

# Uvod u veb i internet tehnologije





# Jezici za obeležavanje





# Načini rada sa tekstualnim dokumentima



# Rad sa tekstualnim dokumentima

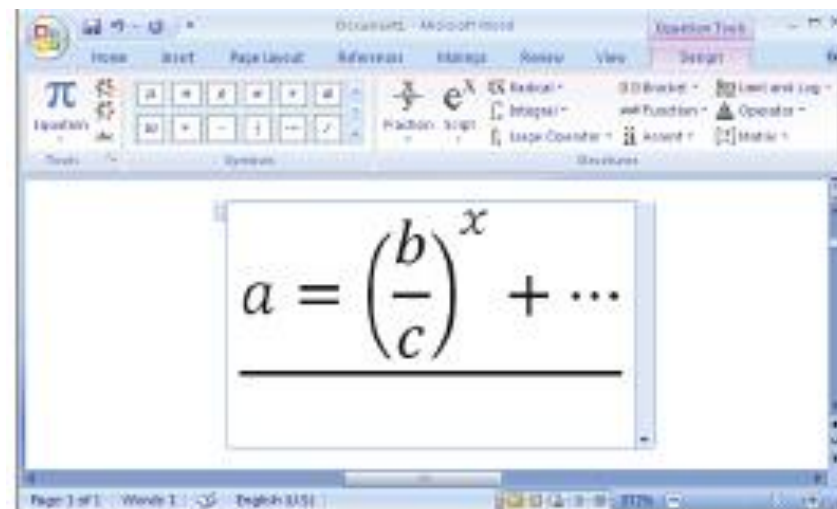
- U današnjem dobu računara, izdvajaju se dva paradigmatična pristupa za kreiranje tekstualnih dokumenata
  - WYSIWYG (What You See Is What You Get) pristup
  - korišćenje jezika za obeležavanje
- U nastavku će ukratko biti opisana oba pristupa, a potom će (zbog mnogobrojnih prednosti koje ovaj pristup donosi) naglasak biti stavljen na eksplicitno obeležavanje teksta korišćenjem jezika za obeležavanje



## Rad sa tekstualnim dokumentima (2)

### ○ WYSIWYG pristup

- Alati zasnovani na WYSIWYG pristupu zahtevaju od korisnika da tekst uredi u obliku koji je spreman za konačno prikazivanje na ciljnom medijumu (npr. štampanje na papiru)
- Tekst se uređuje oslanjajući se direktno na njegovu grafičku prezentaciju, najčešće korišćenjem miša i elemenata grafičkog korisničkog okruženja
- Tipični primeri ovakvih alata su alati za kancelarijsko poslovanje (npr. Microsoft Office, OpenOffice.org)







## Rad sa tekstualnim dokumentima (3)

### ○ Pristup eksplicitnim obeležavanjem teksta

- Tehnika eksplicitnog obeležavanja strukture dokumenata olakšava njihovu automatsku obradu
- Obeleženi dokumenti postaju uskladištene informacije koje je moguće automatski obrađivati korišćenjem raznovrsnim računarskih aplikacija, ali i prikazivati u obliku pogodnom za čitanje od strane čoveka
- Ovaj pristup dobija na značaju kada se izvrši jasno i eksplicitno razdvajanje obeležavanja logičke strukture i obeležavanja vizuelne prezentacije dokumenta
- **Logička struktura** dokumenta podrazumava njegovu organizaciju na manje jedinice (npr. poglavlja, sekcije, pasuse), kao i označavanje njegovih istaknutih delova (npr. primeri, citati, definicije i teoreme)
- **Vizuelna prezentacija** određuje izgled dokumenta u trenutku prikazivanja ili štampanja

```
\documentclass[a4paper]{article}
\begin{document}

$$a = \left(\frac{b}{c}\right)^x + \ldots$$

\end{document}
```



## Rad sa tekstualnim dokumentima (4)

- Pristup eksplicitnim obeležavanjem teksta
  - Razdvajanje logičke strukture dokumenata od njihove grafičke prezentacije daje mogućnost da se uz minimalan trud istim podacima pridruže sasvim različiti vizuelni prikazi
  - Prilikom eksplicitnog obeležavanja teksta, koriste se jezici za obeležavanje teksta (markup languages)
    - To su veštački jezici u kojima se korišćenjem posebnih oznaka opisuje logička struktura teksta ili njegov grafički izgled.
    - Najpoznatiji jezici za obeležavanje su HTML, Tex tj. LaTeX, PostScript, RTF, itd.
    - Svaki od ovih jezika odlikuje se konkretnom sintaksom označavanja i koristi se za označavanje jednog tipa dokumenta (npr. HTML se koristi za označavanje hipertekstualnih dokumenata)
  - U praksi se često javlja potreba za označavanjem velikog broja različitih tipova dokumenata (npr. označavanje pisama, tehničkih izveštaja, zbirki pesama, itd.)



## Rad sa tekstualnim dokumentima (5)

- Pristup eksplicitnim obeležavanjem teksta
  - Jasno je da svaki pojedinačni tip dokumenata zahteva svoj način označavanja i skup oznaka pogodnih za njegovo označavanje
  - Ovo dalje omogućava izradu specifičnih softverskih alata pogodnih za određenu vrstu obrade specifičnih tipova dokumenata
  - Kako bi se na precizan i uniforman način omogućilo definisanje konkretnih jezika za označavanje različitih tipova dokumenata, razvijeni su i meta jezici
  - Najpoznatiji meta jezici za obeležavanje su SGML i XML, u čijem okviru su definisani jezici HTML, XHTML, MathML, SVG itd.





# SGML





# Karakteristike i istorijat SGML

- Standardni opšti jezik za obeležavanje (Standard Generalized Markup Language) je meta jezik za obeležavanje standardizovan od strane medunarodne organizacije za standarde (pod oznakom „ISO 8879:1986 SGML”)
- Jezik je razvijen za potrebe kreiranja mašinski čitljivih dokumenata u velikim projektima industrije, državne uprave, vojske itd.
- Osnovna motivacije prilikom standardizovanja ovog jezika je bila da se obezbedi trajnost dokumentima i njihova nezavisnost od aplikacija kojima su kreirani
- Informacije skladištene u okviru SGML dokumenta su nezavisne od platforme tj. od softvera i hardvera
- Pretečom jezika SGML smatra se jezik GML (Generalized Markup Language) nastao u kompaniji IBM 1960-tih



## Karakteristike i istorijat SGML (2)

- Jedna od značajnijih primena jezika SGML je bila izrada drugog, elektronskog, izdanja Oksfordskog rečnika engleskog jezika (OED)

```
Document: Bungler OED           At: "<entry>"

<entry>
  <hwsec>
    <hwgp>
      <hwlem>bungler</hwlem>
      <pron>b<1>ɒ</1>ˈŋɡl̩ə</pron>. </hwgp>
      <vfl>Also <vd>b</vd> <vf>bongler</vf>,
        </vfl>
      <etym>f. as prec. + <xra><xlem>-ER</xlem>
    </hwsec>
  <sen>One who bungles; a clumsy unskilful
    <quot>
      <qdat>1533 </qdat>
      <auth>MORE </auth>
      <wk>Answ. Poyson. Bk. </wk>Wks. (1557)
      <qtxt>He is even but a very bungler.
```

Fragment oksfordskog  
rečnika obeležen SMGL  
elementima



## Karakteristike i istorijat SGML (3)

- Može se reći da je najznačajnija primena jezika SGML došla kroz jezik HTML, čije su prve verzije definisane upravo u okviru jezika SGML
- Jezik HTML služi za obeležavanje hipertekstualnih dokumenata i postao je standardni jezik za obeležavanje dokumenata na vebu
- Svaki jezik za obeležavanje koji je definisan u SGML-u naziva se i SGML aplikacija, pa se i jezik HTML smatra SGML aplikacijom
- SGML se koristi da bi se obeležila struktura dokumenata određenog tipa



# Ilustracije korišćenja SGML

- Primer: Zbirka pesama sadri nekoliko pesama, pri čemu se svaka pesma sastoji od nekoliko strofa, a svaka strofa od nekoliko stihova
  - SGML uvodi oznake kojima se obeležavaju elementi dokumenta

```
<!DOCTYPE zbirka SYSTEM "zbirka-pesama.dtd">
<zbirka>
  <pesma autor="Čika Jova Zmaj">
    <strofa>
      <naslov>Žaba čita novine</naslov>
      <stih>Sedi žaba sama
      <stih>na listu lokvanja.
      <stih>Od žarkoga sunca
      <stih>štitom se zaklanja.
    </strofa>
  </pesma>
</zbirka>
```



## Ilustracije korišćenja SGML (2)

- Primer: Jedan jednostavni HTML dokument

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Moj prvi HTML dokument</title>
  </head>
  <body>
    <p>Zdravo svete! 
    <p>Copyright (&copy;) Milena
  </body>
</html>
```

- U oba prethodna primera, sadržaj dokumenta je obeležen oznakama koje određuju njegovu strukturu





# Struktura SGML

- Dokumenti se sastoje od međusobno ugnježenih **elemenata**
  - Za obeležavanje elemenata se koriste **etikete** (tagovi) oblika **<ime-elementa>** i **</ime-elementa>** (na primer <strofa> i </strofa> ili <body> i </body>)
  - Elementi sadrže tekst, druge elemente ili kombinaciju i jednog i drugog
- Elementi mogu biti dodatno okarakterisani **atributima**
  - Atributi su oblika **ime-atributa="vrednostatributa"** (na primer naslov= "Žaba čita novine")
- U okviru teksta mogu se pojaviti i znakovni entiteti
  - Oni su oblika **&ime-entiteta;** (na primer &copy;) koji označavaju određene znakove



## Struktura SGML (2)

- Sadržaj i značenje elemenata nije propisano meta jezikom, već svaki jezik definisan u okviru SGML-a definiše sopstveni skup etiketa koje koristi za obeležavanje i definiše njihovo značenje i moguće međusobne odnose
- Svakom dokumentu, pridružen je njegov tip
- Tip dokumenta određuje sintaksu dokumenta tj. određuje koji elementi, atributi i entiteti se mogu javiti u okviru dokumenta i kakav je njihov međusobni odnos
- Posebni programi koji se nazivaju SGML parseri ili SGML validatori mogu da ispituju da li je dokument u skladu sa svojim tipom tj. da li zadovoljava sva sintaksna pravila propisana odgovarajućim tipom



## Struktura SGML (3)

- Pripadnost određenom tipu dokumenta, izražava se deklaracijom `<!DOCTYPE>` koja se navodi na početku samog dokumenta
  - U okviru ove deklaracije se nalaze informacije o imenu tipa dokumenta, organizaciji koja ga je kreirala i sl.
  - Obično se u okviru ove deklaracije nalazi uputnica na definiciju tipa dokumenta (Document type definition - DTD)
  - Ove datoteke definišu elemente od kojih se grade konkretni dokumenti
  - U prvom primeru tip dokumenta je definisan datotekom `zbirka-pesama.dtd`
  - U drugom primeru tip dokumenta je definisan datotekom <http://www.w3.org/TR/html4/strict.dtd>
    - Oznaka `PUBLIC` u drugom primeru ukazuje na to da je tip dokumenta javan i dostupan



## Struktura SGML (4)

### ● Primer:

- Tip dokumenta zbirke pesama uvodi elemente **zbirka**, **pesma**, **strofa** i **stih** i zahteva da se zbirka sastoji od nekoliko pesama, da se svaka pesma sastoji od nekoliko strofa, a da se svaka strofa sastoji od nekoliko stihova
- U okviru ove definicije tipa dokumenta, specificirano je da pesma ima atribut **autor** kao i šta sve može biti vrednost ovog atributa

```
<!ELEMENT zbirka - - (pesma+)>
<!ELEMENT pesma - - (naslov?, strofa+)>
<!ATTLIST pesma autor CDATA #REQUIRED>
<!ELEMENT naslov - 0 (#PCDATA)>
<!ELEMENT strofa - 0 (stih+)>
<!ELEMENT stih 0 0 (#PCDATA)>
```

- Dakle, korišćenje SGML-a podrazumeva kreiranje sopstvenih ili korišćenje javnih tipova dokumenata i obeležavanje dokumenata u skladu sa njihovim željenim tipom



## Struktura SGML (5)

- Proces kreiranja novih tipova dokumenata podrazumeva izradu
  - **SGML deklaracije** - formalnog opisa leksike samih dokumenata koja prevashodno određuje koji znaci se koriste prilikom kreiranja dokumenata
  - **Definicije tipa dokumenta** - formalnog opisa sintakse samih dokumenata koja određuje od kojih elemenata, etiketa, atributa i entiteta se dokument sastoji i kakav je njihov međusobni odnos
  - **Semantičke specifikacije** - neformalnog opisa semantike elemenata, etiketa i atributa koji se koriste u okviru dokumenata
    - Ovakva specifikacija može u sebi da sadrži i neka dodatna ograničenja koja se ne mogu izraziti u okviru formalne definicije tipa dokumenta



# Osnovni konstrukti SGML

## ● Elementi i etikete

- Osnovna gradivna jedinica SGML dokumenata su **elementi**
- Elementi su obično označeni **etiketama** (tag). Razlikuju se otvarajuće etikete koje označavaju početak elementa i koje su oblika **<ime-elementa>** i zatvarajuće etikete koje označavaju kraj elementa i koje su oblika **</ime-elementa>**
- Treba istaći da elementi nisu isto što i etikete
  - Element sačinjava početna etiketa, završna etiketa i sav sadržaj (tekst i drugi elementi) koji se nalaze između njih
- Ime elementa se navodi i početnoj etiketi i u završnoj etiketi
- Imena elemenata dozvoljeno je pisati i malim i velikim slovima i ne pravi se razlika između velikih i malih slova





# Osnovni konstrukti SGML (2)

## ● Elementi i etikete

- Primer: element **ul** jezika (tipa dokumenta) HTML, služi da označi neku listu nabrojanih stavki, i njegov sadržaj čine tri elementa **li**, čiji su sadržaji niske Lista 1, Lista 2 i Lista 3

```
<ul>
<LI>Lista 1</LI>
<li>Lista 2</li>
<li>Lista 3</li>
</ul>
```

- Kod nekih SGML elemenata moguće je izostaviti završne etikete, dok je kod nekih čak moguće izostaviti i početne etikete
- Primer: u jeziku HTML, elementi **p** služe da označe pasuse. Pasusi ne zahtevaju navođenje završne etikete **</p>**. Početak novog pasusa **<p>** implicitno označava kraj prethodnog, slično kao i oznaka kraja obuhvatajućeg elementa **</body>**

```
<body>
  <p>Zdravo svete!
  <p>Copyright (&copy;) Milena
</body>
```



# Osnovni konstrukti SGML (3)

- Elementi i etikete
  - Neki SGML elementi nemaju svoj sadržaj
  - Primer: HTML element koji označava prelazak u novi red **br**
  - Kod praznih elemenata najčešće je zabranjeno navoditi završnu etiketu



# Osnovni konstrukti SGML (4)

## ● Atributi

- Atributi sadrže dodatne informacije o SGML elementima
- Atributi imaju svoj naziv i vrednost
  - Naziv atributa je razdvojen od vrednosti znakom jednakosti
  - Vrednost atributa treba biti navedena u okviru navodnika ("" ) ili apostofa (")
  - U okviru navodnika moguće je korišćenje apostofa i obratno
  - Ponekad navodnici i/ili apostofi, kod vrednosti atributa, mogu biti izostavljeni
  - Atributi elementa se navode u okviru njegove početne etikete
- Primer: atribut href elementa a jezika HTML određuje odredište hiperveze

```
<a href="http://www.google.com">Link na google</a>
```



# Osnovni konstrukti SGML (5)

## ● Atributi

- Atributi sadrže dodatne informacije o SGML elementima
- Atributi imaju svoj naziv i vrednost
  - Naziv atributa je razdvojen od vrednosti znakom jednakosti
  - Vrednost atributa treba biti navedena u okviru navodnika (") ili apostofa (')
  - U okviru navodnika moguće je korišćenje apostofa i obratno
  - Ponekad navodnici i/ili apostofi, kod vrednosti atributa, mogu biti izostavljeni
  - Atributi elementa se navode u okviru njegove početne etikete
- Primer: atribut href elementa a jezika HTML određuje odredište hiperveze

```
<a href="http://www.google.com">Link na google</a>
```
- Imena atributa su nezavisna od veličine slova, dok vrednosti nekada zavise, a nekada ne zavise od veličine slova



# Osnovni konstrukti SGML (6)

## ● Entiteti

- SGML daje mogućnost imenovanja delova sadržaja na portabilan način
- Koncept entiteta u SGML uvodi izvesnu vrstu makro zamena
- Zamena entiteta se vrši kada se dokumenti analiziraju odgovarajućim parserom
- Primer: moguće je deklarirati entitet pod imenom **uvit** koji se zamenjuje tekстом **Uvod u Veb i Internet tehnologije**, i zatim se u okviru ovog dokumenta na ime predmeta pozivati korišćenjem reference na entitet
- Postoji nekoliko vrsta entiteta i referenci na entitete:
  1. obični entiteti (regular entities)
  2. parametarski entiteti (parameter entities)
  3. znakovni entiteti (character entities)



# Osnovni konstrukti SGML (7)

## ● Entiteti

### 1. Obični entiteti

- Reference na obične entitete počinju sa znakom **&** i završavaju se sa **;**
- Moguće ih je navoditi u okviru teksta dokumenta (ne u okviru DTD)
- Primer: ako se negde u okviru dokumenta javi sadržaj

Nastava iz predmeta "&uvit;" se odvija utorkom.

ovim je u stvari kodiran tekst

Nastava iz predmeta "Uvod u Veb i Internet tehnologije" se odvija utorkom.

### 2. Parametarski entiteti

- Reference na parametarske entitete počinju znakom **%** i završavaju se sa **;**
- Moguće ih je navoditi samo u okviru DTD dokumenta (ne u okviru objektnih dokumenata)





# Osnovni konstrukti SGML (8)

## ● Entiteti

### 3. Znakovni entiteti

- Njima se uvode imena koja označavaju određene znakove
- Koriste se da bi se naveli znakovi koji imaju specijalno značenje, zatim neki retko korišćeni znakovi, znakovi koji nisu podržani tekućim kodiranjem ili znakovi koje je nemoguće uneti u okviru softvera za kreiranje dokumenata
- Primer: u jeziku HTML "&lt;" označava znak `<`, dok "&quot;" označava znak `"`
- Pored referenci na znakovne entitete, za predstavljanje znakova u dokumentima je moguće koristiti i numeričke znakovne reference
- One se navode kao brojevi (dekadni ili heksadekadni) zapisani između `&#` i `;`
- Obično ove vrednosti odgovaraju ISO 10646, tj. UNICODE-u
- Heksadekadni kodovi počinju sa x ili X



# Osnovni konstrukti SGML (9)

## ● Komentari

- U okviru SGML dokumenata moguće je navoditi i komentare, i to na sledeći način:

```
<!-- Ovo je jedan komentar -->  
<!-- Ovo je komentar,  
      koji ne staje u jedan red -->
```

## ● Označene sekcije

- Označene sekcije (marked sections) se koriste da bi se označili delovi dokumenta koji zahtevaju posebnu vrstu procesiranja
- One su sledećeg oblika:

```
<![ ključna reč [ ... označena sekcija ... ]]>
```

- Najčešće korišćene ključne reči su:

- CDATA - označava doslovan sadržaj koji se ne parsira
- IGNORE - označava da se sekcija ignoriše tokom parsiranja
- INCLUDE - označava da se sekcija uključuje tokom parsiranja
- TEMP - označava da je sekcija privremeni deo dokumenta



# Osnovni konstrukti SGML (10)

- Instrukcije procesiranja
  - Instrukcije procesiranja (processing instructions) su lokalne instrukcije aplikaciji koja obrađuje dokument
  - One su napisane na način specifičan za aplikaciju
  - Navode se između `<? i ?>`
  - Primer: u delu HTML dokumenta

```
<p>Sada je <?php echo date("h:i:s"); ?> </p>
```

instrukcija `<?php echo date("h:i:s"); ?>` govori PHP interpreteru koji obrađuje dokument da je u pitanju deo PHP koda koji je onda potrebno interpretirati



# Definicije tipa dokumenta

Svaki element i atribut u okviru neke SGML aplikacije se definiše u okviru definicije tipa dokumenta (DTD)

- Deklaracije entiteta

- Entiteti se deklariraju korišćenjem `<!ENTITY` za kojim sledi ime entiteta, vrednost entiteta pod navodnicima i završni znak `>`

- Primer: Ovim je deklarisan entitet

```
<!ENTITY uvit "Uvod u Veb i Internet tehnologije">
```

- U slučaju parametarskih entiteta, koristi se oznaka `%`

- Primer: Ovim je deklarisan parametarski entitet

```
<!ENTITY % fontstyle "TT | I | B | BIG | SMALL">
```

- Već deklarisan entitet može učestvovati u deklaraciji drugih entiteta

- Primer: Pethodno deklarisan entitet se dalje koristi u okviru DTD za deklaraciju drugih entiteta

```
<!ENTITY % inline  
    "#PCDATA | %fontstyle; | %phrase; | %special; | %formctrl;">
```



## Definicije tipa dokumenta (2)

### ● Deklaracije elemenata

- Većina DTD se sastoji od deklaracija elemenata i njihovih atributa
- Deklaracija elementa počinje sa `<!ELEMENT`, završava se sa `>`, a između se navodi:

1. Ime elementa
2. Pravila minimalizacije, koja određuju da li je neka od etiketa opcionala
  - Dve crtice - nakon imena označavaju da su obe etikete obavezne
  - Crtica - za kojom sledi O označava da se završna etiketa može izostaviti
  - O- za kojom sledi crtica označava da se početna etiketa može izostaviti
  - Dva slova O označavaju da se obe etikete mogu izostaviti
3. Sadržaj elementa.

Dozvoljeni sadržaj elementa se naziva model sadržaja (content model)

Za definiciju modela sadržaja koriste se:

- prosti modeli sadržaja
- složeni modeli sadržaja



## Definicije tipa dokumenta (3)

- Deklaracije elemenata

- prosti modeli sadržaja

- EMPTY - elementi koji nemaju sadržaj, tj. prazni elementi
    - ANY - element može imati proizvoljan sadržaj koji se sastoji od teksta i drugih elemenata
    - CDATA (character data) – sadržaj koji se neće analizirati pomoću SGML parsera  
Sadržaj se tumači doslovno kako je napisan tj. reference na entitete se ne zamenjuju entitetima, a etikete koje se u njemu nalaze ne označavaju elemente.
    - RCDATA (replacable character data) - slično kao CDATA, osim što se reference zamenjuju (etikete i dalje ne označavaju elemente)

- složeni modeli sadržaja - koriste se u slučaju kada element može da sadrži druge uneždene elemente

- Modeli grupe su predstavljeni izrazima u zagradama





# Definicije tipa dokumenta (4)

## ● Deklaracije elemenata

### ○ Atomi u izrazima modela grupe za složeni modeli sadržaja su:

- imena elemenata - označavaju uneždene elemente
- #PCDATA (parsed character data) - tekst koji će se analizirati pomoću parsera

Reference na entitete se u okviru ovog teksta se zamenjuju entitetima i etikete koje se u njemu nalaze označavaju elemente

### ○ Ovi atomi se dalje mogu kombinovati sledećim veznicima

- A? - atom A se može, ali ne mora pojaviti
- A+ - atom A se mora pojaviti jedan ili više puta
- A\* - atom A se mora pojaviti nula ili više puta
- A | B - ili atom A ili atom B se mora pojaviti, ali ne oba
- A, B - oba atoma A i B se moraju pojaviti u tom redosledu
- A & B - oba atoma A i B se moraju pojaviti u bilo kom redosledu



# Definicije tipa dokumenta (4)

## ● Deklaracije elemenata

- Moguće je definisati dodatna pravila uključivanja i isključivanja sadržaja
  - +(S) - sadržaj S se može pojaviti.
  - -(S) - sadržaj S se ne sme pojaviti
- Definicije elemenata mogu da sadrže reference parametarskih entiteta
- Primer: Delovi DTD za zbirku pesama

```
<!ELEMENT zbirka - - (pesma+)>
```

Element **zbirka** u sebi sadrži jedan ili više elemenata **pesma**, pri čemu se obe etikete moraju navoditi

```
<!ELEMENT pesma - - (naslov?, strofa+)>
```

Element **pesma** može, a ne mora, da sadrži element **naslov** za kojim sledi jedan ili više elemenata **strofa**. Obe etikete se opet moraju navesti



## Definicije tipa dokumenta (5)

- Deklaracije elemenata

- Primer: Delovi DTD za zbirku pesama

```
<!ELEMENT stih 0 0 (#PCDATA)>
```

Sadržaj elementa **stih** je proizvoljan tekst koji može da uključi i reference entiteta, ali ne sme da uključi druge elemente

- Primer: Element u HTML-u koji predstavlja hiper-vezu

```
<!ELEMENT A - - (%inline;)* -(A)>
```

Ovde je korišćeno je dodatno pravilo isključivanje sadržaja, pa element **A** sadrži nula ili više elemenata obuhvaćenih parametarskim entitetom **%inline;**, ali ne sme da sadrži drugi element **A**



## Definicije tipa dokumenta (6)

### ● Deklaracije atributa

- Deklaracija atributa u okviru DTD počinje sa `<!ATTLIST`, nakon koga se navodi element za koji se deklariše atribut, potom sledi lista deklaracija pojedinačnih atributa i na kraju se navodi simbol `>`
- Svaka deklaracija pojedinačnih atributa je trojka koja definiše:
  1. Ime atributa
  2. Tip vrednosti atributa, ili eksplicitno naveden skup dopustivih vrednosti  
Najčešće korišćeni tipovi su:
    - CDATA (character data) - kao i u slučaju elemenata, označava tekst koji se neće analizirati pomoću SGML parsera
    - NAME - označava imena
    - ID - Označava jedinstvene identifikatore tj. imena koja moraju biti jedinstvena u celom dokumentu
    - NUMBER - Označava brojevne vrednosti
  3. Naznaku da li je vrednost atributa implicitna, fiksirana ili zahtevana



# Definicije tipa dokumenta (7)

## ● Deklaracije atributa

- Ako je naznačeno da je vrednost atributa implicitna (ključna reč #IMPLIED), to znači da podrazumevanu vrednost određuje softver koji vrši obradu dokumenta
  - Ako je naznačeno da je vrednost atributa fiksirana (ključna reč #FIXED), to podrazumeva da atribut može da ima samo jedinu moguću vrednost koja je u nastavku navedena
  - Ako je naznačeno da je vrednost atributa zahtevana (ključna reč #REQUIRED), tada je na ovom mestu moguće i eksplicitno specificirati podrazumevanu vrednost atributa
- Naravno, definicije atributa mogu da sadrže reference parametarskih entiteta



# Definicije tipa dokumenta (8)

- Deklaracije atributa

- Primer: Delovi DTD za zbirku pesama

```
<!ATTLIST pesma  
  autor CDATA #REQUIRED  
>
```

Ovim je za element **pesma** deklarisan atribut **autor**, čija je vrednost neki tekst, pri čemu je navođenje atributa obavezno

- Primer: Delovi DTD za tabelu u HTML-u

```
<!ATTLIST td  
  rowspan NUMBER 1  
  colspan NUMBER 1  
>
```

Ovim se za element **td** uvode atributi **rowspan** i **colspan** čije su vrednosti brojevi, dok je podrazumevana vrednost za oba atributa 1



# Definicije tipa dokumenta (9)

- Deklaracije atributa

- Primer: Delovi DTD za deo HTML-a

```
<!ATTLIST html  
  version CDATA #FIXED "%HTML.Version"  
>
```

Ovim se označava da vrednost atributa **version** elementa **html** može da bude isključivo vrednost određena parametarskim entitetom **HTML.Version** (koji definiše tekuću verziju)



# Uključivanje DTD

- DTD može biti naveden ili kroz unutrašnju ili kroz spoljašnju deklaraciju
  - Unutrašnja deklaracija podrazumeva da se DTD deklaracije nalaze u zaglavlju datoteke u kojoj je smešten dokument

## Primer: Unutrašnja deklaracija

```
<!-- test.sgml -->
<!DOCTYPE test
  <!ELEMENT test - - PCDATA>
>
<test>Zdravo</test>
```

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend</body>
</note>
```





## Uključivanje DTD (2)

### ○ Primer: Unutrašnja deklaracija

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

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```



## Uključivanje DTD (2)

- DTD može biti naveden ili kroz unutrašnju ili kroz spoljašnju deklaraciju
  - Spoljašnja deklaracija podrazumeva da se DTD deklaracije nalaze u spoljašnjoj datoteci, bilo na lokalnom sistemu ili javno na vebu
  - U tom slučaju se u okviru `<!DOCTYPE>` navodi ime datoteke koja sadrži DTD
  - Primer: Spoljašnja deklaracija

```
<!-- test.sgml -->  
<!DOCTYPE test SYSTEM "test.dtd">  
<test>Zdravo</test>
```

```
<!-- test.dtd -->  
<!ELEMENT test - - PCDATA>
```



## Uključivanje DTD (3)

### Primer: Spoljašnja deklaracija

#### note.dtd

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

#### note.xml

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```



# XML





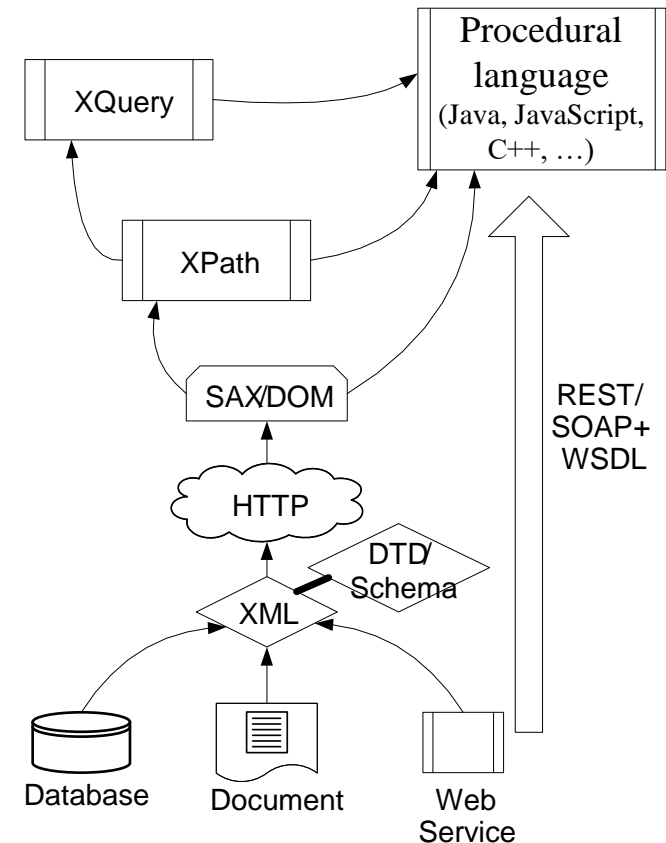
# Šta je XML?

Hijerarhijski format čitljiv za čoveka

- Jezik “potomak” HTML-a, koji se uvek može parsirati
- “Lingua franca” za podatke: služi za čuvanje dokumenata strukturisanih podataka
- Smešani su podaci i struktura

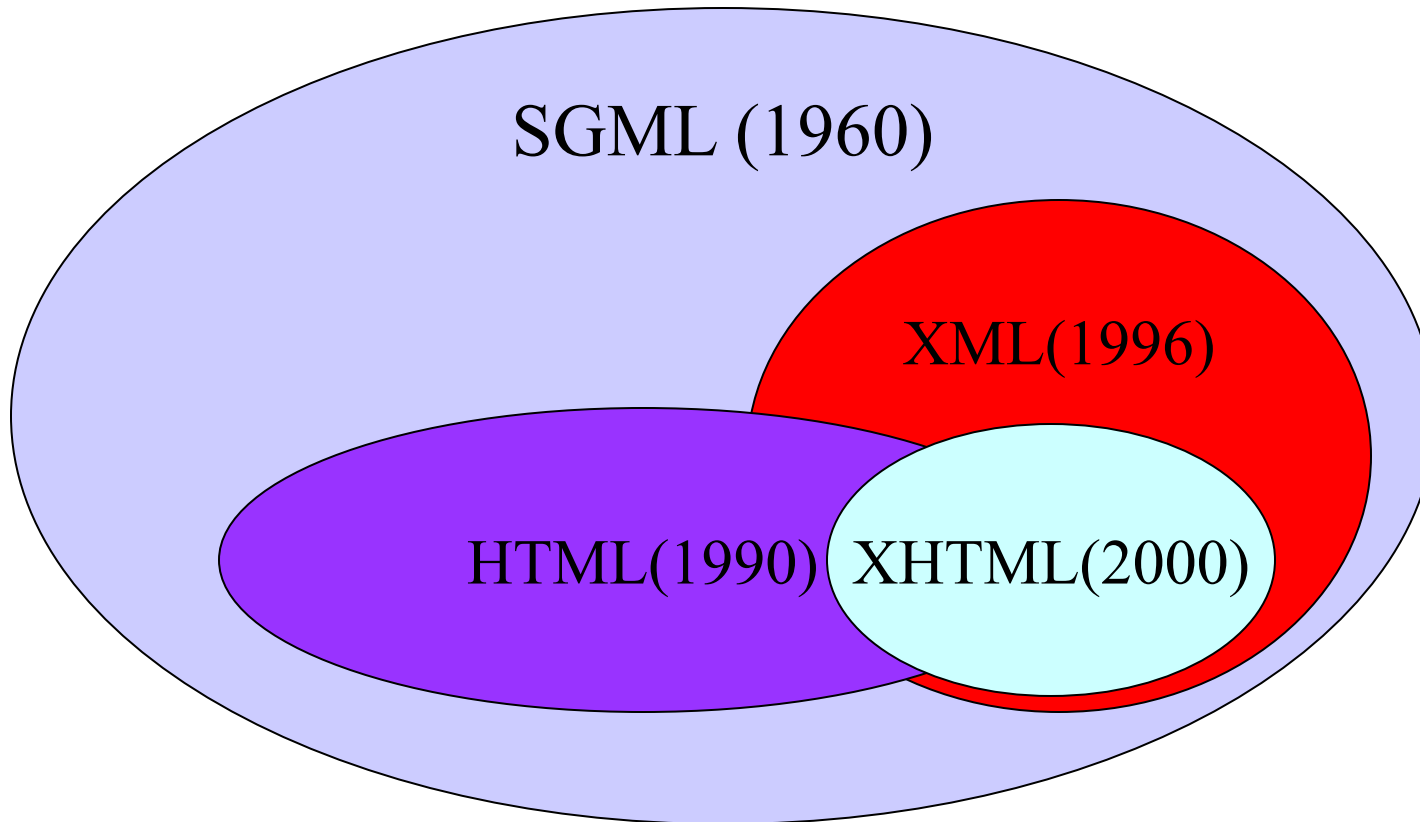
Jezgro šireg ekosistema

- Podaci – XML
- Shema – DTD i XML Shema
- Programerski pristup – DOM i SAX
- Upiti – XPath, XSLT, XQuery
- Distribuisano programiranje – veb servisi





# Istorija: SGML vs. HTML vs. XML





# Zašto XML

- Olakšava težnju da se „sadržaj“ razdvoji od „prezentacije”
  - Prezentacija obezbeđuje lepotu pri posmatranju
  - Sadržaj se može interpretirati od strane računara, a za računare prezentacija predstavlja hendikep
- Semantičko označavanje podataka
- XML je „polu-struktuiran“

```
<parents>
  <parent name="Jean" >
    <son>John</son>
    <daughter>Joan</daughter>
    <daughter>Jill</daughter>
  </parent>
  <parent name="Feng">
    <daughter>Ella</daughter>
  </parent>
```

...



# Zašto XML (2)

```
<book year="1967">  
  <title>Politics of experience</title>  
  <author>  
    <firstname>Ronald</firstname>  
    <lastname>Laing</lastname>  
  </author>  
</book>
```

Informacije o knjizi sačuvane u XML formatu

- Informacija je:
  1. razdvojena od prezentacije, pa
  2. isečena u male delove, i na kraju
  3. označena sa semantičkim značenjem
- Informacija u ovom formatu se lako može procesirati računarima
- XML opisuje samo sintaksu, a ne apstraktni logički model podataka





# Ključni pojmovi XML-a

- To su:
  - Dokumenti
  - Elementi
  - Atributi
  - Deklaracije prostora imena
  - Tekst
  - Komentari
  - Instrukcije procesiranja
- Svi ovi pojmovi su nasleđeni iz SGML-a



# Anatomija XML-a

<?xml version="1.0" encoding="ISO-8859-1" ?>

<dblp> ← *Otvorajuća etiketa*

<mastersthesis mdate="2002-01-03" key="ms/Brown92">

<author>Kurt P. Brown</author>

<title>PRPL: A Database Workload Specification

Language</title>

<year>1992</year>

<school>Univ. of Wisconsin-Madison</school>

</mastersthesis>

<article mdate="2002-01-03" key="tr/dec/SRC1997-018">

<editor>Paul R. McJones</editor>

<title>The 1995 SQL Reunion</title>

<journal>Digital System Research Center Report</journal>

<volume>SRC1997-018</volume>

<year>1997</year>

<ee>db/labs/dec/SRC1997-018.html</ee>

<ee>http://www.mcjones.org/System\_R/SQL\_Reunion\_95/</ee>

</article>

*Instrukcija procesiranja*

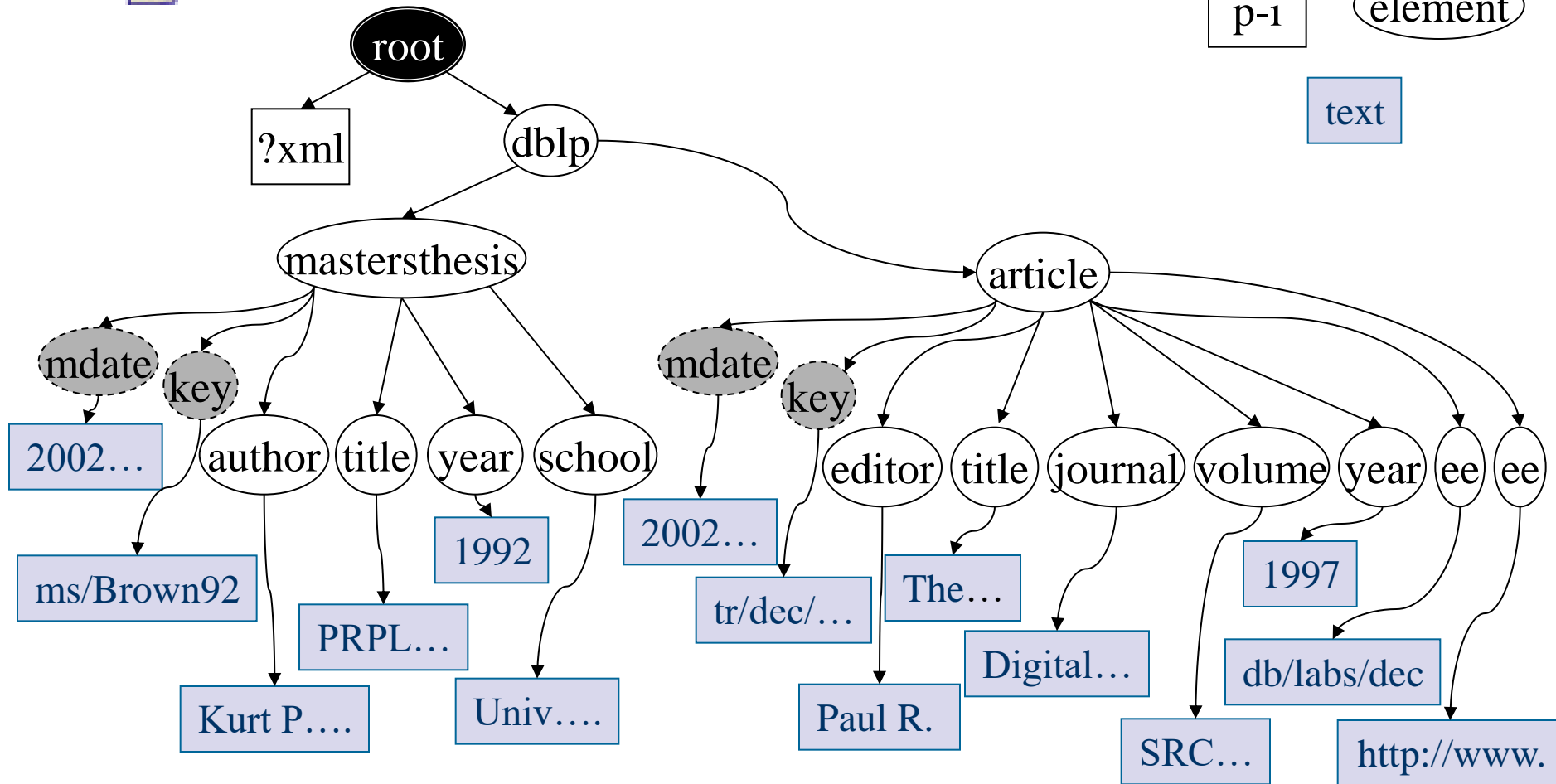
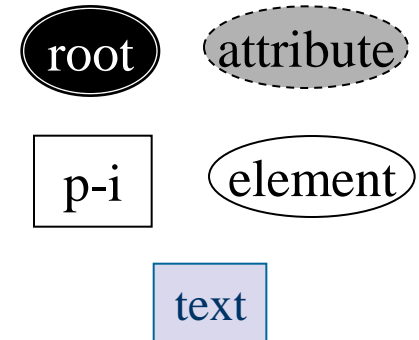
*Element*

*Atribut*

*Zatvarajuća etiketa*



# Anatomija XML-a (2)





# Anatomija XML-a (3)

- XML lako čuva relacije  
Primer: Relacija student-course-grade

sid	cid	exp-grade
1	570103	B
23	550103	A

```
<student-course-grade>
  <tuple><sid>1</sid><cid>570103</cid>
    <exp-grade>B</exp-grade>
</tuple>
<tuple><sid>23</sid><cid>550103</cid>
  <exp-grade>A</exp-grade>
</tuple>
</student-course-grade>
```

ili

```
<student-course-grade>
  <tuple sid="1" cid="570103" exp-grade="B"/>
  <tuple sid="23" cid="550103" exp-grade="A"/>
</student-course-grade>
```



# Elementi

```
<book year="1967">  
  <title>Politics of experience</title>  
  <author>  
    <firstname>Ronald</firstname>  
    <lastname>Laing</lastname>  
  </author>  
</book>
```

Elemente karakteriše:

- Ugnježdena struktura
- Drvoidna struktura
- Redosled je važan
- Sadrži samo znakove, a ne cele brojeve, itd.



## Elementi (2)

- Obuhvaćeni su etiketama
  - Otvarajuća etiketa: npr. `<bibliography>`
  - Zatvarajuća etiketa: npr. `</bibliography>`
  - Elementi bez sadržaja (prazni): npr. `<bibliography />` je skraćenica za `<bibliography> </bibliography>`
- Elementi mogu biti ugnježdeni  
`<bib> <book> Wilde Wutz </book> </bib>`
- Elementi koji su ugnježdeni mogu biti višečlani  
`<bib> <book> ... </book> <book> ... </book> </bib>`
- U tim slučajevima, redosled je veoma važan!
- Dokumenti moraju biti dobro formirani  
`<a> <b> </a> </b>` nije dopušteno!  
`<a> <b> </b>` nije dopušteno!



# Atributi

- Atributi su pridruženi elementima

Primer:

```
<book price = "55" year = "1967" >  
  <title> ... </title>  
  <author> ... </author>  
</book>
```

- Elementi mogu sadržavati samo attribute (unutar otvarajuće etikete)

Primer: `<person name = "Wutz" age = 33"/>`

- Imena atributa moraju biti jedinstvena!

Primer: Nelegalna je sledeća konstrukcija

```
<person name = "Wilde" name = "Wutz"/>
```

- Koja je razlika između umetnutog elementa i atributa?

Da li su atributi korisni?

- Odluka pri modeliranju: da li da **name** bude atribut ili element ugnježđen u element **person**?

Šta da se radi sa elementom **age**?



# Tekst i izmešani sadržaj

- Tekst se može javiti unutar sadržaja elementa

Primer:

```
<title>The politics of experience</title>
```

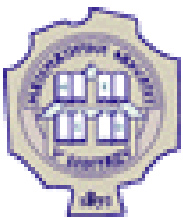
- Tekst može biti izmešan sa ostalim elementima ugnježđenim u dati element

Primer:

```
<title>The politics of <em>experience</em></title>
```

- Karakteristike izmešanog sadržaja:
  - Veoma je koristan za podatke u obliku dokumenata, tj. rečenica
  - Nema potreba za mešanim sadržajem u scenarijima „procesiranja podataka“, jer se tada obično obrađuju entiteti i relacije
  - Ljudi komuniciraju rečenicama, a ne entitetima i relacijama. XML omogućuje da se sačuva struktura prirodnog jezika, uz dodavanje semantičkih oznaka koje mogu računarski interpretirane





# Prelaz između prirodnog jezika, polu-strukturiranih i strukturiranih podataka

## 1. Prirodni jezik:

Dana said that the book entitled "The politics of experience" is really excellent !

## 2. Polu-strukturisani podaci (tekst):

```
<citation author="Dana"> The book entitled "The politics of experience" is really excellent ! </citation>
```

## 3. Polu-strukturisani podaci (mešani sadržaj):

```
<citation author="Dana"> The book entitled <title> The politics of experience</title> is really excellent ! </citation>
```

## 4. Strukturisani podaci:

```
<citation>
  <author>Dana</author>
  <aboutTitle>The politics of experience</aboutTitle>
  <rating> excellent</rating>
</citation>
```



# Sekcija CDATA

- Ponekad treba sačuvati originalne znake, a ne interpretirati njihova označavanja
- Sekcija CDATA određuje da se sadržaj unutar nje ne parsira kao XML
- Primer (poruka Hello,world! je označena):

```
<message>  
  <greeting>Hello,world!</greeting>  
</message>
```

- Primer (označena poruka neće biti parsirana kao XML):

```
<message>  
  <![CDATA[<greeting>Hello, world!</greeting>]]>  
</message>
```



# Komentari, instrukcije za procesiranje i prolog

- Komentar je tekst između `<!--` i `-->`

Primer: `<!-- ovo je komentar -->`

- Instrukcije za procesiranje

One ne sadrže podatke, već ih interpretira procesor

Sastoje se od para reči **meta sadržaj**, razdvojenih zarezom, kojima prethodi `<?`, a iza kojih sledi `?>`

Primer: U instrukciji za procesiranje `<?pause 10 secs ?>` **pause** je meta, a **10secs** je sadržaj

- Reč **xml** je rezervisana reč za metu, koja služi za označavanje prologa

- Prolog

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

Napomene:

- Atribut **standalone** određuje da li postoji DTD
- Encoding je obično Unicode.



# Deklaracija korišćenja belina

- Beline predstavljaju neprekidnu sekvencu znakova **Space**, **Tab** i **Return**
- Za kontrolu korišćenja belina služi specijalan atribut **xml:space**
- Primer čitljivog XML-a (koji sadrži beline):  

```
<book xml:space="preserve" >  
  <title>The politics of experience</title>  
  <author>Ronald Laing</author>  
</book>
```
- Primer efikasnog (računarski čitljivog) XML-a:  

```
<book xml:space="default" ><title>The politics of  
experience</title><author>Ronald Laing</author></book>
```
- U drugom primeru su (u odnosu na prvi) performanse ubrzane sa faktorom 2



# Prostori imena

- Omogućavaju integraciju podataka iz različitih izvora
- Omogućavaju integraciju različitih XML rečnika (tj. prostora imena)
- Svaki „rečnik“ ima jedinstven ključ, identifikovan URI-jem
- Isto lokalno ime iz različitih rečnika može imati
  - Različita značenja
  - Različite pridružene strukture
- Kvalifikovana imena (Qualified Names - QName) služe za prigrušivanje imena „rečniku“
- Kvalifikovana imena se odnose na sve čvorove XML dokumenta koji imaju imena (atribute, elemente, Instrukcije za procesiranje)



# Prostori imena (2)

- Način korišćenja
  - Povezivanje (tj. prefiks i URI) se uvode u otvarajućoj etiketi elementa
  - Kasnije se za opis koristi prefiks, a ne URI
  - Postoje podrazumevani prostori imena, pa je prefiks opcionalan
  - Prefiks se od lokalnog imena razdvaja dvotačkom, tj. znakom :
- Prostori imena se zapisuju slično atributima
  - Identifikuju se bilo sa “xmlns:prefix“, ili sa “xmlns“ (ako se radi o podrazumevanom prostoru imena)
  - Dati prefiks se, korišćenjem prostora imena, pozezuje sa URI-jem
- Opseg prostora imena je ceo elemenat u kome je taj prostor imena deklarisan – uključuje sam elemenat, njegove attribute i sve elemente koji su ugnježdeni u njega
- Primer:

```
<ns:a xmlns:ns="someURI" ns:b="foo">  
  <ns:b>content</ns:b>  
</ns:a>
```



# Podrazumevani prostori imena

- Kad se specificira podrazumevani prostor imena, ne koristi se prefiks

```
<a xmlns="someURI" >  
    <b/> <!-- a and b are in the someURI namespace! -->  
</a>
```

- Podrazumevani prostor imena se odnosi samo na ugnježdene elemente, a ne na atribut

```
<a xmlns="someURI" c = "not in someURI namespace">  
    <b/> <!-- a and b are in the someURI namespace! -->  
</a>
```



# Primer rada sa prostorima imena

- Tanjiri iz servisa za ručavanje i satelitski „tanjiri“
  - Neka „rečnik“ DQ1 definiše dish for china
    - Diameter, Volume, Decor, ...
  - Neka „rečnik“ DQ2 definiše dish for satellites
    - Diameter, Frequency
  - Postavlja se pitanje: Koliko ovde ima „tanjira“?
  - To pitanje se svodi na jedno od sledeća dva pitanja:
    1. „How many dishes are there?“  
or
    2. „How many dishes are there?“





# Primer rada sa prostorima imena (2)

XML opis „tanjira“ iz pribora za ručavanje:

```
<gs:dish xmlns:gs = "http://china.com" >  
  <gs:dm gs:unit = "cm">20</gs:dm>  
  <gs:vol gs:unit = "l">5</gs:vol>  
  <gs:decor>Meissner</gs:decor>  
</gs:dish>
```

XML opis „tanjira“ za prijem satelitskog signala:

```
<sat:dish xmlns:sat = "http://satelite.com" >  
  <sat:dm>200</sat:dm>  
  <sat:freq>20-2000MHz</sat:freq>  
</sat:dish>
```



# Primer rada sa prostorima imena (3)

XML opis „tanjira“ za ručavanje, gde su merene jedinice iz drugog prostora imena:

```
<gs:dish xmlns:gs = "http://china.com" xmlns:uom =  
  "http://units.com">  
  <gs:dm uom:unit = "cm">20</gs:dm>  
  <gs:vol uom:unit = "l">5</gs:vol>  
  <gs:decor>Meissner</gs:decor>  
  <comment>This is an unqualified element name</comment>  
</gs:dish>
```



# Primer rada sa prostorima imena (4)

- Kod prostora imena se razlikuju:

- Vezivanje prostora imena sa URI-jem
- Kvalifikovanje imena

*Podrazumevani prostor imena se odnosi na sva imena koja nisu kvalifikovana*

```
<root xmlns="http://www.first.com/aspace" xmlns:othersns="...">
  <myns:tag xmlns:myns="http://www.fictitious.com/mypath">
    <thistag>is in the default namespace
      (www.first.com/aspace)</thistag>
    <myns:thistag>is in myns</myns:thistag>
    <othersns:thistag>is a different tag in
      othersns</othersns:thistag>
  </myns:tag>
</root>
```

*Definiše "othersns" kvalifikator*



# Primeri XML podataka

- XHTML (pregledač/prezentacija)
- RSS (blogovi)
- UBL (univerzalni poslovni jezik)
- HealthCare Level 7 (medicinski podaci)
- XBRL (finansijski podaci)
- XMI (meta podaci)
- XQueryX (programo)
- XForms, FXML (forme)
- SOAP (poruke za komunikaciju)
- Microsoft ADO.Net (baze podataka)
- Microsoft Office, Powerpoint (dokumenti)



# Struktuiranje XML-a

- Za razliku od drugih formata podataka, XML je veoma fleksibilan i elementi se mogu umetati na različite načine
- Može se početi sa pisanjem XML-a koji predstavlja podatke i bez prethodnog dizajniranja strukture
  - Tako se radi kod relacionih baza podataka ili kod Java klasa
- Međutim, strukturisanje ima veliki značaj:
  - Podspešuje pisanje aplikacija koje procesiraju podatke
  - Ograničava podatke na one koje su korektni za datu aplikaciju
  - Definiše „a priori“ protokol o podacima koji se razmenjuju između učesnika u komunikaciji
- Struktura XML-a modelira podatke i sadrži:
  - Definicije struktura
  - Definicije tipova
  - Podrazumevane vrednosti



# Struktuiranje XML-a (2)

Karakteristike struktuiranja:

- Nije formalizovano na način kako je to urađeno kod relacionih baza podataka
  - Ono je obično zasnovano na strukturi koja se već nalazi u podacima, npr relacionoj SUBP ili raširenoj elektronskoj tabeli
- Šri struktuiranju se XML drvo orijentiše prema „centralnim” objektima
- Velika dilema: element ili atribut
  - Element se koristi za osobinu koja sadrži svoje osobine ili kada se može očekivati da će biti više takvih unutar elementa koji ih sadrži
  - Atribute se koristi kada se radi o jednoj osobini – mada je OK da se i tada koristi element!



# Istorija i uloga jezika za opis strukture XML-a

- Postoji nekoliko standardnih jezika za opis strukture XML-a  
DTD-ovi, XML Shema, RelaxNG
- Jezici koji opisuju strukturu se definišu ortogonalno u odnosu na sam XML
- Kod XML-a su opis strukture i podaci potpuno razdvojeni
  - Podaci mogu postojati i uz opis strukture i bez njega
  - Podaci mogu postojati i uz više opisa struktura
  - Evolucija strukture veoma retko dovodi do evolucije podataka
  - Može se raditi tako što se struktura definiše pre podataka, a može i tako što se struktura ekstrahuje iz podataka
- Jezici za opis strukture čine da XML postaje pravi izbor za manipulaciju polu-strukturisanim podacima, podacima koji brzo evoluiraju ili podacima koji su podesivi u velikoj meri



# Korektnost XML dokumenata

## 1. Dobro formirani dokumenti

- Kod njih se verifikuju samo osnovna XML ograničenja, npr. `<a></b>`

## 2. Validni dokumenti

- Kod njih se verifikuju dodatna ograničenja opisana u nekom od jezika za opis strukture (npr. DTD, XML shema)

- XML dokumenti koji nisu dobro formirani ne mogu biti procesirani
- XML dokumenti koji nisu validni ipak mogu biti procesirani (mogu se upitima izvlačiti podaci, mogu biti transformisani, itd.)





# XML i DTD





# DTD

- DTD je deo originalne XML 1.0 specifikacije
- DTD opisuje “gramatiku” za XML datoteku
  - **Deklaracije elemenata**: pravila i ograničenja koja opisuju dopuštene načine ugnježdavanja elemenata
  - **Attributes lists**: opisuje koji su atributi dopušteni nad kojim elementima
  - Dodatna ograničenje na vrednosti elemenata i atributa
  - Koji je element koreni čvor XML strukture
- Provera strukturnih ograničenja pomoću DTD se naziva **DTD validacija** (određuje se da li je XML dokument validan ili invalidan)
- Zbog svojih ograničenja, DTD se sada relativno retko koristi za validaciju XML dokumenata



# Referisanje na DTD u okviru XML-a

- Nema DTD-a (radi se o dobro formiranom XML dokumentu)
- DTD je unutar dokumenta:  
`<!DOCTYPE name [definition] >`
- Spoljašnji DTD, specificiran URI-jem:  
`<!DOCTYPE name SYSTEM "demo.dtd">`
- Spoljašnji DTD, dato je ime i (opcionarno) URI:  
`<!DOCTYPE name PUBLIC "Demo">`  
`<!DOCTYPE name PUBLIC "Demo" "demo.dtd">`
- DTD je unutrašnji + spoljašnji:  
`<!DOCTYPE name1 SYSTEM "demo.dtd" >`



# Primeri XML validacije sa DTD

Primer DTD-a koji opisuje strukturu dblp sloga:

```
<!ELEMENT dblp((mastersthesis | article)*)>
<!ELEMENT
  mastersthesis(author,title,year,school,committeemember*)>
<!ATTLIST mastersthesis(mdate CDATA #REQUIRED
  key ID #REQUIRED
  advisor CDATA #IMPLIED>
<!ELEMENT author(#PCDATA)>
```

...

Primer referisanja na DTD u okviru XML datoteke:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE dblp SYSTEM "my.dtd">
<dblup>...
```



# Primeri XML validacije sa DTD

## (2)

Primer dela DTD koji opisuje strukturu knjige i indeksa:

```
<!ATTLIST book
    isbn      ID          #REQUIRED
    price     CDATA       #IMPLIED
    index     IDREFS      "" >
```

...

Primer dela XML-a koji opisuje knjige:

```
<book id="1" index="2 3 " >
```

```
<book id="2" index="3"/>
```

```
<book id ="3"/>
```



# Primeri XML validacije sa DTD

## (3)

Primer dela XML-a sa identifikatorima i referencama na identifikatore:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<!DOCTYPE graph SYSTEM "special.dtd">
```

```
<graph>
```

```
  <author id="author1">  
    <name>John Smith</name>
```

*Pretpostavimo da je definisano  
da ovo bude tipa ID*

```
  </author>
```

```
  <article>
```

```
    <author ref="author1" /> <title>Paper1</title>
```

```
  </article>
```

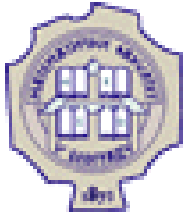
*Pretpostavimo da je ovo tipa  
IDREF*

```
  <article>
```

```
    <author ref="author1" /> <title>Paper2</title>
```

```
  </article>
```

...



# XML Sheme





# Ograničenja DTD-ova

- DTD opisuje samo „gramatiku“ XML datoteke, a ne detaljnu strukturu niti tipove
- Tako, na primer, preko DTD se ne može iskazati da:
  - elemenat “length” mora sadržavati nenegativan ceo broj  
*(ograničenje koje se odnosi a tip vrednosti elementa ili atributa)*
  - elemenat “unit” treba da bude dopušten samo onda kada je prisutan elemenat “amount” *(ograničenje koje se odnosi na zajedničko pojavljivanje)*
  - elemenat “comment” može da se pojavi na bilo kom mestu  
*(fleksibilnost sheme)*
  - DTD-ov ID nije preterano dobra implementacija za vrednost ključa
  - Ne postoji podrška za nasleđivanje kao kod objektno-orijentisanih jezika
  - Sintaksa koja je bliska XML-u, ali nije XML nije pogodna da se na toj osnovi razvijaju alati





# Principi dizajna za sheme

Jezik XML shema treba da bude:

1. Izražajniiji od XML DTD-ova
2. Izražen pomoću XML-a
3. Samo-opisiv
4. Pogodan za korišćenje za širok opseg aplikacija koje koriste XML
5. Direktno pogodan za korišćenje na Internetu
6. Optimizovan za interoperabilnost
7. Dovoljno jednostavan da se može implementirati na skromnim resursima
8. Usaglašen sa relevantnim W3C specifikacijama



# Osnove XML sheme

Kreirana tako da prevaziđe probleme sa DTD-ovima

- Ima XML sintaksu
- Može definisati ključeve korišćenjem XPath konstrukcija
- Tip korenog elementa za dokument je globalno naniže komptibilan sa DTD
- Prostori imena su deo XML shema
- Podržano je nasleđivanje tipova, koje uključuje i ograničavanje opsega
  - Nasleđivanje proširivanjem (by extension) kojim se dodaju novi podaci
  - Nasleđivanje ograničavanjem (by restriction) kojim se dodaju nova ograničenja
- Podržani su domeni i predefinisani tipovi podataka



# Osnove XML sheme (2)

- Prosti tipovi predstavljaju način restrikcije domena na skalarne vrednosti
  - Tako se može definisati prosti tip zanosvan na celobrojnomo tipu, pri čemu su vrednosti tog prostog tipa utar zadatog opsega
- Složeni tipovi su način definisanja struktura element/atribut
  - U osnovi ekvivalentni sa !ELEMENT kod DTD-a, ali moćnije
  - Specificira bilo sekvencu, bilo izbor među elementima - potomcima
  - Specificira minimalni i maksimalni broj pojavljivanja (minOccurs i maxOccurs), pri čemu je podrazumevana vrednost 1
- Elementima se može pridružiti složeni tip ili prosti tip
- Atributima se može pridružiti samo prosti tip
- Tipovi mogu biti predefinisani ili korisnički
  - Složeni tipovi mogu biti samo korisnički, ne mogu biti predefinisani



# Struktura sheme

Primer dela sheme koji opisuje strukturu knjige:

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://w3.org/2001/XMLSchema">
  <xsd:element name="book" type="BookType"/>
  <xsd:complexType name="BookType">
    <xsd:sequence>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="author" type="PersonType"
        minOccurs="1" maxOccurs="unbounded"/>
      <xsd:complexType name="PersonType">
        <xsd:sequence> ... <xsd:sequence>
      </xsd:complexType>
      <xsd:element name="publisher" type="xsd:anyType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```



# Prolog sheme

Primer prologa sheme:

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

- Napomene:

- Shema se obično čuva u odvojenom XML dokumentu
- Rečnik za shemu se definiše u specijalnom prostoru imena, a kao prefiks se obično koristi **xsd**
- Postoji shema koja opisuje XML sheme
- Element **schema** je uvek koren za XML shemu



# Globalne deklaracije

- Instance (tj. primerci) globalne deklaracije elementa predstavljaju potencijalne korene elemente za date XML dokumente
- Globalne deklaracije se mogu referencirati na sledeći način:

```
<xsd:schema xmlns:xsd="...">  
  <xsd:element name="book" type="BookType"/>  
  <xsd:element name="comment" type="xsd:string"/>  
  <xsd:ComplexType name="BookType">  
    ... <xsd:element ref="comment" minOccurs="0"/>  
  ...  
...
```

- Ograničenja
  - **ref** se ne može koristiti u globalnoj deklaraciji
  - Ni `minOccurs`, `maxOccurs` se ne mogu koristiti u globalnoj deklaraciji



# Deklaracija globalnog elementa

Primer definisanja elementa u XML shemi:

```
<xsd:element name="book" type="BookType"/>
```

- Napomene:

- Etiketa `element` služi za deklarisanje elemenata
- Atribut `name` služi za imenovanje elementa
- Atribut `type` definiše tip elementa

- Primer:

U prethodnom slučaju, tip elementa `book` je tip `BookType`, koji je definisan u nastavku

- Declaracije direktno unutar elementa `schema` su **globalne**

- Samo oni elementi čije su deklaracije globalne mogu doći u obzir da budu koreni za XML shemu

- Primer:

U prethodnom slučaju, jedini globalni element je `book`, pa stoga koren za validni XML dokument opisano ovim shemom mora biti `book`



# Deklaracija globalnog tipa

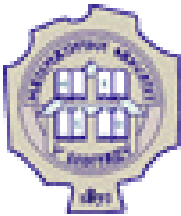
Primer definisanja složenog tipa u XML shemi:

```
<xsd:complexType name="BookType">  
  <xsd:sequence>  
    ...  
  </xsd:sequence>  
</xsd:complexType>
```

- Napomene:

- Ovaj složen tip je definisan kao sekvenca elemenata
- Atribut **name** određuje ime tipa
- Ova definicija tipa je **globalna** (nalazi se direktno unutar elementa **schema**), pa se ovako definisan tip može koristiti u ma kojoj drugoj definiciji





# Lokalni elemenat

Primer definisanja lokalnog elementa u XML shemi:

```
<xsd:sequence>  
    <xsd:element name="title" type="xsd:string"/>  
</xsd:sequence>
```

- Napomene:

- Ovo je lokalni elemenat, jer se ne nalazi direktno u elementu schema, već unutar kompleksnog tipa  
Dakle, elemenat **title** ne može biti koreni elemenat dokumenta
- Atribut **name** služi za imenovanje elementa
- Atribut **type** definiše tip elementa
- Tip **xsd:string** je predefinisani tip za XML shemu



# Deklaracija lokalnog elementa

Primer definisanja lokalnog elementa **author**:

```
<xsd:element name="author" type="PersonType"  
minOccurs="1" maxOccurs="unbounded"/>
```

## ● Napomene:

- Ako se analizira cela XML shema, jasno je da se radi o deklaraciji lokalnog elementa
- Tip **PersonType** je korisnički definisan tip
- Atributi minOccurs i maxOccurs određuju kardinalnost elementa **author** u tipu **BookType**
- Ovakva vrsta definisanja atributa se u žargonu naziva „brušenje“ („facet“);
- Postoji 15 različitih vrsta brišenja, npr. minExclusive, totalDigits, itd.
- Podrazumevane vrednosti za ove attribute su minOccurs=1, maxOccurs=1



# Definicija lokalnog tipa

Primer definisanja lokalnog tipa `PersonType`:

```
<xsd:complexType name=„PersonType“>
  <xsd:sequence>
    <xsd:element name=„first“ type=„xsd:string“/>
    <xsd:element name=„last“ type=„xsd:string“/>
  <xsd:sequence>
</xsd:complexType>
```

- Napomene:

- S obzirom da je definisan u okviru definicije tipa `BookType`, tip `PersonType` je **lokalan** i može biti korišćen samo unutar opsega definicije tipa `BookType`
- Sintaksa ovog tipa je slična sintaksi tipa `BookType`



# Definicija lokalnog elementa

Primer definisanja lokalnog elementa `publisher`:

```
<xsd:element name="publisher" type="xsd:anyType"/>
```

- Napomene:

- S obzirom da je definisan u okviru definicije tipa `BookType`, ovaj element je **lokalan**
- Svaka knjiga ima tačno jedan elemenat `publisher`
- Brušenja `minOccurs`, `maxOccurs` imaju podrazumevanu vrednost 1
- Tip `anyType` je predefinisani tip koji dozvoljava bilo kakav sadržaj
- Tip `anyType` je podrazumevan – ako se ništa ne napiše, onda se radi o tom tipu

Definicija koja sledi je ekvivalentna definciji iz primera:

```
<xsd:element name="publisher" />
```



# Primer sheme

Shema koji opisuje strukturu knjige:

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://w3.org/2001/XMLSchema">
  <xsd:element name="book" type="BookType"/>
  <xsd:complexType name="BookType">
    <xsd:sequence>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="author" type="PersonType"
        minOccurs="1" maxOccurs="unbounded"/>
      <xsd:complexType name="PersonType">
        <xsd:sequence> ... <xsd:sequence>
      </xsd:complexType>
      <xsd:element name="publisher" type="xsd:anyType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```



# Primer sheme (2)

XML koji je saglasan sa prethodnom shemom:

```
<?xml version=„1.0“>
<book>
  <title>Die Wilde Wutz</title>
  <author><first>D.</first>
    <last>K.</last></author>
  <publisher>
    Addison Wesley, <state>CA</state>, USA
  </publisher>
</book>
```



# Primer sheme (3)

*Pridružuje "xsd" prostor imena sa XML shemom*

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

```
<xsd:element name="mastersthesis" type="ThesisType"/>
```

*ovo je element koren, sa tipom čija specifikacija sledi*

```
<xsd:complexType name="ThesisType">
```

```
<xsd:attribute name="mdate" type="xsd:date"/>
```

```
<xsd:attribute name="key" type="xsd:string"/>
```

```
<xsd:attribute name="advisor" type="xsd:string"/>
```

```
<xsd:sequence>
```

```
<xsd:element name="author" type="xsd:string"/>
```

```
<xsd:element name="title" type="xsd:string"/>
```

```
<xsd:element name="year" type="xsd:integer"/>
```

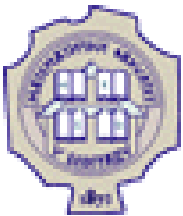
```
<xsd:element name="school" type="xsd:string"/>
```

```
<xsd:element name="committeemember" type="CommitteeType"
  minOccurs="0"/>
```

```
</xsd:sequence>
```

```
</xsd:complexType>
```

```
</xsd:schema>
```



# Deklaracije atributa

- Atributi mogu biti samo prostog tipa (npr. **string**)
- Deklaracije atributa mogu biti globalne
  - Takve deklaracije se mogu ponovo iskoristiti pomoću ref
- Deklaracije atributa su kompatibilne sa listom atributa u DTD
  - Moguće je korišćenje podrazumevanih vrednosti
  - Moguće je attribute proglasiti zahtevanim ili opcionalnim
  - Postojanje fiksnih atributa
  - Kao dodatna osobina, postoje i „zabranjeni“ atributi





# Deklaracije atributa (2)

Primer dela sheme koji opisuje strukturu knjige i indeksa:

```
<xsd:complexType name="BookType">
  <xsd:sequence> ... </xsd:sequence>

  <xsd:attribute name="isbn" type="xsd:string" use="required"/>
  <xsd:attribute name="price" type="xsd:decimal"
    use="optional" />
  <xsd:attribute name="curr" type="xsd:string"
    fixed="EUR" />
  <xsd:attribute name="index" type="xsd:idrefs"
    default="" />

</xsd:complexType>
```



# Anonimni tipovi

Primer dela sheme koji opisuje strukturu knjige, gde se tip koji opisuje osobu ne imenuje:

```
<xsd:complexType name="BookType">  
  ...  
  <xsd:element name="author">  
    <xsd:complexType>  
      <xsd:sequence>  
        <xsd:element name="first" type="xsd:string"/>  
        <xsd:element name="last" type="xsd:string"/>  
      </xsd:sequence>  
    </xsd:complexType>  
  </xsd:element>  
  ...
```



# Elementi i atributi

Primer dela sheme koji opisuje strukturu cene:

```
<xsd:element name="price">  
  <xsd:complexType>  
    <xsd:simpleContent>  
      <xsd:extension base="xsd:decimal" >  
        <xsd:attribute name="curr" type="xsd:string"/>  
      </xsd:extension>  
    </xsd:simpleContent>  
  </xsd:complexType>  
</xsd:element>
```

Primer za validan primerak cene:

```
<price curr="USD" >69.95</price>
```



# Elementi i atributi (2)

Primer dela sheme koji opisuje strukturu cene:

```
<xsd:element name="price">  
  <xsd:complexType>  
    <xsd:attribute name="curr" type="xsd:string"/>  
    <xsd:attribute name="val" type="xsd:decimal"/>  
  </xsd:complexType>  
</xsd:element>
```

Primer za validan primerak cene:

```
<price curr="USD" val="69.95" />
```



# Predefinisani prosti tipovi

- Numerički tipovi

Integer, Short, Decimal, Float, Double, HexBinary, ...

- Tipovi za datume i periode

Duration, DateTime, Time, Date, ...

- Tipovi za niske

String, NMToken, NMTOKENS, NormalizedString

- Ostali tipovi

QName, AnyURI, ID, IDREFS, Language, Entity, ...

- Kao zaključak, postoje 44 predefinisana prosta tipa



# Izvedeni prosti tipovi

Primer dela sheme sa tipom gde je izvršeno ograničavanje domena:

```
<xsd:simpleType name="MyInteger">  
  <xsd:restriction base="xsd:integer">  
    <xsd:minInclusive value="10000"/>  
    <xsd:maxInclusive value="99999"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

Primer definicije tipa gde je izvršeno ograničavanje domena preko regularnih izraza, tako da valute može biti zapisana samo pomoću tri velika slova:

```
<xsd:simpleType name="Currency">  
  <xsd:restriction base="xsd:string" >  
    <xsd:pattern value="[A-Z]{3}"/>  
  </xsd:restriction>  
</xsd:simpleType>
```



# Izvedeni prosti tipovi (2)

Primer definicije tipa gde je izvršeno ograničavanje domena preko enumeracije:

```
<xsd:simpleType name="Currency">  
  <xsd:restriction base="xsd:string" >  
    <xsd:enumeration value="ATS"/>  
    <xsd:enumeration value="EUR"/>  
    <xsd:enumeration value="GBP"/>  
    <xsd:enumeration value="USD"/>  
  </xsd:restriction>  
</xsd:simpleType>
```

## ● Napomene:

- Najveći broj predefinisanih tipova je izveden restrikcijom iz drugih predefinisanih tipova, npr tip Integer je izveden iz tipa Decimal
- Od 44 predefinisana tipa, samo njih 19 su osnovni tipovi



# Prosti tip listi

- Postoji više vrsta prostih tipova za liste:
  - Predefinisani tipovi listi: IDREFS, NMTOKENS
  - Korisnički definisani tipovi listi

Primer sheme za korisnički definisan tip liste

```
<xsd:simpleType name = "intList" >  
  <xsd:list itemType = "xsd:integer" />  
</xsd:simpleType>
```

- Karakteristike:
  - Elementi u primerku takve liste su razdvojeni belinama  
"5   -10   7                    -20"
  - Brušenja za restrikcije kod ovih listi su: length, minLength, maxLength, enumeration





# Prosti tip listi sa restrikcijom

Primer sheme za korisnički definisan tip liste sa restrikcijom:

```
<xsd:simpleType name = "Participants" >  
  <xsd:list itemType = "xsd:string" />  
</xsd:simpleType>  
  
<xsd:simpleType name = "Medalists" >  
  <xsd:restriction base = "Participants" >  
    <xsd:length value = "3" />  
  </xsd:restriction>  
</xsd:simpleType>
```



# Prosti tip unije

- Odgovara znaku | kod DTD
- Ima isto značenje kao slogovi sa promenljivim delom u Pascal-u ili kao unije u C-u
- Instance su validne ako su validne za jedan od pobrojanih tipova

Primer sheme sa prostim tipom unije:

```
<xsd:simpleType name = "Potpurri" >  
  <xsd:union memberTypes = "xsd:string intList"/>  
</xsd:simpleType>
```

Primer XML instanci validnih za gornju shemu:

```
"fünfzig" "1 3 17" "wunderbar" "15"
```

- Za prosti tip unije su podržana brušenja **pattern** i **enumeration**



# Element choice

Primer sheme za knjigu koja ima ili element **author** ili element **editor**:

```
<xsd:complexType name = "Book" > <xsd:sequence>  
  <xsd:choice>  
    <xsd:element name = "author" type = "Person"  
      maxOccurs = "unbounded" />  
    <xsd:element name = "editor" type = "Person" />  
  </xsd:choice>  
</xsd:sequence> </xsd:complexType>
```



# Grupe elemenata

Opis sheme kada se treba postići da ako element **book** sadrži element **editor**, tada **book** takođe sadrži i element **sponsor**:

```
<xsd:complexType name = „Book“ > <xsd:sequence>  
  <xsd:choice>  
    <xsd:element name = „Author“ type = „Person“ .../>  
    <xsd:group ref = „EditorSponsor“ />  
  </xsd:choice> </xsd:sequence> </xsd:complexType>
```

```
<xsd:group name = „EditorSponsor“ > <xsd:sequence>  
  <xsd:element name = „Editor“ type=„Person“ />  
  <xsd:element name = „Sponsor“ type = „Org“ />  
</xsd:sequence> </xsd:group>
```



# Grupe atributa

Opis sheme sa grupom atributa:

```
<xsd:attributeGroup name = „PriceInfo“ >  
  <xsd:attribute name = „curr“ type = „xsd:string“ />  
  <xsd:attribute name = „val“ type = „xsd:decimal“ />  
</xsd:attributeGroup>
```

```
<xsd:complexType name = „Book“ >  
  ...  
  <xsd:attributeGroup ref = „PriceInfo“ />  
</xsd:complexType>
```



# Definicija ključeva

- Ključevi jednoznačno identifikuju element i definisani su kao deo elementa
- Uveden je specijalni element koji se ugnježdava, nazvan **key**
  - U okviru tog novog elementa uvedeni su:
    - **selector**: opisuje kontekst na koji se odnosi ključ
    - **field**: opisuje koje je polje ključ u kontekstu opisanim selektorom
  - Ako ima više elemenata **field** u okviru ključa, tada se radi o tzv. kompozitnom ključu
- Vrednosti za selector i za field su XPath izrazi
- Validacija ključa u XML-u se realizuje na sledeći način:
  1. Evaluira se **selector** i dobije **sekvenca čvorova**
  2. Evaliraju se vrednosti za **field** na **sekvenci čvorova** i dobije se **skup uređenih n-torki vrednosti**
  3. Proverava se da li ima duplikata u **skupu uređenih n-torki vrednosti**



# Definicija ključeva (2)

Primer sheme u kojoj je **isbn** definisano kao ključ za **books** u **bib**:

```
<element name = „bib“> <complexType> <sequence>  
  <element book maxOccurs = „unbounded“  
    <complexType> <sequence> ... </sequence>  
    <attribute name = „isbn“ type = „string“ />  
  </complexType> </element> </sequence>  
  <key name = „constraintX“ >  
    <selector xpath = „book“ />  
    <field xpath = „@isbn“ />  
  </key>  
</complexType> </element>
```



# Reference (strani ključevi)

- Strani ključevi predstavljaju deo definicije elementa
- Uveden je specijalni element koji se ugnježdava, nazvan **keyref** (sa atributom **refer**) i u okviru njega elementi **selector** i **field** (sa atributom **xpath**)
  - **selector**: određuje kontekst stranih ključeva
  - **field(s)**: specificira strani ključ
  - **refer**: daje opseg za reference (ograničenja za ključ)

Primer sheme za knjige koje referišu prema drugim knjigama:

```
<keyref name = "constraintY" refer = "constraintX" >  
  <selector xpath = "book/references" />  
  <field xpath = "@isbn" />  
</keyref>
```





# XML i programerske paradigme





# XML i OO

- Encapsulacija
  - OO sakriva podatke
  - XML čini da podaci budu eksplicitni
- Hijerarhija tipova
  - OO definiše relacije podskup/nadskup
  - XML deli strukturu, pa skupovne relacije nemaju smisla
- Podaci i ponašanje
  - OO ih pakuje zajedno u jednu celinu
  - XML razdvaja podatke od njihove interpretacije



# XML i relacione baze podataka

- **Strukturne razlike**

- Drvo naspram tabele
- Heterogene naspram homogenih
- Opcionalni tipovi naspram striktnog tipiziranja
- Nenormalizovani podaci naspram normalizovanih

- **Neke od sličnosti**

- Logička i fizička nezavisnost podataka
- Deklarativna semantika
- Generički model podataka



# Programerski modeli procesiranja XML-a

- Ogromna korist od XML-a su standardni parseri i standardni API-ji (nezavisni od jezika) za njihovo procesiranje
- DOM je objektno-orjentisana reprezentacija XML drveta parsiranja
  - **DOM objekti** sadrže
    - metode kao što su `getFirstChild`, `getNextSibling`, koje predstavljaju uobičajen način prolaska kroz drvo
    - Takođe mogu da modifikuju samo DOM drvo, tj. da izmene XML, korišćenjem metoda `insertAfter`, itd
- SAX se koristi u situacijama kada nisu potrebni svi podaci
  - **Interfejs za parser** je ključan u ovom pristupu:
    - On poziva funkciju svaki put kada parsira instrukciju procesiranja, element itd.
    - Razvijeni kod može odrediti šta treba raditi u datom slučaju, npr. modifikovati strukturu podataka ili ukloniti deo delove podataka



# XML upiti

- **Upitni jezik** predstavlja alternativni pristup procesiranju XML podataka
  - Definiše se neka vrsta **šablona** koji opisuje prolasku (tj. putanje) od korenog čvora usmerenog grafa koji predstavlja XML
  - Potencijalna korist ovakvog pristupa ogleda se u eksploataciji paralelizma, pogleda, mapiranja shema itd.
  - Kod jezika XML, osnova za ovakve šablone se naziva XPath
    - XPath takođe može deklarirati neka ograničenja na vrednosti koje se traže
    - XPath kao rezultat upita vraća **skup čvorova** koji predstavlja poklapanja



# XPath

- U svom najprostijem obliku, XPath liči na opis putanje u sistemu datoteka:  
`/mypath/subpath/*/morepath`
- Međutim, XPath vraće *skup čvorova* koji predstavljaju XML čvorove (i njihova poddrveta) koji se nalaze na kraju zadate putanje
- XPaths na samom kraju putanje može sadržavati *testove* za čvorove, i tako kreirati filter po tipu čvora metodama `text()`, `processing-instruction()`, `comment()`, `element()`, `attribute()`
- XPath vodi računa o uređenju, može se postaviti upit tako da se vodi računa o uređenju i dobiti odgovor koji poštuje dato uređenje



# Zahvalnica

Delovi materijala ove prezentacije su preuzeti iz:

- Skripte iz predmeta Uvod u veb i internet tehnologije, na Matematičkom fakultetu Univeziteta u Beogradu, autor prof. dr Filip Marić
- Skripte iz predmeta Informatika na Univerzitetu Milano Bicocca, autor dr Mirko Cesarini