Ruby - dragi kamen, španska serija ili programski jezik?

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Vladana Đorđević, Aleksandra Jovičić, Jovana Nikolić, Tijana Tošev
vladana1711@gmail.com, jovicicaleksandra96ajs@gmail.com,
jovananiki7@gmail.com, tijana.tosev@gmail.com

6. april 2019

Sažetak

U ovom tekstu je ukratko prikazana osnovna forma seminarskog rada. Obratite pažnju da je pored ove .pdf datoteke, u prilogu i odgovarajuća .tex datoteka, kao i .bib datoteka korišćena za generisanje literature.

Sadržaj

1 Uvod **2**

2 Istorijski razvoj, uticaj i namena **2**

3 Osnovne osobine i koncepti **4**

4 Specifičnosti programskog jezika **7**

5 Razvojna okruženja **9**

6 Instalacija i uputstvo za pokretanje programa na Linux/Windows operativnim sistemima **10**

7 Primer jednostavnog kôda **11**

8 Zaključak **12**

Literatura **12**

1 Uvod

U mnoštvu različitih programskih jezika šta je to što Ruby izdvaja od ostalih i zašto treba posvetiti pažnju učenju istog? Zanimljivo je da je učenje Ruby-ja lako, kako iskusnim programerima, tako i početnicima, jer iako ima korene u mnogim poznatim jezicima, njegova sintaksa je prirodna i u velikoj meri se poklapa sa jezikom koji ljudi koriste. Moto Ruby zajednice je da članovi budu ljubazni, ugledajući se na tvorca jezika koji je upravo poznat po velikoj ljubaznosti. Stoga, pri učenju jezika možete da računate na pomoć Ruby programera. Ovim radom ćemo Vam predstaviti nastanak i razvoj jezika, namenu, osnovne osobine i specifičnosti kao i razvojna okruženja u kojima se on koristi, uputstva za instalaciju i primere kodova. Cilj nam je da Vam prikažemo mogućnosti koje Ruby pruža i da Vas zainteresujemo da ga naučite.

2 Istorijski razvoj, uticaj i namena

Ideju za stvaranje Ruby-ja kao novog objektno-orijentisanog (OO) skript jezika, Yukihiro "Matz" Matsumoto, njegov tvorac i softverski inženjer iz Japana, dobio je 23. februara 1993. godine u razgovoru sa kolegom. Pre stvaranja Ruby-ja bio je upoznat sa mnogo drugih programskih jezika, ali njima nije bio zadovoljan. Smatrao je da su bili ružniji, teži, kompleksniji ili jednostavniji nego što je očekivao, pa je želeo je da napravi novi OO skript jezik koji je prirodan, a ujedno i moćan. Matz je želeo da jeziku da ime po dragom kamenu, pod uticajem imena jezika Perl, stoga mu je dao ime Ruby (rubin) jer je to dragi kamen koji odgovara mesecu rođenja jednog njegovog kolege.[7][6]

2.1 Nastanak i istorijski razvoj

Tokom razvoja Ruby-ja Matz se fokusirao da programiranje učini bržim i lakšim, stoga je sve karakteristike jezika dizajnirao da rade onako kako programeri to očekuju. Upravo to je ono što Ruby čini elegantnim, lakim za korišćenje i zadovoljstvom za programiranje.

Ruby je postao veoma popularan u Japanu 1990-ih godina, međutim, svetsko priznanje dobio je tek 2006. godine pojavljivanjem, danas veoma popularnog, okruženja Ruby on Rails. U decembru 1995. godine objavljen je Ruby 0.95, a u decembru 1996. Ruby 1.0. Svaka verzija se objavljuje na katolički Božić. Najnovija verzija, Ruby 2.6, izbačena je na katolički Božić 2018. godine, dok se verzija 3.0 očekuje 2020. godine.[7][6]

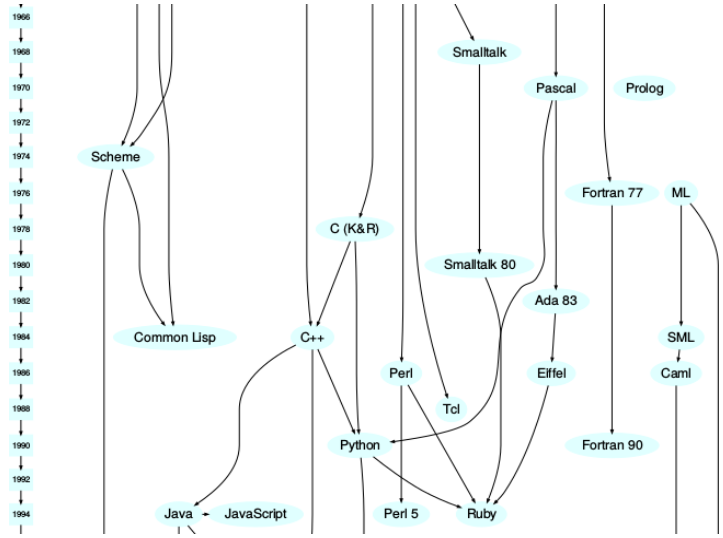
2.2 Mesto u razvojnom stablu i uticaji drugih programskih jezika

Kao dugogodišnji ljubitelj OO programiranja, Matz je tražio jezik koji će zadovoljiti njegove kriterijume. Bio je upoznat sa jezicima Perl 4 i Python, ali mu se nisu svideli. Python nije smatrao pravim OO jezikom, jer su OO karakteristike delovale kao dodatak jeziku. Tražio je OO skript jezik, jer mu je izgledao obećavajuće, ali ga nije našao, stoga je odlučio da ga sam napravi. Spojio je delove svojih omiljenih jezika – Perl, Smalltalk, Eiffel, Ada, Lisp i Python. Položaj jezika u današnjem vremenu i njegovu ravnopravnost sa nekim od poznatih programskih jezika možemo videti iz tabele 1.[6]

Tabela 1: GitHub statistike i godine nastanka programskih jezika

Jezik	Aktivni repozitorijumi	Prosečan broj posmatrača po repozitorijumu	Godina nastanka
C#	56 062	4.74	2000.
C++	86 505	5.77	1983.
Ruby	132 848	5.92	1995.
Python	164.852	5.72	1991.
Java	222 852	6.24	1995.

Kao osnovu za Ruby, Matz je koristio programski jezik Lisp. Karakteristike koje su mu se svidele iz drugih jezika je dodavao. OO paradigma je inspirisana jezicima Smalltalk i Perl. Sintaksa je inspirisana Perl-om, a semantika Smalltalk-om, dok je fleksibilnost uzeta iz Python-a. Na slici 1 prikazan je položaj Ruby-ja u razvojnom stablu.[7][1]



Slika 1: Razvojno stablo

2.3 Namena, svrha i mogućnosti

Ruby spada u skript jezike i u većini slučajeva se koristi za uopštenu namenu. Važno je napomenuti da je Ruby potpuno besplatan, kao i da ga možemo preuzeti, modifikovati i distribuirati.

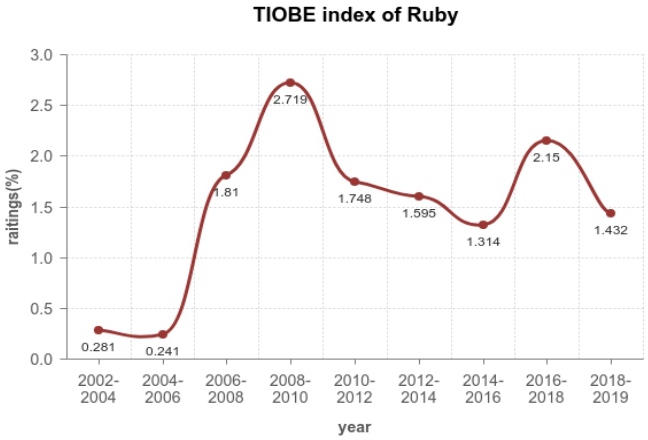
Možemo ga koristiti za pisanje veb aplikacija i veb servera, za rad sa bazama podataka, automatizaciju poslova, kreiranje mobilnih/desktop aplikacija i parsiranje. Neki programeri ga koriste i za pisanje programa iz oblasti veštačke inteligencije i mašinskog učenja. Takođe, postoji biblioteka BioRuby koja se koristi u oblasti biologije i medicine.

Poznati projekti koji su implementirani u Ruby-ju su:

- Dicourse - open source platforma za komunikaciju

- Ruby on Rails - razvojno okruženje za veb
- GitHub - servis za kontrolu verzije
- Airbnb - popularna rental-platforma
- Hulu - emitovanje televizijskih serija
- Sinatra - oblasno-specifičan jezik za pisanje veb aplikacija

U 2002. godini kada se pojavio Ruby je zauzimao 39. poziciju na TIOBE listi indeksa programskih jezika. Popularnost jezika se menjala kroz godine što možemo zaključiti sa slike 2, a trenutno se nalazi na 15. poziciji na TIOBE listi.



Slika 2: TIOBE indeks za Ruby od 2002. godine do danas

3 Osnovne osobine i koncepti

U ovom poglavlju upoznaćemo se sa osnovama jezika, a to počinjemo primerom 1.

```
0  def hello
     # puts naredba moze sadrzati i zagrade
2    puts "Hello Matf"
   end
```

Listing 1: Primer ispisa

Ruby poseduje popularne mehanizme poput regularnih izraza, obrade izuzetaka, sakupljače otpadaka i mogućnost pisanja automatske dokumentacije. U nastavku ćemo ukratko opisati neke od najkorišćenijih koncepata jezika.

3.1 String

Već u prvom našem programu imamo primer upotrebe Stringa. Stringovi su jedan od podrazumevanih tipova većine jezika. Nadovezivanje stringova se može izvršiti pomoću operatora "+" ili "<<" koji ne menja postojeći String objekat nego dolazi do kreiranja novog. Ukoliko

pokušamo nadovezati stringove pomoću "," tip koji smo dobili zapravo predstavlja objekat klase Array.

Karakterima u stringu možemo pristupati korišćenjem notacije s[i,j], gde je s neki string a i i j indeksi prvog i poslednjeg karaktera podniske koja će biti izdvojena. Neke od ponuđenih klasičnih metoda su *length*(vraća dužinu stringa), *reverse*(obrće redosled čarova u stringu), *swapcase*(menja mala slova u velika i obrnuto), *insert(position, string)* (dodaje nisku na zadatu poziciju u stringu).

3.2 Array

Niz je sekvencijalna kolekcija stavki u kojima se svaka stavka može indeksirati. U Ruby-ju jedan niz može da sadrži stavke mešovitih tipova podataka, kao što su stringovi, celi brojevi pa čak i pozivi metoda koje vraćaju neku vrednost. Indeksiranje kreće od nule. Postoji i mogućnost kreiranja matrica kao niza nizova. Moguće je duboko kopiranje gde se promenom jednog niza menja i drugi, novonastali, i plitko gde menjanje jednog ne utiče na drugi niz[2]. Prvo se vrši pomoću operatora dodele, a drugo pomoću metoda *clone*.

Niz može da sadrži i *nil* objekte ukoliko neki od članova nije inicijalizovan ili je izvan definisanog opsega. Na primer, za niz n=[1,2,3], n[3] nam daje vrednost *nil* iz klase *nilClass*. Neke metode koje možemo pozivati nad nizovima su *delete(object)*(briše navedeni element niza), *sort*(sortira niz), *clean*(briše sve elemente), *compact*(vraća niz sa uklonjenim *nil* vrednostima).

3.3 Hash

Mogućnost da se umesto brojevnih vrednosti niz indeksira vrednostima drugog tipa je u Ruby-ju omogućena uvođenjem klase Hash. Hash je sličan strukturi koja je poznata i kao mapa u nekim drugim jezicima, gde indeks predstvlja ključ, a pridruženi član indeksa vrednost. Ukoliko za isti ključ definišemo različite vrednosti, ključ će imati vrednost poslednje definisane vrednosti za taj ključ.

Kopiranje vrednosti je analogno kopiranju kod nizova. Redosled elemenata u hešu varira u skladu sa verzijom Ruby-ja koja se koristi. U Ruby 1.8 heš se obično skladišti u leksičkom redosledu ključeva, a u Ruby 1.9 heš se čuva u redosledu u kojem su definisani članovi.[2]

3.4 IO

Ruby obezbeđuje klase posvećene rukovanju ulaza i izlaza pomoću klase IO. IO klasa omogućava upravljanje ulazno/izlaznim tokovima bajtova, odnosno otvaranje, zatvaranje, čitanje i pisanje. Klasa File je potklasa klase IO koja poseduje mogućnost navođenja opcija za otvaranje fajla poput r(čitanje), w(pisanje), a(pisanje na kraju fajla). Naredna linija koda otvara tok podataka iz zadatog fajla, čita podatke i ispisuje ih.

```
0 IO.foreach("testfile.txt") {|line| print( line ) }
```

Možemo primetiti da Ruby ima mehanizam koji brine o otvaranju i zatvaranju datoteka umesto korisnika.

3.5 Kontrola toka programa i funkcije

Ruby omogućava brojne načine za pisanje petlji. Imajući u vidu da čitalac poznaje namenu i način funkcionisanja istih, u nastavku predstavljamo moguće sintakse. Takođe, podržan je i koncept grananja naredbama if-then-else.

U Ruby-ju sve metode uvek vraćaju vrednost s tim da korisnik nije dužan da ih upotrebi. Kada nije navedena povratna vrednost, metode vraćaju rezultat poslednjeg izraza. Prvi navedeni kod u primeru 2 vraća vrednost 3, dok je u drugom eksplicitno zadata povratna vrednost.

```
0  def method1                    def method2
     a = 1                          a = 1
2    b = 2                          b = 2
     c = a + b                      c = a + b
4  end                                return b
                                  end
```

Listing 2: Prvi primer koda

Neke funkcije mogu imati više povratnih vrednosti i tada je povratna vrednost tipa Array. Objekat tipa Hash takođe može biti povratna vrednost. Argumenti funkcije mogu imati svoje podrazumevane vrednosti navođenjem istih na sledeći način:

```
0  funkcija( a=10, b=20, c=100)
```

Značajna novina je da neka funkcija može da ima neodređeni broj argumenata što se postiže dodavanjem zvezdice poslednjem argumentu:

```
0  funkcija( a=10, b=20, c=100, *d)
```

Pri pozivu funkcije svi argumenti definisani posle osnovno-definisanih argumenata se smeštaju u niz i u ovom slučaju predstavljaju argument d.

3.6 Podržane paradigme

Ruby je prvenstveno predstavnik OO paradigme koja je, za razliku od Jave potpuna, što znači da i primitivni tipovi predstavljaju objekat. Takođe, gotovo sve operacije predstavljaju metode klasa. Kao primer navodimo operaciju množenja "*" u izrazu 5*2=10, koja je metoda objekta 5. Integer 2 je parametar metode, a 10 povratna vrednost.

Iako je OO paradigma glavna odlika Ruby-ja, on podržava i pisanje programa u funkcionalnom i imperativnom stilu. Funkcionalna paradigma se odlikuje prvenstveno mogućnošću pisanja lambda izraza. Sintaksa jednog lamda izraza je sledeca:

```
0  add = lambda { |x,y| x + y }
```

Dati metod mozemo pozvati sa *add.call(1,2)*.

Referentna transparentnost gde neku funkciju možemo zameniti sa rezultatom i kompozicija funkcija gde se dve ili više funkcija kombinuje u jednu su još neki od funkcionalnih koncepata koji se mogu koristiti.

Ruby obezbeđuje dobar skup funkcija višeg reda, naročito u klasama Array i Hash, kao što su *each*(za svaki), *select*(izabrati), *push*(dodati na kraj), *pop*(skinuti sa početka), *delete_if*(izbrisati ako), *map*(mapiranje), *reduce*(sažimanje), *merge*(spajanje), *replace*(zamena). Korišćenje ovih

metoda može pomoći da veće probleme brzo svedemo na manje, izbegavajući veći broj imperativnih metoda. Dobijamo čistiji i modularniji kod. U nastavku je dat primer koda 3 koji ispisuje kvadrate članova niza na klasičan imperativni i elegantan funkcionalni način.

```
# imperativni nacin
i = 0
while i < array.length do
  puts array[i]**2
# funkcionalni nacin
array.collect { | element | element ** 2 }
```

Listing 3: Prvi primer koda

4 Specifičnosti programskog jezika

Ruby poseduje neke osobine koje ga izdvajaju od ostalih programskih jezika. U nastavku pomenućemo neke od njih.

4.1 Oznake promenljivih i simboli

Promenljive se razlikuju po početnom znaku u imenu. Znakom $ se predstavljaju globalne promenljive, promenljive instance počinju znakom @, dok klasne promenljive počinju @@.

```
# $x, $iterator, $1
```

Promenljive koje počinju znakom @ su promenljive instance. One se obično koriste za definisanje atributa.

```
# @x, @y, @width, @height
```

Promenljive koje počinju sa @@ označava da su to klasne promenljive.

```
# @@x, @@z, @@counter
```

Simboli su promenljive koje počinju sa ":" i zatim sledi njihov naziv. Nemaju vrednosti ili objekte kao što to imaju promenljive. Koriste se kao konzistentna imena u okviru koda. Razlika je u tome što umesto simbola možemo da navedemo stringove u okviru koda i za svaki navedeni string će se kreirati novi objekat koji će se čuvati u memoriji. Ako koristimo simbole biće inicijalizovani samo jednom. Simboli omogućavaju i čistiji kod. Moguće je konvertovati string u simbol pomoću odgovarajućih metoda, i obrnuto.[5]

Dva stringa imaju istu vrednost, ali će se posmatrati kao dva različita objekta, dok će u slučaju dva različita simbola oni uvek imati različite vrednosti. Poređenje dva stringa je sporije nego kad poredimo dva simbola i zato se simboli generalno koriste kao ključevi heša.

4.2 Blokovi, klasa Proc i lambde

Svaki kod koji je okružen zagradama "{}" ili obuhvaćen ključnim rečima *do* i *end* označava blok. Ideja blokova je da grupišu naredbe koje želimo da se kroz program prosleđuju zajedno. Argumenti blokova su skup promenljivih koje koristimo u bloku i navode se na početku.

```
# {|x, y| x + y}
```

Blokovi su sintaksne strukture u Ruby-ju i ne smatraju se objektima. Ipak, moguće je kreirati objekat koji predstavlja blok. U zavisnosti kako je taj objekat kreiran razlikujemo *procs* i *lambde*. Klasa Proc omogućava da pozovemo metodu nad njenim instancama (*procs*) i da ih dodelimo promenljivama. Takođe, procs mogu biti i povratne vrendosti metoda. Lambde su Proc objekti koji se više ponašaju kao metode, dok se sami Proc objekti više ponašaju kao blokovi. Razlika se ogleda u tome što lambde proveravaju broj argumenata i drugačiji način tretiraju ključnu reč *return*.

4.3 Moduli i mixins

Modul predstavlja grupisane metode, konstante i klasne promenljive. Definisani su kao klase, ali je razlika u tome što oni ne mogu da budu instancirani, kao i da ne mogu biti nasleđeni. To znači da ne postoji višestruko nasleđivanje.

Mixins predstavljaju više pomešanih modula (*eng. mixed in*) i način da se prevaziđe nepostojanje višestrukog nasleđivanja. Omogućavaju kontrolisan način dodavanja funkcionalnosti klasama. Moduli ne mogu da imaju instance zato što nisu klase, ali možemo da uključimo module u okviru neke klase. Na taj način svi metodi modula su na raspolaganju kao metodi u datoj klasi. Zapravo, pomešani moduli se ponašaju kao superklase.[1]

Primerom 4 pokazujemo upotrebu *mixin Comparable* koji može da se koristi za dodavanje operatora poređenja (<, >, >=, <=, ==) i metode *between?(min,max)* nekoj klasi. Kako bi ovo radilo *Comparable* pretpostavlja da klasa koja ga koristi ima definisan operator "<=>". Konačno, potrebno je da definisemo jedan metod koji implementira operator "<=>" i uključimo *mixin Comparable* kako bismo dobili dodatnih šest metoda poređenja besplatno.

```
0  class Primer
     include Comparable
2    attr :str
     def <=>(other)
4      str.size <=> other.str.size
     end
6    def initialize(str)
       @str = str
8    end
     def inspect
10     @str
     end
12 end

14 s1 = Primer.new("pas")
   s2 = Primer.new("rubin")
16 s3 = Primer.new("tacka")

18 puts s1 < s2         # => true
   puts s3.between?(s1, s2)  # => true
```

Listing 4: Prvi primer koda

4.4 Transparentnost i fleksibilnost

U koliko je potrebno odratiti složen posao u sto manje linija koda i na jednostavnoji način pogodno je koristiti Ruby. Jedna od odlika Ruby-ja je njegova transparentnost. Pod tim se podrazumeva da za rešenja naših problema ne moramo imati mnogo dokumentacije kako bismo odradili

neke jednostavne stvari. Pomoću ovog programskog jezika možemo da direktno izrazimo ideje i dizajne rešenja problema, i to na elegantan način. Sve ovo znači da ćemo kucati brže, kao i da su naši kodovi čitljiviji i lakše održivi.

```
require 'net/http'
Net::HTTP.start ('www.ruby-lang.org',80) do |http|
  print (http.get('/en/about/license.txt').body)
end
```

5 Razvojna okruženja

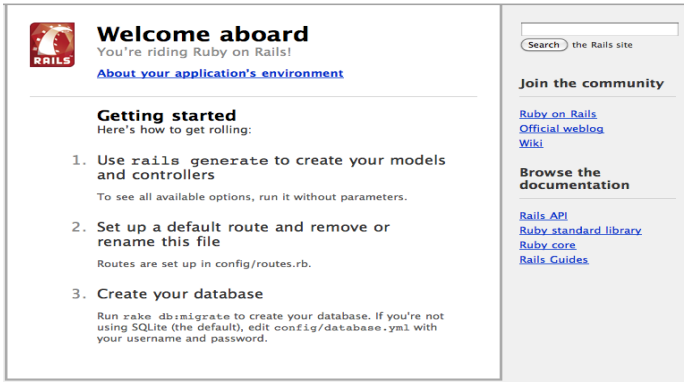
Ruby ima mnoštvo razvojnih okruženja, posebno u domenu razvoja veb aplikacija. Najpoznatija okruženja za razvoj veb aplikacija su svakako Ruby on Rails, koji je najpopularniji u ovom domenu, zatim Sinatra, Padrino, Hanami (ex. Lotus), NYNY, Cuba, Crepe, Grape, Nancy, Hobbit, Scorched, Rack.

5.1 Ruby on Rails

Ruby on Rails je jedno od najpopularnijih razvojnih okruženja za razvoj veb aplikacija. Zapravo, njegovim pojavljivanje, Ruby je postao svetski poznat jezik. Napravljeno je da učini programiranje veb aplikacija lakšim praveći pretpostavke o tome šta je svakom programeru potrebno da bi počeo. Omogućava da se napiše manje koda, a da se pritom ostvari više nego u mnogim drugim jezicima i okruženjima. Iskusni Rails programeri kažu da čini razvoj veb aplikacija zabavnijim.[4]

Rails pravi pretpostavke o tome da postoji “najbolji” način da se nešto uradi i dizajniran je tako da podstiče programera da koristi taj način, a u nekim slučajevima i da ga obeshrabri da koristi alternativne načine. Rails ima dva osnovna principa - Đogovor umesto konfiguracije”(eng. Convention over configuration) i ”Ne ponavljaj se”(eng. Don’t repeat yourself). To znači da okruženje obezbeđuje šablone i očekuje se da će ih programer koristiti bez izmena, kao i da će otklanjati kod koji se ponavlja. Ovaj pristup razvoju veb aplikacija funkcioniše veoma dobro kod većine ljudi, međutim, potrebno je napraviti kompromis. Da bi se obezbedio sistem koji funkcioniše bez konfiguracija i koda koji se ponavlja, Rails u velikoj meri koristi dinamičke karakteristike Ruby-ja. Klase se konstruišu pri izvršavanju a metode se dinamički generišu. Programeri koji nauče da pravilno koriste ovo okruženje verovatno će otkriti ogroman porast produktivnosti. Međutim, ukoliko su uporni u nameri da donesu stare navike iz drugih jezika u okruženje Rails i pokušavaju da koriste šablone koje su naučili negde drugde, mogu imati lošije iskustvo.[4][9]

Mnoge poznate aplikacije napisane u Ruby-ju su napravljene upravo pomoću Ruby on Rails okruženja koje je prvi put objavljeno 25.10.2004. Poslednja verzija 5.2.2.1 objavljena je u martu 2019. godine. Rails je softver otvorenog koda, čijem razvoju je doprinelo više od 4500 ljudi, a preuziman je skoro 166 miliona puta. [8]



Slika 3: Razvojno okruženje - Ruby on Rails

6 Instalacija i uputstvo za pokretanje programa na Linux/Windows operativnim sistemima

Instalacija i izvršavanje standardne implementacije programskog jezika Ruby je podržana na različitim operativnim sistemima. U narednom delu prikazujemo slučajeve kod operativnih sistema Linux i Windows.

6.1 Instalacija

Najjednostavniji način za instalaciju Ruby programskog jezika na Linux operativnom sistemu(distribucija Debian i Ubuntu) je korišćenjem *apt* menadžer paketa. Prvo, ažuriramo listu dostupnih paketa:

```
$ sudo apt update
```

Instaliramo Ruby naredbom:

```
$ sudo apt install ruby-full
```

Radi provere da li je instalacija uspela koristimo sledeću naredbu koja nam prikazuje verziju Ruby programskog jezika:

```
$ ruby --version
```

Treba da se dobije izlaz nalik ovom:

```
ruby 2.3.1p112 (2016-04-26) [x86_64-linux-gnu]
```

Naredbe za instalaciju razlikuju se na različitim distribucijama Linuxa. U primeru su navedene one koje se odnose na distribuciju Debian i Ubuntu. Za korišćenje Fedora distibucije umesto *apt* možemo koristiti *yum* menadžer paketa.

Program RubyInstaller je jedan od načina instalacije Ruby-ja na Windows operativnim sistemima koji se može naći na adresi http://rubyinstaller.org/downloads/. Prilikom njegovog pokretanja moramo prihvatiti licencu i izabrati putanju instalacije.Paketi za instalaciju za ostale operativne sisteme se mogu naći na adresi http://www.ruby-lang.org/en/downloads.[3]

6.2 Pokretanje programa

Kod napisan u Ruby se može pokrenuti na tri različita načina. Prvi je kada kreiramo datoteku sa programskim kodom i pokrenemo je naredbom:

```
$ ruby primer.rb
```

Kada nam je potrebno izvršavanje nekoliko linija koda možemo pokrenuti i interaktivnu konzolu Ruby jezika naredbom:

```
$ irb
```

Moguće je izvršiti kod koji se piše zajedno sa pozivom interpretera korišćenjem opcije *-e*. Ovde se kod prosleđuje kao argument:

```
$ ruby -e "puts 'Zdravo!'"
```

7 Primer jednostavnog kôda

Naredni primer koda 5 prikazuje kako je moguće pristupiti objektu klase String, u našem primeru pod imenom *linija*, preko drugog objekta klase String, u primeru 'Ruby'. Ukoliko se vrednost 'Ruby' poklapa sa nekim delom vrednosti *linija* onda će se vratiti objekat klase String sa vrednošću 'Ruby', u suprotnom se vraća *nil*. Takođe, u primeru se vidi da smo ovu mogućnost jezika iskoristili da promenimo vrednost 'Ruby' u '*Ruby*'.

```
linija = "Boja i ukus cokolade Ruby poticu od zrna
    kakaovca"

# Izlaz "BOJA I UKUS COKOLADE *RUBY* POTICU OD ZRNA
    KAKAOVCA"
linija['Ruby'] = "*Ruby*"
puts linija.upcase

# pet puta vrsi ispis "Boja i ukus cokolade *Ruby* poticu
    od zrna kakaovca"
5.times { puts linija }
```

Listing 5: Prvi primer koda

Primer koda 6 nam pokazuje da Ruby zna šta programer hoće čak i kada izvršava matematičke operacije nad objektom klase Array. Ispisuje elemente koji predstavljaju razliku dva objekta klase Array.

```
#Izlaz "Treba nam jos da bi sve bilo posadjeno:
#    kruska
#    sljiva
#    breskva"

sadnice  = %w[ jabuka kruska sljiva tresnja breskva]
zasadjeno = %w[tresnja jabuka]
puts "Trebaju nam jos i naredne sadnice" +
     "da bi sve bilo posadjeno:",
     sadnice - zasadjeno
```

Listing 6: Drugi primer koda

8 Zaključak

Osim toga što je namenjen da usreći programera koji ga koristi, takođe je upotrebljiv u različitim oblastima primene i poseduje veliku moć. Zbog toga je za veoma kratko vreme postao veoma popularan i cenjen širom sveta. Ovim radom smo početnicima obezbedili osnovne informacije dovoljne za razumevanje karakteristika i namene Ruby-ja, ne zalazeći u detalje.

Literatura

[1] J. Britt. Ruby Documentation. on-line at: http://www.ruby-doc.com/docs/ProgrammingRuby/.

[2] Huw Collingbourne. *The Book of Ruby: A Hands-On Guide for the Adventurous*. No Starch Press, 2011.

[3] The Ruby community. Ruby Language. on-line at: https://www.ruby-lang.org/en/.

[4] The Ruby community. Ruby on Rails. on-line at: https://rubyonrails.org/.

[5] P. Cooper. *Beginning Ruby from Novice to Professional*. Apress, 2007.

[6] D. Flanagan and M. Yukihiro. *The Ruby Programing Language*. O'Reilly, 2008.

[7] Shin-Ichiro Hara. Ruby talk. on-line at: http://blade.nagaokaut.ac.jp/ruby/ruby-talk/index.shtml.

[8] Christoph Olszowka. The Ruby Toolbox, 2009. on-line at: https://www.ruby-toolbox.com/projects/rails.

[9] Chris Seaton. Specialising dynamic techniques for implementing the Ruby Programming Language. page 26, 2015.