

INTRODUÇÃO AO RUBY

Eduardo Mendes



AGENDA

- Revisão dos conceitos de OO
- Introdução Ruby
- Ruby on Rails

ORIENTAÇÃO A OBJETOS



- O que é encapsulamento ?
- O que é polimorfismo ?
- O que é associação ?
- O que é superclasse e subclasse ?
- O que é hierarquia de classes ?
- O que é método sobrescrito ?
- O que é construtor ?
- O que é um Objeto ?
- O que uma é classe? ?
- O que é Orientação a Objetos ?
- Quais são os membros de uma classe ?
- O que é herança ?



INTRODUÇÃO AO RUBY

INTRODUÇÃO AO RUBY:

INTRODUÇÃO AO RUBY: O QUE É RUBY?

- Linguagem de Programação
- Criada no Japão em 1995 por Yukihiro "Matz" Matsunomoto
- Sintaxe baseada em Perl, Python e Smalltalk
- Não é uma linguagem compilada (como C++, Java, Visual Basic)
- Linguagem interpretada, exige um interpretador Ruby

INTRODUÇÃO AO RUBY: POR QUE RUBY?

- Orientado a objeto
- Código fácil de ler
- Sintaxe, nomeações e comportamentos sem surpresas
- Independente de espaços em branco
- Sem ponto e vírgulas
- Muito de "syntactic sugar"
<http://www.rubyinside.com/syntactic-sugar.html>

INTRODUÇÃO AO RUBY: RUBY X RUBY ON RAILS

Ruby on Rails

Framework web escrito em Ruby

Ruby

Linguagem de propósito geral

Não é apenas para web

USANDO RUBY

IRB (INTERACTIVE RUBY SHELL)

USANDO RUBY: 03 MANEIRAS

- Comandos simples
- Arquivo ruby (.rb)
- Interactive Ruby Shell (IRB)

COMANDOS SIMPLES

`ruby -e 'puts abc'`

`ruby -e 'print abc'`

ARQUIVO RUBY

extensão .rb

nome_do_arquivo.rb

executar

ruby nome_do_arquivo.rb

IRB [INTERACTIVE RUBY SHELL]

- Digitar comandos
- Criar um arquivo inteiro dentro
- Permite interagir com o código em tempo real
- Funciona como uma calculadora
- Excelente para testar códigos
- Basta digitar irb em linha de comando ou terminal
 - variação irb --simple-prompt

DOCUMENTAÇÃO

www.ruby-doc.org/core

ou

`ri comando`

RUBY

TIPOS DE OBJETO

CONSTANTES

- constantes não são objetos
- qualquer identificador que inicializa com letra maiúscula é uma constante

RANGES

- é uma sequência de números ou strings

Array
[1,2,3,4,5,6,7,8,9,10]
Range
1..10

Inclusivos 1..10
[1,2,3,4,5,6,7,8,9,10]
Exclusivos 1...10
[1,2,3,4,5,6,7,8,9]

BOOLEANS

TRUE / FALSE

OPERADORES LÓGICOS E COMPARAÇÃO



SYMBOLS :

- parecem com Strings
- mas... não
- parecem variáveis
- mas... não
- nome do símbolo

Um símbolo é uma "etiqueta" (label) utilizada para identificar um pedaço de dado

útil para representar chaves de hashes



Um símbolo é armazenado na memória apenas uma vez

útil para representar chaves de hashes

VARIÁVEIS

- permite-nos referenciar objetos
- são indefinidas ou atuam como um objeto
- precisam ser inicializadas

INTEGERS

- Inteiros

Fixnum Bignum

[métodos]

abs next

Numéricos

FLOATS

- Ponto flutuante / decimal

[métodos]

round to_i ceil

STRINGS

- sequência de caracteres
- utilizadas com " ou '
- concatenação +

[métodos]

reverse length
downcase upcase capitalize

ARRAYS []

- coleção ordenada de objetos indexados por inteiros
- podem ser de tipos mistos
- << para adicionar um elemento

[métodos]

push pop
shift unshift
each each_index

HASHES {}

- coleções não-ordenadas de objetos indexados por objetos chave -> valor

Um hash é uma coleção de pares de dados que associam uma chave a um valor

[métodos]

OBJETO

- Ruby é uma linguagem orientada a objetos
- O objeto é a célula fundamental da linguagem

VARIÁVEIS

- permite-nos referenciar objetos
- são indefinidas ou atuam como um objeto
- precisam ser inicializadas

CONVENÇÕES

letras_minusculas_separadas_por_sublinhados

primeiro_nome = 3

~~primeiroNome~~
~~primeir nome~~
~~primeironome~~

ESCOPOS

Global	\$variavel
Classe	@@variavel
Instância	@variavel
Local	variavel
Bloco	variavel

CONVENÇÕES

letas_minusculas_separadas_por_sublinhados

primeiro_nome = 3

~~primeiroNome~~
~~primeiro-name~~
~~primeironome~~

dê às suas variáveis
nomes que as identifiquem



dê às suas variáveis
nomes que as identifiquem



ESCOPOS

Global	\$variavel
Classe	@@variavel
Instância	@variavel
Local	variavel
Bloco	variavel

INTEGERS

- Inteiros

Fixnum

Bignum

métodos

.abs

.next

ETÓ

m uma linguagem
ida a objetos
to é a célula
mental da linguagem

RV

Numéricos

Operadores
= + - / *
** +=

FLOATS

- Ponto flutuante / decimal

métodos

.round

.floor

.to_i

.ceil

Operadores

= + - / *
** +=

INTEGERS

- Inteiros

Fixnum

Bignum

métodos

.abs

.next

F

métodos

.abs

.next

Operadores

= + - / *
** +=

FLOATS

- Ponto flutuante / decimal

métodos

.round

.floor

.to_i

.ceil



métodos

.round

.floor



.to_i

.ceil

STRINGS

- sequência de caracteres
- utilizadas com " ou '
- concatenação +

métodos

.reverse	.length	
.downcase	.upcase	.capitalize

```
"Oba"*5  
# => "ObaObaObaObaOba"  
  
'1'*5  
# => 5  
  
'1'*5  
# => "11111"  
  
'Gota d\'água'  
# => "Gota d\'água"
```

```
saudacao = "Hello"  
# => "Hello"  
  
algo = "world"  
# => "world"  
  
saudacao + ' ' + algo  
# => "Hello world"  
  
puts "Este é o meu #{saudacao} #{algo}."  
# Este é o meu Hello world.  
# => nil
```

ATENÇÃO
puts "Este é o meu #{saudacao} #{algo}."
Este é o meu #{saudacao} #{algo}.
=> nil

"Oba" * 5
=> "ObaObaObaObaOba"

1 * 5
=> 5

'1' * 5
=> "11111"

'Gota d\'água'
=> "Gota d'água"

S

```
saudacao = "Hello"  
# => "Hello"
```

```
algo = 'world'  
# => "world"
```

```
saudacao + ' ' + algo  
# => "Hello world"
```

```
puts "Este é o meu #{saudacao} #{algo}."  
# Este é o meu Hello world.  
# => nil
```

ATENÇÃO

```
puts 'Este é o meu #{saudacao} #{algo}'  
# Este é o meu #{saudacao} #{algo}.  
# => nil
```

É o meu Hello world.

ATENÇÃO

```
puts 'Este é o meu #{saudacao} #{algo}.'
# Este é o meu #{saudacao} #{algo}.
# => nil
```

Concatenação +

métodos

.reverse

.length

.downcase

.upcase

.capitalize

ARRAYS []

- coleção ordenada de objetos indexados por inteiros
- podem ser de tipos mistos
- << para adicionar um elemento

BACANA

```
array = [1, 2, 5, 0, 3]
a => [1, 2, 5, 0, 3]
novo_array = array + [9,10]
x => [1, 2, 5, 0, 3, 9, 10]
novo_array = array - [9,10]
# => [1, 2, 5, 0, 3]
novo_array = array - [2]
# => [1, 5, 0, 3]
```

métodos

```
.inspect .to_s .join
.split .reverse .uniq .sort
.delete_at() .delete() .clear()
.push .pop .shift .unshift()
```

bjetos

métodos

.inspect

.to_s

.join



.split

.reverse

.uniq

.sort

.delete_at()

.delete()

.clear()

.push

.pop

.shift

.unshift()



.reverse!

.uniq!

.sort!

ROS

S

BACANA

```
array = [1, 2, 5, 0, 3]  
# => [1, 2, 5, 0, 3]
```

```
novo_array = array + [9,10]  
# => [1, 2, 5, 0, 3, 9, 10]
```

```
novo_array = array - [9,10]  
# => [1, 2, 5, 0, 3]
```

```
novo_array = array - [2]  
# => [1, 5, 0, 3]
```

HASHES {}

- coleções não-ordenadas de objetos indexados por objetos

chave -> valor

irb

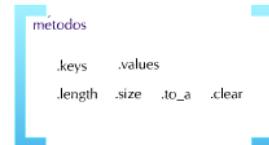
```
pessoa = ['Eduardo', 'Mendes', 'masculino', 'preto', 'preto']
# => ["Eduardo", "Mendes", "masculino", "preto", "preto"]

pessoa = { 'primeiro_nome' => 'Eduardo', 'sobre_nome' => 'Mendes' }
# => {"primeiro_nome=>'Eduardo", "sobre_nome=>'Mendes"}

pessoa['primeiro_nome']          pessoa.index('Eduardo')
# => "Eduardo"                  # => "primeiro_nome"

pessoa['sobre_nome']
# => "Mendes"

pessoa['sexo'] = 'masculino'
# => {"sexo=>'masculino", "primeiro_nome=>'Eduardo", "sobre_nome=>'Mendes"}
```



- coleções não-ordenadas de objetos indexados por objetos
- podem ser iterados
- podem ser convertidos em arrays

IRB

```
pessoa = ['Eduardo', "Mendes", 'masculino', 'preto', 'preto']
# => ["Eduardo", "Mendes", "masculino", "preto", "preto"]
```

```
pessoa = { 'primeiro_nome' => 'Eduardo', 'sobre_nome' => 'Mendes' }
# => {"primeiro_nome"=>"Eduardo", "sobre_nome"=>"Mendes"}
```

```
pessoa['primeiro_nome']
# => "Eduardo"
```

```
pessoa.index('Eduardo')
# => "primeiro_nome"
```

```
pessoa['sobre_nome']
# => "Mendes"
```

```
pessoa['sexo'] = 'masculino'
# => {"sexo"=>"masculino", "primeiro_nome"=>"Eduardo", "sobre_nome"=>"Mendes"}
```



SYMBOLS :

- parecem com Strings
 - mas... não
 - parecem variáveis
 - mas... não



:nome_do_simbolo

Um símbolo é uma
"etiqueta"(label) utilizada para
identificar um pedaço de dado

Um símbolo é
armazenado na memória
apenas uma vez

útil para representar
chaves de hashes

- parecem variáveis

- mas... não

:nome_do_simbolo

Um símbolo é uma
"etiqueta"(label) utilizada para
identificar um pedaço de dado

Um símbolo é
armazenado na memória
apenas uma vez

útil para representar
chaves de hashes



BOOLEANS

TRUE / FALSE

OPERADORES LÓGICOS E COMPARAÇÃO

Igualdade `==`

Negação `!`

Menor do que `<`

Diferente `!=`

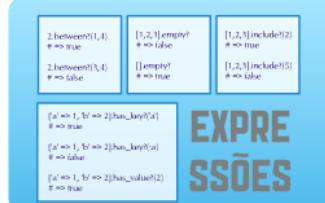
Maior do que `>`

Conjunção: 'E' `&&`

Menor ou igual do que `<=`

Disjunção: 'OU' `||`

Maior ou igual do que `>=`



TRUE / FALSE

OPERADORES LÓGICOS E COMPARAÇÃO

Igualdade ==

Negação !

Menor do que <

Diferente !=

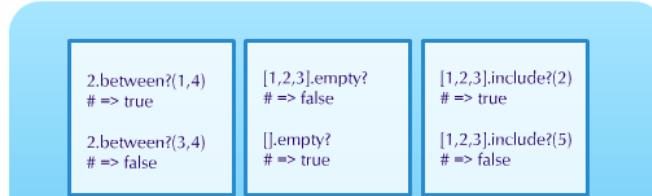
Maior do que >

Conjunção: 'E' &&

Menor ou igual do que <=

Disjunção: 'OU' ||

Maior ou igual do que >=



que

>=

```
2.between?(1,4)  
# => true
```

```
2.between?(3,4)  
# => false
```

```
[1,2,3].empty?  
# => false
```

```
[] .empty?  
# => true
```

```
[1,2,3].include?(2)  
# => true
```

```
[1,2,3].include?(5)  
# => false
```

```
{'a' => 1, 'b' => 2}.has_key?('a')  
# => true
```

```
{'a' => 1, 'b' => 2}.has_key?(:a)  
# => false
```

```
{'a' => 1, 'b' => 2}.has_value?(2)  
# => true
```

**EXPRE
SSÕES**

RANGES

métodos

.begin .end .include?()

```
BACANA
x = 1..10
# => 1..10
z = [*x] #splat
# => [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
'a'..'m'
# => "a".."m"
alpha = 'a'..'m'
# => "a".."m"
alpha.include?('g')
```

- é uma sequência de números ou strings

Array

[1,2,3,4,5,6,7,8,9,10]

Inclusivos 1..10

[1,2,3,4,5,6,7,8,9,10]

Range

1..10

Exclusivos 1...10

[1,2,3,4,5,6,7,8,9]

Array

[1,2,3,4,5,6,7,8,9,10]

Range

1..10



Inclusivos 1..10

[1,2,3,4,5,6,7,8,9,10]

Exclusivos 1...10

[1,2,3,4,5,6,7,8,9]



métodos

.begin

.end

.include?()

BACANA

```
x = 1..10  
# => 1..10
```

```
z = [*x]          #splat  
# => [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
'a'..'m'  
# => "a".."m"
```

```
alfa = 'a'..'m'  
# => "a".."m"
```

```
alfa.include?('g')
```

CONSTANTES

- constantes não são objetos
- qualquer identificador que inicializa com letra maiúscula é uma constante

RUBY

RUBY ESTRUTURAS DE CONTROLE

IF, ELSE, ELSIF

```
if boolean  
  ...  
end
```

```
if boolean  
  ...  
else  
  ...  
end
```

```
if boolean  
  ...  
elsif  
  ...  
else  
  ...  
end
```

UNLESS, CASE

```
unless boolean  
  ...  
end
```

```
if boolean  
  ...  
end
```

```
case  
when boolean  
  ...  
when boolean  
  ...  
else  
  ...  
end
```

```
case  
when boolean  
  ...  
when boolean  
  ...  
else  
  ...  
end
```

TERNÁRIO

boolean ? código1 : código2

ITERADORES

```
k = 0  
while k < 5  
  puts "Hello"  
  k+=1  
end
```

```
5.times  
  puts "Hello"  
end
```

```
Loop(5){ puts "Hello" }
```

```
5.downto(1){ puts "Hello" }
```

```
(1..5).each{ puts "Hello" }
```

```
Loop(5){|i| puts "Hello" + i.to_s}
```

```
return ["Pérola", "Anais", "Maricá"]  
for item in items  
  puts item.capitalize  
end
```

LAÇOS

```
loop do  
  ...  
end
```

controles

break: finaliza o laço
next: pula para a próxima iteração
redo: refaz a iteração atual
retry: repete o laço inteiro

```
while boolean  
  ...  
end
```

```
until boolean  
  ...  
end
```

```
until boolean  
  ...  
end
```



if, ELSE, ELSIF

```
if boolean  
...  
end
```

```
if boolean  
...  
else  
...  
end
```

```
if boolean  
...  
elsif  
...  
else  
...  
end
```

if boolean

...

end

```
if boolean  
...  
end
```

```
if boolean  
...  
else  
...  
end
```

if, ELSE, ELSIF

```
if boolean  
...  
end
```

```
if boolean  
...  
else  
...  
end
```

```
if boolean  
...  
elsif  
...  
else  
...  
end
```

UNLESS, CASE

```
unless boolean  
...  
end
```

```
if !boolean  
...  
end
```

```
case  
when boolean  
...  
when boolean  
...  
end
```

```
case  
when boolean  
...  
when boolean  
...  
else  
...  
end
```

UNLEASH, UNHIDE

```
unless boolean  
...  
end
```

```
if !boolean  
...  
end
```

```
case
```

```
case  
when boolean  
...  
when boolean  
...  
end
```

```
case  
when boolean  
...  
when boolean  
...  
else  
...  
end
```

TERNÁRIO

boolean ? código1 : código2

LAÇOS

loop do
...
end

```
x = 0
for i in range(10)
    x += 2
    if x >= 20
        break
    print(x)
end
```

controles

break: finaliza o laço

next: pula para a próxima iteração

redo: refaz a iteração atual

retry: repete o laço inteiro

while boolean
...
end
... while boolean

until boolean
...
end
... until boolean

LAÇOS

loop do

...

end

```
x = 0  
# => 0  
  
loop do  
  x += 2  
  break if x >= 20  
  next if x == 6  
  puts x  
end
```

controles

break: finaliza o laço

next: pula para a próxima iteração

redo: refaz a iteração atual

retry: repete o laço inteiro

while boolean

until boolean

x = 0

=> 0

loop do

x += 2

break if x >= 20

next if x == 6

puts x

end

while boolean

...

end

... while boolean

until boolean

...

end

... until boolean

ITERADORES

```
x = 0  
while x < 5  
  puts "Hello"  
  x +=1  
end
```

```
5.times  
  puts "Hello"  
end
```

```
1.upto(5) { puts "Hello" }  
  
5.downto(1) { puts "Hello" }  
  
(1..5).each { puts "Hello" }
```

```
1.upto(5) do |i|  
  puts "Hello" + i.to_s  
end  
  
frutas = ["pera", "uva", "maca"]  
for fruta in frutas  
  puts fruta.capitalize  
end
```



1.upto(5) { puts "Hello" }

5.downto(1) { puts "Hello" }

(1..5).each { puts "Hello" }

```
1.upto(5) do |i|
  puts "Hello" + i.to_s
end
```

```
frutas = ["pera", "uva", "maca"]
for fruta in frutas
  puts fruta.capitalize
end
```