"@name": "value"}	o.e["@name"]
4	<e name="value">text</e>	"e": { "@name": "value", "#text": "text" }	o.e["@name"] o.e["#text"]
5	<e> <a>text text </e>	"e": { "a": "text", "b": "text" }	o.e.a o.e.b
6	<e> <a>text <a>text </e>	"e": { "a": ["text", "text"] }	o.e.a[0] o.e.a[1]
7	<e> text <a>text </e>	"e": { "#text": "text", "a": "text" }	o.e["#text"] o.e.a

XML – JSON mapiranja

- Ideja #2: XSLT 3.0 pristup
  - https://www.w3.org/TR/xslt-30/#json-to-xml-mapping

```
{
  "desc"    : "Distances... ",
  "updated" : "2014-02-04T18:50:45",
  "uptodate": true,
  "author"  : null,
  "cities"  : {
    "Brussels": [
      {"to": "London",    "distance": 322},
      {"to": "Paris",     "distance": 265},
      {"to": "Amsterdam", "distance": 173}
    ]
  }
}
```

```
<map xmlns="http://www.w3.org/2005/xpath-functions">
    <string key='desc'>Distances...</string>
    <string key='updated'>2014-02-
04T18:50:45</string>
    <boolean key="uptodate">true</boolean>
    <null key="author"/>
    <map key='cities'>
      <array key="Brussels">
        <map>
            <string key="to">London</string>
            <number key="distance">322</number>
        </map>
        <map>
            <string key="to">Paris</string>
            <number key="distance">265</number>
        </map>
        <map>
            <string key="to">Amsterdam</string>
            <number key="distance">173</number>
        </map>
      </array>
    </map>
</map>
```

JSON vs XML

JSON Schema

- JSON Schema: A Media Type for Describing JSON Documents
- Namijenjena **opisivanju** i **validiranju** JSON-dokumenata
- Trenutačno stanje donošenja norme:
  - IETF draft
  - nastavlja se izrada vlastitih nacrta (*Internet Draft,* trenutačno 2020-12)
    - **<- žele se odmaknuti od IETF-a, trenutne rasprave**
  - **VRLO VJEROJATNE IZMJENE** SPECIFIKACIJE!
- Po svojoj strukturi je i sama JSON-dokument
- 4 dijela specifikacije:
  - **JSON Schema (Core)** – media type za opis dokumenta
  - **JSON Schema Validation** – rječnik za validaciju strukture dokumenta
  - **JSON Hyper-Schema** – rječnik za hipermedijske anotacije u dokumentu
  - **Relative JSON Pointers** – specificiranje lokacija u dokumentu

Početak dokumenta JSON Schema

- Deklaracija na početku:

```
{ "$schema": "http://json-schema.org/schema#" }
```

- Jedinstveni identifikator scheme s URI-referencom:

```
{ "$id": "http://moj_website.hr/schemas/address.json" }
```

  - U kombinaciji s $ref može služiti i kao pokazivač na druge podsheme

JSON Schema i tipovi podataka

- Ključne riječi:
  - *type, properties* (za objekte), *items* (za polja)
- Svaki tip podatka ima svoja dodatna ograničenja
  - raspon, duljina, regularni izrazi...
- Primjeri:

```
{ "type": "number" }                    <- samo brojevi
{ "type": ["number", "string"] }        <- brojevi i stringovi

{
  "type": "object",
  "properties": {
    „broj":      { "type": "number" },
    „naziv_ulice": { "type": "string" },
    „tip_ulice": { "type": "string",
          "enum": [„Ulica", „Avenija",„Trg"]
                 }
               }
 }
```

```
{
  "type": "array",
  "items": [
    {
      "type": "number"
    },
    {
      "type": "string"
    },
    {
      "type": "string",
      "enum": [„Ulica", „Avenija", „Trg"]
    }
  ]
}
```

JSON i anotacije - metapodaci

- Ključne riječi za opis podataka:
  - title – kratak naziv
  - description – dulji opis
  - default – zadana vrijednost
  - examples – polje primjera koji bi prolazili validaciju

```
{
  "$id": "https://moj_website.com/osoba.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": „Osoba",
  "type": "object",
  "properties": {
    "firstName": {
      "type": "string",
      "description": „Ime osobe"
    },
    "lastName": {
      "type": "string",
      "description": „Prezime osobe"
    },
    "age": {
      "type": "integer",
      "description": „Broj godina osobe",
      "minimum": 0
    }
  }
}
```

JSON Schema – implementacije, alati i biblioteke

- https://json-schema.org/implementations.html
  - Po pojedinim verzijama nacrta (*drafts*)
  - Vrlo promjenjiv popis
- Validatori
- Generatori schema
  - Iz kôda
  - Iz podataka
- Generatori web-obrazaca za slanje podataka
- Konverteri
- Alati za testiranje
- Uređivači JSON i JSON Schema datoteka

Otvoreno računarstvo

3b. Otvorenost zapisa podataka

- Formati zapisa podataka, CSV
- XML
- JSON