Ruby on Rails Webinar

- Michał Makaruk

Grupa docelowa

- Osoby, które mają podstawową wiedzę o programowaniu.
- A chcą się nauczyć Ruby on Rails.

Materiały

- Www.clab.type.pl

Skąd uczyć się Ruby??

- http://tryruby.org/levels/1/challenges/0
- http://rubykoans.com/
- http://www.codecademy.com/tracks/ruby

Czy da się nauczyć ??

- Ruby powinniśmy się uczyć na początku przed C++ i przed Java.
- Jedynie prostszy język od Ruby to podobno Groovy.

Historia Ruby

Ruby (wym. /ˈru:bi/)[2] to interpretowany, w pełni obiektowy i dynamicznie typowany język programowania stworzony w 1995 roku przez Yukihiro Matsumoto (pseudonim Matz). W języku angielskim ruby oznacza rubin.

Źródło wikipedia.

RUBY PROSTOTA

- Nie musimy deklarować typów.
- Nie musimy, aby rozpocząć program deklarować include , klas,itd. Po prostu rozpoczynamy pisać:)

Przegląd składni

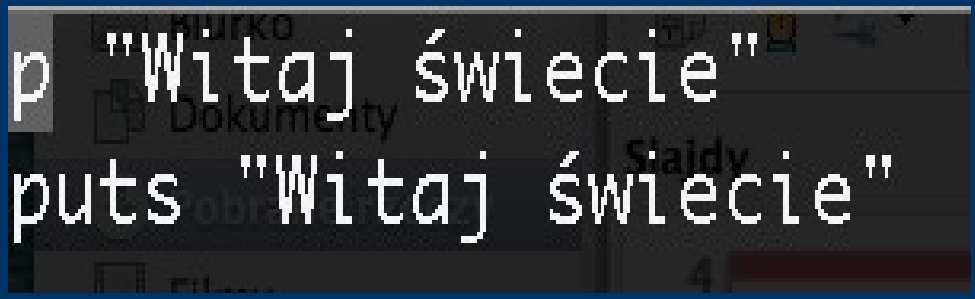
- Brak średników (pod warunkiem ,że nie umieszczamy wiele poleceń w jednej linii).
- 1-linijkowe komentarze rozpoczynają się od #.
- Specjalny obiekt reprezentujący wartość nil.
- Brak deklaracji typów.
- Opcjonalne nawiasy.

Przykład składni

a= 5
b = „Alamakota” Puts a
Puts b

Polecenie p bądź puts

- Polecenie p bądź puts wypisuje na ekran dane.

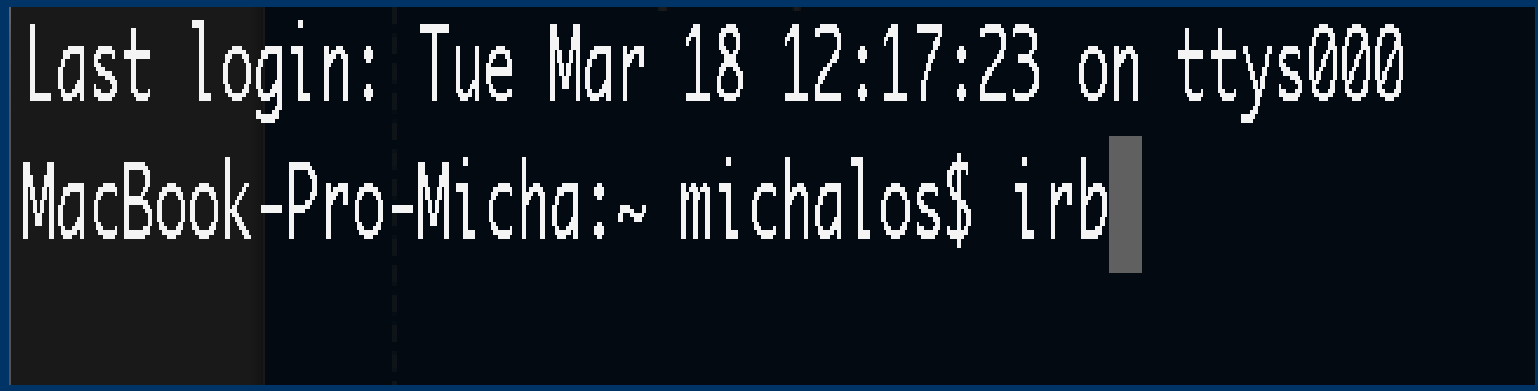


IRB

Interaktywny ruby , używamy za pomocą wykonania irb
Dzięki niemu możemy testować interpretować na bieżąco instrukcje naszych skryptów.

Uruchamiamy IRB i po prostu piszemy....

- Przykład IRB



Inny sposób uruchomienia kodu..

- Plik musi być z rozszerzeniem .rb
- Uruchomienie kodu.
- Poleceniem ruby 1.rb uruchamiamy nasz napisany kod.

```
MacBook-Pro-Micha:~ michalos$ ruby 1.rb
Witam świecie
MacBook-Pro-Micha:~ michalos$
```

Zakresy

W niektórych sytuacjach lepiej przechowywać samą listę zamiast konkretnych wartości . Na przykład ,aby zaprezentować listę liter od A do Z w pierwszym odruchu większość programistów zapisze:
x = ['A','B','C', itd...]
Do tego celu służy Zakres , w tym celu ('A'..'Z') dla liczb od 1 do 10 , zakres 1..10

Wypisanie zakresu

(1..5).each do |p| puts p
End
Często popełniany błąd 1..5 bez nawiasów

Tablice

```
a = [ "zero", "one", "two", "three"]
a[0]
#=> "zero"
a[-1]
#=> "three"
a[-2]
"two"
a[10]
nil
```

Symbole

Symbole to iterały , które nie mają wartości, a których najważniejszą cechą jest nazwa. Symbol rozpoczynamy za pomocą dwukropoku :symbol.

Tablice asocjacyjne

Tablice asocjacyjne, zwane również hashami, słownikami lub mapami to struktury danych przypominające zwykłe tablice, lecz indeksowane dowolnymi wartościami. W Ruby tablice asocjacyjne tworzy się, umieszczając w nawiasach klamrowych oddzielone przecinkami pary klucz => wartość. Natomiast dostęp do poszczególnych wartości odbywa się w taki sam sposób jak w przypadku tablic (nazwa tablicy, po której następuje, ujęty w nawiasy kwadratowe, klucz). W przypadku odwołania do klucza, który nie znajduje się w tablicy domyślnie zwracana jest wartość nil.

Tablice asocjacyjne

hash = { "jeden" => 1, "dwa" => 2, "trzy" => 3}
hash["jeden"] #=> 1 hash["cztery"] #=> nil
hash["cztery"] = 4 hash["cztery"] #=> 4

String

Jeżeli liczby są podstawowym typem danych przetwarzanych przez komputery , to zaraz za nimi plasuje się tekst. Tekstu używa się wszędzie, zwłaszcza w celu komunikowania się z użytkownikami.

```
X = "Test"
Y = "CiągZnaków”
Puts "Udało się!!!" if x+y =="TestCiągZnaków'
```

String

Aby włączyć kilka wierszu tekstu , musimy wykonać następującą deklarację:

x = %q{ To jest tekst, który składa się z Kilku wierszy}

x = %q!To jest tekst, Który także składa się Z kilku wierszy !

Zamiana na String

Do zamiany na String służy metoda to_s.

Może być ona wywoływana min. na tablicach,zmiennych całkowitych i zmiennoprzecinkowych.

```
irb(main):013:0> a = [1,2,3]
=> [1, 2, 3]
irb(main):014:0> a.to_s
=> "[1, 2, 3]"
irb(main):015:0> a.to_s.class
=> String
irb(main):016:0> a = 5
=> 5
irb(main):017:0> a.to_s
=> "5"
```

Pętle

Uwaga w Ruby nie używamy praktycznie pętli for , oraz while . W zastępstwie używamy bloków, które poznamy na dalszym wykładzie.Realizacja powtarzania jakiejś czynności odbywa się za pomocą times np.

```
10.times do
puts "Witaj świecie"
end
```

If

```
a = 8
if a==8
puts a end
a =10 unless a==8 a = 8 if a!=8
```

RubyGems

RubyGems to narzędzie do tworzenia pakietów programów i bibliotek języka Ruby. Narzędzia pozwala tworzyć pakiety bibliotek Ruby, które łatwo utrzymywać i instalować. RubyGems ułatwia zarządzanie różnymi wersjami tej samej biblioteki na komputerze , a także pozwala instalować biblioteki za pomocą jednego polecenia wpisywanego w wierszu poleceń.

Użycie Gemów

Gem install nazwa_gemu – instalacja gemu
Gem list – wyświetla wszystkie zainstalowane gemy
Gem list – remote – wyświetla liste zdalnych gemów Przykład użycia:

```
require 'RedCloth'
r = RedCloth.new(" to jest *test” _działania)
```

Sprawdzanie dostępnych gemów

https://www.ruby-toolbox.com/

czym gem bardziej popularny tym większa szansa ,że będzie dla nas dobry

czym częstsze udpate tym większa pewność tego ,że gem nie posiada większej ilości błędów

co zrobić jeśli nie możemy znaleźć gemu do naszego rozwiązania???

Metoda

Metodę w Ruby definiujemy za pomocą słowa kluczowego def.

Metody służą do tego , aby raz napisany kod móc wywoływać wielokrotnie.

Przykład definicji metody

```
Def hello(n)
n.times do
 Puts "Hello"
end
end
```

Metoda

W Ruby możemy wywoływać metodę bez nawiasów czyli np. Hello 2,3.

W Ruby metoda może zwracać kilka argumentów np. Return a,b,c.

W Ruby nie musimy pisać słowa return , wtedy metoda zwraca ostatnią linijkę kodu.

W Ruby metoda może przyjmować argument domyślny np. hello(a=5).

Metoda wywołanie przykład

```
irb(main):001:0> def hello(n)
irb(main):002:1> n.times do
irb(main):003:2* puts "Hello"
irb(main):004:2> end
irb(main):005:1> end
=> nil
irb(main):006:0> hello(5)
Hello
Hello
Hello
Hello
Hello
=> 5
```

RubyKoans

rubykaons to ćwiczenia, które są pomocne w nauce języka Ruby.

http://rubykoans.com/

ściągamy spakowane zadania i rozpakowujemy.

Luki Ruby Koans

- Przykładowa luka do wypełnienia.

```
class AboutClasses < Neo::Koan
  class Dog
  end

  def test_instances_of_classes_can_be_created_with_new
    fido = Dog.new
    assert_equal __, fido.class
  end
```

Jak sprawdzić co dana metoda robi bądź jakie metody mamy dostępne??

- http://apidock.com/
- Methods.

```
irb(main):001:0> a = 5
=> 5
irb(main):002:0> a.methods
=> [:to_s, :inspect, :-@, :+, :-, :*, :/, :div, :%, :modulo, :divmod, :fdiv, :**
, :abs, :magnitude, :==, :===, :<=>, :>, :>=, :<, :<=, :~, :&, :|, :^, :[], :<<,
 :>>, :to_f, :size, :zero?, :odd?, :even?, :succ, :integer?, :upto, :downto, :ti
mes, :next, :pred, :chr, :ord, :to_i, :to_int, :floor, :ceil, :truncate, :round,
 :gcd, :lcm, :gcdlcm, :numerator, :denominator, :to_r, :rationalize, :singleton_
method_added, :coerce, :i, :+@, :eql?, :quo, :remainder, :real?, :nonzero?, :ste
p, :to_c, :real, :imaginary, :imag, :abs2, :arg, :angle, :phase, :rectangular, :
rect, :polar, :conjugate, :conj, :between?, :nil?, :=~, :!~, :hash, :class, :sin
gleton_class, :clone, :dup, :taint, :tainted?, :untaint, :untrust, :untrusted?,
:trust, :freeze, :frozen?, :methods, :singleton_methods, :protected_methods, :pr
ivate_methods, :public_methods, :instance_variables, :instance_variable_get, :in
stance_variable_set, :instance_variable_defined?, :remove_instance_variable, :in
stance_of?, :kind_of?, :is_a?, :tap, :send, :public_send, :respond_to?, :extend,
 :display, :method, :public_method, :define_singleton_method, :object_id, :to_en
um, :enum_for, :equal?, :!, :!=, :instance_eval, :instance_exec, :__send__, :__i
d__]
```