LYNDA

BEYOND THE BASICS

(2) CHAPTER

-u projektu

$ rails . - updatea projekt i odulive se što "sbronit"

-još bolji za update - stvorit novi projekt i mirno kopirat fileove

$ rake -T - sve što rake može

$ rake rails:freeze:gems - kopira cijeli RAILS u /vendor folder i tako "fiksira" aplikaciju

$ rake rails:unfreeze

$ ri find - querya rails dokumentaciju

(3) CHAPTER — rails i ruby u terminalu

- $ ruby -e 'puts 1+1' → OVO JE U COMMAND LINEU
  ↳ 2
- $ ruby ~/Desktop ... path../file.rb
  ↳ vraća output iz datoteke
- $ irb → interactive ruby

```
for name in first_array
  puts name
end
```

(first_array = ['Kevin', 'Bob'])

- ako copy-pasteamo u IRB - paziti na tabove
- 1 + 1
  ↳ 2
- _ + 1 → OVO JE ZADNJA VRAĆENA VRIJEDNOST
  ↳ 3
- _ * 3
  ↳ 9
- rails c — rails konzola. Za razliku od IRB-a ima pristup bazi i svemu definiranom u rails aplikaciji
  ↳ my_array = ['ime', 5, [4,5,6]]
  - y my_array — outputa yaml od my_array

Arrays

(4) CHAPTER – RUBY tehnike sa arrays

IRB ARRAYS

↳ word_array = %w (Kevin Bob Sally) → pravi string array od riječi (nema zareza)

↳ ["Kevin", "Bob", "Sally"]

↳ array1.include?(6) → DALI ARRAY SADRŽI ODREĐENI BROJ (ILI BILO KOJI DRUGI OBJEKT)

↳ TRUE ILI FALSE

↳ array1.delete(2) → BRIŠE BROJ 2 IZ ARRAYA, A NE OBJEKT S POZICIJE 2

↳ vraća nil, ako broj (objekt) nije u arrayu

↳ inače vraća izbrisani objekt

↳ x = array1.delete(3) → IZBRISANI OBJEKT (BROJ 3) JE PREŠO U X

→ AKO OBJEKT NE POSTOJI U ARRAYU X ĆE BIT NIL

x

↳ =>3 x=3

↳ array1.delete_at(3) → ovo briše treći element u nizu

↳ array1.empty? → ispituje jeli array prazan

→ TRUE (PRAZAN) ILI FALSE (IMA NEŠTO U ARRAYU)

↳ array1.uniq → vraća jedinstvene vrijednosti iz arraya (duplikate pr. duple brojeve) izbacuje

↳ array1.uniq! → jedinstvene vrijednosti iz arraya sprema u isti array

→ radi destruktivno

↳ array 2 = array 1.dup → duplicira array 1 i onda tay duplikat postaje array 2

[1,2,3] [4,5]

↳ array 1 + array 2 → spoji samo 2 niza

↳ [1,2,3,4,5]

[1,2,3] [3,4,5]

↳ array 1 | array 2 → unija → isto kao (array1 + array2).unija

↳ [1,2,3,4,5]

[1,2,3] [2,3,4]

↳ array 1 & array 2 → maka elemente koji su i u arrayu 1 i u arrayu 2

↳ [2,3]

[1,2,3] [3,4]

↳ array 2 - array 1 → miče iz arraya 2 sve što ima i u arrayu 1

↳ [1,2]

Hashes

HASHES (IRB) → HASH JE KEY-VALUE PAR

my_hash = { "day" => "Monday", "name" => "Bruno", "ruby" => "Rox" }

KEY VALUE

↳ my_hash.include? ("day") OVO TRAŽI PO KEYEVIMA → dali hash sadrži odredeni key (ne value)

↳ TRUE

ISTO

↳ my_hash.has_key? ("day")

↳ TRUE

↳ my_hash.has_value? ("Monday") → dali hash sadrži odredeni VALUE

↳ TRUE

↳ my_hash.empty?

→ TRUE ili FALSE

↳ my_hash.delete("subj")
⇒ "Ror"

→ briše key-value par i vraća value!
→ vraća nil ako nema key-value para!

KEY
↳ my_hash["name"]
⇒ "Bruno" → VRAĆA VALUE

VALUE
↳ my_hash.key("Bruno") → vraća key od valuea
=> "name"
KEY

vraća spoj
↳ my_hash.merge(new_hash) → ~~spaja~~ 2 hasha. New_hash ide na kraj my_hasha
→ my_hash ostaje isti kao prije (nije destruktivna naredba)

↳ my_hash.merge!(new_hash) → sad je destruktivno

VRAĆA
↳ my_hash.keys ⇒ ["day", "name", "subj"] → KEYEVE
↳ my_hash.values ⇒ ["Monday", "Bruno", "ror"] → VALUES

CODE BLOCK .EACH

- 9.times { puts "Hello" }
  - ↳ CODE BLOCK JE UNUTAR {} ZAGRADA
  - ↳ RAZLIČITO OD HASHEVA !

- objects.each { |obj| block } → SINGLE LINE
- objects.each do |obj|
  line 1
  line 2
  end

  } MULTI LINE

↳ OVO JE ISTO KAO

- for obj in objects → radi isto
  line 1
  line 2
  end

→ names = ['kevin', 'Larry'] indexi od hasha [0, 1, ..]

- names.each_index { |index| puts "Index #{index}" }

- names.each_with_index { |name, index| puts '...' }
  2 VARIJABLE

↳ person = {'ime' => 'Ivo', 'prezime' => 'Sutić', 'job' => 'programer'}

- person.each_key { |k| puts ... }
  YIELDA SE KEY IZ HASH-a

- person.each_value { |v| puts... }
  YIELDA SE VALUE IZ HASH-a

person.each_pair { |k, v| puts ... }

YIELDA i KEY i VALUE IZ HASH-a.

FIND -CODE BLOCK { RUBY FIND, NE RAILS FIND)
↑
ACTIVE RECORD.

nums = [1, 2, 3, 4, 5]

- nums.find { |n| n>3 }
  → 4

BOOLEAN TEST } ISTO

→ VRAĆA SAMO JEDNU (PRVU) VRIJEDNOST

ILI ISTO

- nums.detect { |n| n>3 }
  → 4

- nums.find_all { |n| n>1 && n<5 }
  → [2,3,4]

→ vraća SVE vrijednosti koji odgovaraju boolean izrazu

ISTO

- nums.select { |n| n>1 }
  → [2,3,4,5]

BOOLEAN

↳ nums.any? { |n| n ... }
→ TRUE ILI FALSE

→ JELI JEDAN ELEMENT IZ ARRAYA ZADOVOLJAVA BOOLEAN TEST

↳ nums.all? { |n| n ... }
→ TRUE ILI FALSE

→ JELI SVI ~~JEDAN~~ ELEMENTI ZADOVOLJAVAJU?

↳ nums.delete_if {|n| n<3} → BRIŠE SVE KOJI ZADOVOLJAVAJU

HASHEVI – ne rade sve naredbe i na arrays i na hashs!

person = {'ime' ⇒ 'Bruno', 'prezime' ⇒ 'Sutić'}

↳ person.select {|k,v| k=="last"}

YIELDA I KEY I VALUE

→ person.delete_if {|k,v| k==...}

MERGE – CODE BLOCK

OVO IDE KAD JE KONFLIKT!?

• hash.merge(other_hash) {|key, oldval, newval| block}

↳ person.merge(new_person) {|key, old, new| old} → zadržava stanu vrijednost

ili

.... old + " " + new} → radi operaciju

COLLECT, MAP -code blocks

Collect, map

- rezultat collecta je uvijek array! Uvijek vraća isti broj objekata!

- nums = [1, 2, 3, 4, 5]
- nums.collect {|n| n+1}
  ↳ [2, 3, 4, 5, 6]
- nums.map {|n| n*20}
  [20, 40, 60, 80, 100]

COLLECT I MAP SU SINONIMI → NEDESTRUKTIVNI

↳ nums.map! {|n| n+1}
→ nums = [2, 3, 4, 5, 6]

HASHES

KAD JE HASH ONDA YIELDA 2 VRIJEDNOSTI KEY I VALUE

person.collect {|k,v| v} → VRAĆA ARRAY UVIJEK

Inject, sort

INJECT - CODE BLOCKS

memo =

(nums = [20, 40, 60, 80, 100])

- nums.inject { |memo, n| memo + n }
  => 300
  ↳ u svakom koraku se memo puni

- nums.inject(200) { |memo, n| memo + n }
  ↑ INICIJALNA VRIJEDNOST METODA
  => 500

SORT - CODE BLOCKS

- menu = ["Home", "Products", ... , "Contact us"]
- menu.sort → sortira po prvom slovu abecedno
  ↓ ISTO
- menu.sort { |item1, item2| item1 <=> item2 }
  USPOREDIVA i VRAĆA VEĆI OBJEKT

- menu.sort_by { |item| item.length }

CUSTOM CODE BLOCKS

```
def metoda
    puts "Nesto prije"
    yield
    puts "Nesto poslje"
end

metoda { puts "Ovo u sredini" }
```

OVO SE YIELDA

```
def yield_metoda
    puts "Start"
    x=100
    yield(x)
    puts "Kraj"
end

yield_metoda {|val| puts val * 100}
```

↳
```
Start
10000
Kraj
```

CONTROL FLOW

TERNARY OPERATOR

↳ boolean ? result1 : result2

PRIMJER

TERNARY OPERATOR

```
↳ x = 1
result = x%2 == 0 ? "Even" : "Odd"
```

↳ AKO JE MODUL 0 ONDA ↑ ELSE ↑

Errors & exceptions

CASE OPERATOR

```
case
when x == 1
    puts "one"
when x == 2
    puts "two"
when x == 3
    puts "three"
else
    puts "more"
end
```

ili

OVDJE MOŽE 1 'THEN'

```
case
when x == 1 puts "one"
when x == 2 puts "two"
when x == 3 puts "three"
else puts "more"
end
```

OVO NE IDE

NEMA : NEGO SAMO THEN

ili ako se ispituje samo varijabla x

```
case x
when 1
    puts "one"
when 2
    puts "two"
when 3
    puts "three"
else
    puts "more"
end
```

ili

```
case x
when 1 then puts "one"
when 2 then puts "two"
when 3 then puts "three"
else puts "more"
end
```

ERRORS & EXCEPTIONS

IRB

- x = 100/0
  ↳ division by 0
- x = 100/0 rescue x = 200
  ↳ 200

```
begin
    code to attempt
    ...
rescue
    code if it fails
end
```

npr.

```
def initialize(value)
    begin
        @answer = 100 / value
    rescue
        @answer = 20
    end
```

RAISANJE GREŠKE

```
- class Grumpy
    def initialize
      puts "Dont mess with me"
      raise
    end
  end

- var = Grumpy.new
  ↳ Don't mess with me
    Runtime error
```

IZBACI GREŠKU
- RUNTIME ERROR PO DEFAULTU

SPAŠAVANJE POSEBNE GREŠKE

```
def initialize (value)
    begin
      @answer = 100/value
    rescue ZeroDivisionError
      @answer = 20
    end
end
```

SPAŠAVA SAMO OVU GREŠKU

LISTA ERRORA

ArgumentError — krivi broj argumenata
NameError
NoMethodError
RuntimeError
SyntaxError
TypeError
ZeroDivisionError

```
rescue ZeroDivisionError => e
    puts "You can't divide by 0!!!"
    puts "Error was: " + e
    :
end
```

→ STAVLJA GREŠKU U varijablu e

5 CHAPTER