Wprowadzenie do języka Ruby

Aleksander Pohl
apohllo.pl

Krakow Ruby Users Group

19. maja 2007

Kilka słów o Ruby

- Yukihiro „Matz” Matsumoto:
  [...] Poszukiwałem języka potężniejszego od Perla i bardziej obiektowego od Pythona. Wówczas, mając w pamięci moje stare marzenie, zdecydowałem się zaprojektować mój własny język. [...] Nazwałem go Ruby, tak jak ten czerwony kamień szlachetny, i udostępniłem go w 1995.
- Dynamiczny, w pełni obiektowy język skryptowy.
- Aktualna wersja stabilna: 1.8.6
- Początkowo znany jedynie w Japonii.
- Popularność w USA zawdzięcza platformie *Ruby on Rails* napisanej przez Davida Heinemeiera Hanssona

Ruby – narzędzia i zasoby

- **irb** – interaktywna powłoka Ruby
- **ri** – dokumentacja z linii poleceń
- **RDoc** – system dokumentacji Ruby
- **rake** – wzorowany na Make'u system budowania aplikacji
- **rant** – inny system budowania aplikacji
- **ruby-lang.org** – oficjalna strona języka
- **rubyinstaller.rubyforge.org/wiki/wiki.pl** – instalator dla Windows

Plan prezentacji

- Typy podstawowe
  - liczby (Fixnum, Bignum, Float)
  - przedziały (Range)
  - symbole (Symbol)
  - łańcuchy znaków (String)
  - wyrażenia regularne (Regexp)
  - tablice zwykłe (Array)
  - tablice asocjacyjne (Hash)
- Struktury języka
  - Struktury kontrolne
  - Funkcje
  - Bloki, wyjątki
- Obiektowość w Ruby
  - Klasy i moduły
  - Atrybuty i metody

Typy podstawowe

Liczby, przedziały, symbol

```
1:   1 + 2 * 3 # 1.+(2.*(3))
2:   => 7
3:   2 * 3 + 1 # (2.*(3)).+(1)
4:   => 7
5:   1.hour + 5.minutes
6:   => 3900
7:   (1..10).to_a
8:   => [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
9:   (-1..-5).to_a
10:  => []
11:  class Fred end
12:  s1 = :Fred
13:  Fred = 1
14:  s2 = :Fred
15:  s1.object_id == s2.object_id
16:  => true
```

Łańcuchy i wyrażenia regularne

```
1:   "Ruby on rails".length
2:   => 13
3:   name = "Matz"
4:   "Witaj #{name}"
5:   => "Witaj Matz"
6:   re = Regexp.new("([[:lower:]]{2})")
7:   md = re.match(name)
8:   => #<MatchData:0xb7d41054>
9:   md.pre_match
10:  => "M"
11:  md.value_at 1
12:  => "at"
13:  md.post_match
14:  => "z"
```

Wyrażenia regularne – cd.

```
1:   name = "Matz"
2:   name =~ /([[:lower:]]{2})/
3:   => 1
4:   $'
5:   => "M"
6:   $1
7:   => "at"
8:   $'
9:   => "z"
10:  name.sub(/\A./,"K")
11:  => "Katz"
12:  name
13:  => "Matz"
14:  name.sub!(/\A./,"K")
15:  name
16:  => "Katz"
```

Tablice zwykłe

```
1:   a = [ "zero", "one", "two", "three" ]
2:   a[0]
3:   => "zero"
4:   a[-1]
5:   => "three"
6:   a[0..2]
7:   => ["zero", "one", "two"]
8:   a[0...2]
9:   => ["zero", "one"]
10:  a.sort
11:  => ["one", "three", "two", "zero"]
12:  b = ["one", 2, "three"]
13:  b.sort
14:  -:14:in 'sort':  comparison of String
15:    with 2 failed
```

Tablice asocjacyjne

```
1:   h = { :one => 1, :two => 2, :three => 3 }
2:   h[:one]
3:   => 1
4:   h.has_key?(:two)
5:   => true
6:   h["four"]
7:   => nil
8:   hist = Hash.new(0)
9:   hist["four"]
10:  => 0
11:  hist["four"] += 1
12:  => 1
13:  h1 = {1 => "a", 2 => "b"}
14:  h2 = {2 => "c", 3 => "d"}
15:  h1.merge h2
16:  => {1 => "a", 2 => "c", 3 => "d"}
```

Struktury języka

Struktury kontrolne

```
1:   puts "Positive" if x > 0
2:   unless name.nil?
3:     name.upcase!
4:   end
5:   allowed = case user.role
6:       when /admin/
7:         true
8:       when /developer/
9:         true
10:      else
11:        false
12:    end
13:  while line !~ /\Aend\Z/
14:    line = gets
15:  end
```

Funkcje

```
1:  def hello(name)
2:    "Witaj #{name}!"
3:  end
4:  hello("Jan B.")
5:  => "Witaj Jan B.!"
6:  def good_morning(who="Vietnam")
7:    return "Dzień dobry #{who}!"
8:  end
9:  good_morning
10: => "Dzień dobry Vietnam!"
11: def good_bye(*guys)
12:   "Do widzenia " + guys.join(", ")
13: end
14: good_bye("Divan", "Elfin")
15: => "Do widzenia Divan, Elfin"
```

Bloki

```
1:  def oldstyle(&block)
2:    yield
3:  end
4:  oldstyle { "Witam szanownego Pana!"  }
5:  => "Witam szanownego Pana"
6:  def freestyle(who, &block)
7:    yield who
8:  end
9:  freestyle("Ziom") do |w|
10:   "Yo #{w}!"
11: end
12: => "Yo Ziom!"
13: p = Proc.new {|w| "Hej #{w}"}
14: freestyle("ludzie",&p)
15: => "Hej ludzie"
```

Bloki – cd.

```
1:   def simple(greeting)
2:     lambda {|a| greeting + a }
3:   end
4:   closure = simple("Cześć ")
5:   closure.call("Wacek")
6:   => "Cześć Wacek"
7:   ("a".."g").to_a.collect{|a| a.succ}
8:   => ["b", "c", "d", "e", "f", "g", "h"]
9:   [1,2,3,4,5,6].delete_if{|n| n % 2 == 0}
10:  => [1, 3, 5]
11:  ["abc", "cde", "fgh"].find{|str| str =~ /de/}
12:  => "cde"
13:  File.new("plik.txt") do |f|
14:    # ...
15:  end
```

Wyjątki

```
1:  begin
2:    f = File.new("plik.txt","r")
3:    f.read
4:  rescue Exception => e
5:    puts "Error occured...."
6:    raise
7:  ensure
8:    f.close unless f.nil?
9:  end
10: begin
11:   if angry?
12:     throw Something
13:   end
14: catch Something
15:   #do something...
16: end
```

Obiektowość w Ruby

Klasy

```
1:  class Person
2:    def initialize(name, surname)
3:      @name = name
4:      @surname = surname
5:    end
6:  end
7:  p = Person.new("Yukihiro", "Matsumoto")
8:  => #<Person:0xb7def67c>
9:  class Person
10:   def to_s
11:     "#{@name} #{@surname}"
12:   end
13: end
14: p = Person.new("Yukihiro", "Matsumoto")
15: => "Yukihiro Matsumoto"
```

Klasy i moduły

```
1:   class Player < Person
2:     include Comparable
3:     attr_reader :score
4:     def initialize(name, surname, score)
5:       super(name, surname)
6:       @score = score
7:     end
8:     def <=>(other)
9:       self.score <=> other.score
10:     end
11:  end
12:  p1 = Player.new("John","Poor",1)
13:  p2 = Player.new("Frank","Rich",30)
14:  p1 > p2
15:  => false
```

Atrybuty

```
1:   class Figure
2:     @@instances = 0
3:     RED = 0xff0000
4:     BLUE = 0x0000ff
5:     attr_reader :id
6:     attr :color
7:     def initialize(color)
8:       @@instances += 1
9:       @id = @@instances
10:      @color = color
11:    end
12:    def to_s
13:      "Figura #{@id} z #{@@instances}:" +
14:      "kolor 0x#{@color.to_s(16)}"
15:    end
16:  end
```

Atrybuty – cd.

```
1:   f1 = Figure.new(Figure::RED)
2:   => "Figura 1 z 1:kolor 0xff0000"
3:   f2 = Figure.new(Figure::BLUE)
4:   => "Figura 1 z 2:kolor 0xff"
5:   f1.id
6:   => 1
7:   f1.color.to_s(16)
8:   => "ff0000"
9:   f1.color = Figure::BLUE
10:  f1.color
11:  => "ff"
12:  f1.id = 5
13:  -:12:  undefined method 'id=' for ...
```

Metody

```
1:   class Figure
2:     def Figure.instances
3:       @@instances
4:     end
5:     def position=(new_position)
6:       @position = calculate(new_position)
7:     end
8:     def blue?
9:       @color == BLUE
10:    end
11:  private
12:     def calculate(position)
13:       #do some calculation ...
14:    end
15:  end
```

Metody – cd.

```
1:  "Ala ma kota".length
2:  => 11
3:  "Ala ma kota".send(:length)
4:  => 11
5:  class Empty
6:    def method_missing(method_id)
7:      "Brak metody '#{method_id}'"
8:    end
9:  end
10: e = Empty.new
11: e.brakujaca
12: => "Brak metody 'brakujaca'"
```

Metody – cd.

```
1:  dwa = "dwa"
2:  class « dwa
3:    def +(number)
4:      2 + number
5:    end
6:  end
7:  dwa + 1
8:  => 3
9:  class String
10:   alias length old_length
11:   def length
12:     old_length + 1
13:   end
14: end
15: "a".length
16: => 2
```

Koniec