

Fakultet tehničkih nauka, DRA, Novi Sad

Predmet:  
Organizacija podataka

# YAML

YAML Ain't Markup Language

# YAML

- YAML Ain't Markup Language (YAML)
  - jezik za serijalizaciju podataka
  - projektovan po standardnim ugrađenim tipovima podataka agilnih programskih jezika
    - Perl, Python, PHP, Ruby i JavaScript
  - razumljiv ljudima
  - za razmenu podataka između različitih jezika
  - zasnovan na Unicode
  - esktenzije
    - .yaml, .yml

# YAML

- specifikacije
  - YAML 1.2 (3rd Edition)
    - <http://www.yaml.org/spec/1.2/spec.html>
  - specifikacija
    - uvod u jezik i prateće koncepte
    - sadrži informacije potrebne za razvoj programa za obradu YAML
- Od verzije 1.2 formalno usaglašenje sa JSON
  - JSON podskup od YAML
  - prethodne verzije su u velikoj meri takođe bile kompatibilne sa JSON

# YAML

- Ciljevi
  - lako čitljiv ljudima
  - portabilnost podataka između programskih jezika
  - podudaranje sa ugrađenim strukturama podataka iz agilnih jezika
  - postojanje konzistentnog modela za podršku generičkim alatima
  - podrška za obradu u jednom prolazu
  - ekspresivnost i proširivost
  - lakoća implementacije i upotrebe
- Upotreba
  - konfiguracione datoteke, datoteke sa logovima, perzistencija objekata

# YAML

- sintaksa
  - koristi vidljive Unicode karaktere
    - strukturne informacije
    - podaci
  - minimizuje broj strukturnih karaktera
    - uvlačenje se koristi za strukturu
    - dvotačka razdvaja parove ključ-vrednost
    - crtice formiraju *bullet* liste

# YAML

- strukture
  - raznovrsne
  - tri osnovne primitivne strukture
    - mapiranja (heševi/rečnici)
      - heš tabele u Perl, rečnici u Python
    - sekvence (nizovi/liste)
      - nizovi u Perl, liste u Python
    - skalari (stringovi/brojevi)
      - atomički tipovi podataka

# YAML

- poređenje sa JSON
  - sličnosti
    - formati za razmenu podataka čitljivi ljudima
  - razlike
    - drugačiji prioriteti
      - JSON: jednostavnost i univerzalnost
      - YAML: čitljivost i podrška za serijalizaciju proizvoljnih ugrađenih struktura podataka
    - JSON se lakše generiše i parsira nauštrb čitljivosti, YAML obrnuto
  - svaka JSON datoteka je i validna YAML datoteka



# YAML

- skalari – string
  - stilovi
    - bez navodnika
    - sa jednostrukim navodnicima
    - sa dvostrukim navodnicima

Primer stringa

```
'String unutar jednostrukih navodnika'
```

```
'Jednostruki navodnik '' u stringu unutar jednostrukih navodnika'
```

```
"String unutar dvostrukih navodnika"
```

# YAML

- skalari – string
  - stilovi
    - navodnici pogodni kada su razmaci na početku ili kraju
    - dvostruki navodnici dozvoljavaju \escape sekvence
    - jednostruki navodnici kada \escape sekvence nisu potrebne
    - specijalni karakteri mogu biti unutar navodnika

```
' 'string' ' '
```

```
'#:!/%.).'
```

```
"\u2603\n"
```

# YAML

- skalari – string
  - u više redova
    - Ako se koriste jednostruki ili dvostruki navodnici
      - naredni redovi moraju biti uvučeni
      - uvlačenje tretira kao jedan razmak

```
"jedan  
dva  
tri"
```

# YAML

- skalari – string
  - u više redova
    - "|" – čuva prelaske u novi red
    - ">" – prelazak u novi red biva zamenjen razmakom

```
|  
  \ / | | \ | |  
  / / | |  | | _
```

```
>
```

```
In a hole in the ground  
there lived a hobbit.
```

# YAML

- skalari – brojevi
  - celi brojevi (različiti brojčani sistemi)
    - decimalni, oktalni, heksadecimalni
  - realni brojevi, eksponencijalni zapis, beskonačno

decimalni

123

oktalni

0173

heksadecimalni

0x7B

realni

12.3

sa eksponentom

1.2e+3

beskonačno

.inf

# YAML

- skalari
  - *null* vrednost
  - logičke vrednosti
  - datum i vreme – po standardu ISO-8601

null vrednost

null

~

logičke vrednosti

true

false

datum i vreme

2014-10-15t18:37:14.10+01:00

2014-10-15

# YAML

- kolekcije
  - može biti sekvenca ili mapiranje
    - sekvenca se obeležava pomoću "- " i razmaka
    - mapiranje se obeležava pomoću ": " i razmaka
      - ključ može biti svaki skalar
      - broj razmaka nakon ":" jedan ili više

sekvenca

```
- Tarzan  
- Cheeta  
- Jane
```

mapiranje

```
Tarzan: 180  
Cheeta: 110  
Jane: 170
```

# YAML

- kolekcije – ugnježdavanje
  - definisanje dosega važenja pomoću uvlačenja
    - BLOCK
    - svaki zapis počinje u sopstvenom redu
    - koristi se jedan ili više razmaka za uvlačenje (ne tab)

mapiranje  
skalara na  
sekvence

```
New Guinea:  
  - Indonesia  
  - Papua New Guinea  
Timor:  
  - East Timor  
  - Indonesia
```

sekvenca  
mapiranja

```
-  
  model: BMW 435i Cabrio  
  hp:    306  
  kmph:  250  
-  
  model: Porsche Cayman S  
  hp:    325  
  kmph:  283
```



# YAML

- kolekcije – ugnježdavanje
  - definisanje dosega važenja pomoću uvlačenja
    - BLOCK
    - svaki zapis počinje u sopstvenom redu
    - koristi se jedan ili više razmaka za uvlačenje (ne tab)

mapiranje  
skalara na  
mapiranja

```
BMW 435i Cabrio:
  turbo: 2
  seats: 4
Porsche Cayman S:
  turbo: 0
  seats: 2
```

sekvenca  
sekvenci

```
-
  - PSV
  - Panathinaikos
  - Estoril
  - Dinamo Moskva
-
  - Napoli
  - Sparta Praha
  - Young Boys
  - Slovan Bratislava
```

# YAML

- kolekcije – ugnježdavanje
  - definisanje dosega važenja bez uvlačenja
    - FLOW
    - koriste se posebne oznake
    - sekvenca
      - unutar "[]" a elementi razdvojeni ","
    - mapiranje
      - unutar "{}" a parovi ključ-vrednost razdvojeni ","

sekvenca

```
[Tarzan, Cheeta, Jane]
```

mapiranje

```
{ Tarzan: 180, Cheeta: 110, Jane: 170 }
```

# YAML

- kolekcije – ugnježdavanje
  - definisanje dosega važenja kombinacijom 2 načina

mapiranje  
skalara na  
sekvence

```
New Guinea: [Indonesia, Papua New Guinea]
Timor:       [Indonesia, East Timor]
```

mapiranje  
skalara na  
mapiranja

```
BMW 435i Cabrio: { turbo: 2, seats: 4 }
Porsche Cayman S: { turbo: 0, seats: 2 }
```

# YAML

- kolekcije – ugnježdavanje
  - moguće ugnježdavanje u više nivoa
  - moguće razne kombinacije

```
a: 1
b:
  -
    baa: 2
    bab: 3
  - bb
  -
    bca: 4
    bcb: 5
```

# YAML

- komentari
  - jednolinijski korišćenjem karaktera "#"

```
# komentar o mapiranju
```

```
New Guinea:
```

- ```
- Indonesia          # komentar o sekvenci  
- Papua New Guinea
```

```
Timor:
```

- ```
- East Timor  
- Indonesia
```

# YAML

- alijasi i sidra
  - ponavljanje
    - prva pojava se označava pomoću sidra
      - "&" praćen stringom (nazivom sidra)
    - naredne pojave se označavaju pomoću alijasa
      - "\*" praćen nazivom sidra
    - primenljivo na bilo koji element

```
vegetable:  
  - potato  
  - &T0 tomato  
fruit:  
  - *T0  
  - pear
```

```
- &car:  
  model: Porsche Cayman S  
  seats: 2  
- bicycle  
- *car
```

# YAML

- dokumenti
  - YAML tekst može obuhvatiti više dokumenata
  - svaki dokument je nezavisan od drugih
  - "---" služi za razdvajanje dokumenata (separator)
    - može se naći na kraju dokumenta
    - može se naći na početku dokumenta

na kraju  
dokumenta

```
doc1a: title
doc1b: body
---
doc2: title
```

na početku  
dokumenta

```
---
doc1a: title
doc1b: body
---
doc2: title
```

# YAML

- dokumenti
  - eksplicitni dokumenti
    - počinju sa separatorom
  - implicitni dokumenti
    - ne počinju sa separatorom



# YAML

- dokumenti
  - separator na početku
    - može sadržati direktive za YAML parser
      - npr. verzija
    - direktive počinju sa "%"

```
--- %YAML 1.2
red: 5
blue: 5
```

# YAML

- tagovi
  - označavaju tip
  - definisani na
    - <http://yaml.org/type/index.html>
  - dve vrste
    - implicitni
    - eksplicitni

implicitni  
tagovi

```
integer: 1  
float: 1.23  
boolean: true
```

eksplicitni  
tagovi

```
integer: !!int "1"  
float: !!float "1.23"  
boolean: !!bool "true"
```

# SnakeYAML

- tagovi i Java tipovi – konverzija

YAML tag	Java tip
!!null	null
!!bool	Boolean
!!int	Integer, Long, BigInteger
!!float	Double
!!binary	String
!!timestamp	java.util.Date, java.sql.Date, java.sql.Timestamp
!!omap, !!pairs	List of Object[]
!!set	Set
!!str	String
!!seq	List
!!map	Map

# YAML

- tagovi
  - implicitno razrešenje taga
    - kod skalara bez oznaka i eksplicitnog taga
    - skalar se poredi sa skupom regularnih izraza
    - u slučaju podudaranja
      - odgovarajući tag se veže za skalar

# SnakeYAML

- SnakeYAML (v 1.11)
  - YAML parser i emiter za programski jezik *Java*
    - kompletan parser za YAML 1.1
    - datoteka *SnakeYAML-all-1.11.zip*
      - za kodiranje u *Java* koristiti *snakeyaml-1.11.jar*
      - sadrži prateću *javadoc* dokumentaciju
  - <https://code.google.com/p/snakeyaml/>

# SnakeYAML

- podrazumevane implementacije kolekcija
  - List
    - ArrayList
  - Map
    - LinkedHashMap
      - poredak je implicitno definisan
  - dozvoljeno je definisanje drugih podrazumevanih implementacija

# Primer 1

- Napisati *Java* program koji
  - parsira string u kojem je *YAML* sekvenca
  - parsira string u kojem je *YAML* mapa
  - Parsira sadržaj datoteke koja sadrži više *YAML* dokumenata
- Zadatak uraditi koristeći
  - *SnakeYAML* biblioteku

## Primer 2

- Napisati *Java* program koji
  - parsira *YAML* datoteku *invoice.yaml* i mapira sadržaj na odgovarajuće *Java* objekte
  - vrši izmenu adrese na fakturi
  - zapisuje izmenjenu verziju fakture u *invoice\_mod.yaml*
- Zadatak uraditi koristeći
  - *SnakeYAML* biblioteku



# Zadatak 1

- Napisati *Java* program koji
  - učitava sadržaj *invoice.yaml* u *Java* objekte
  - svaki iznos u fakturi preračunava u odabranu valutu
    - kurs učitava iz *exchange.yaml* (pripremiti test primer)
      - datoteka sadrži više parova tipa ***valuta: kurs***
    - program nudi spisak svih valuta iz *exchange.yaml* a korisnik bira jednu od tih valuta
  - zapisuje rezultat u *invoice\_X.yaml*, gde umesto *X* stoji naziv valute

# Dodatni zadatak 1

- Napisati *Java* program koji
  - učitava sadržaj *invoice.yaml* u *Java* objekte
  - za svaki kupljeni proizvod zapisuje posebnu fakturu u datoteku *invoice\_X.yaml*
    - gde prva izlazna datoteka umesto *X* u nazivu ima 1, a svaka naredna za jedan veću vrednost od prethodne
    - broj svake nove fakture raste za jedan počevši od broja fakture iz *invoice.yaml*
    - ukupni porez se deli na nove pojedinačne fakture srazmerno njihovim pojedinačnim iznosima

# Dodatni zadatak 2

- Napisati *Java* program koji
  - učitava sadržaj *log.yaml* u *Java* objekte
  - zapisuje podatke u dve izlazne datoteke
    - *log\_warn.yaml* koja sadrži poruke sa upozorenjem
      - dokumenti sa ključem *Warning*
    - *log\_error.yaml* koja sadrži poruke sa opisom greške
      - dokumenti sa ključevima *Fatal* i *Stack*