A Linguagem de Programação
Ruby

Apresentação – Diego Rubin

http://diegorubin.com
@diegorubin

Apresentação – Caiena

caiena.net

“*I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That's why I decided to design my own language.*” - Matz

Você vai ouvir falar dessas coisas

- Rdoc (http://ruby-doc.org/)
- Rubygems
- Irb
- Git

Características

- Linguagem de Script
- Totalmente Orientada a Objetos
- Propósito geral

Forte e dinamicamente tipada

```
i = 4
puts i # 4
puts i.class # Fixnum
i/'2' #TypeError: String can't be coerced into Fixnum
I/'2'.to_i # 2
```

Totalmente orientada a objetos

```
5.class # Fixnum
'uma string'.class # String
nil.class # NilClass
```

Algumas Classes Importantes

- Fixnum => 1
- Bignum => 100000000000000
- Float => 1.0
- String => “String”
- Symbol => :symbol
- Range => 1..10
- Array => ['1',2]
- Hash => {:primeiro => 1, 'segundo' => 2.0}
- TrueClass => true
- FalseClass => false
- NilClass => nil

Sintaxe - Variáveis

→ **Variáveis Globais**

As variáveis globais devem começar com “$”.

Exemplo:$i = “global”

```
def teste
    $i = “teste”
end
```

→ **Variareis de Instancia**

As variáveis de instancia devem começar com “@”.

→ **Variareis de Classe**

As variáveis de classe devem começar com “@@”.

Exemplo: class Pessoa

```
    @@pessoas = 1
    def set_nome(nome)
        @nome = nome
    end
    def get_nome
        @nome
    end
end
```

→ **Constantes**

Por padrão, constantes devem possuir todas as letras maiúsculas.

Exemplo: ARGV

Sintaxe – Operadores Lógicos

Operadores	Função
&&	E
||	OU
!	NÃO
==	IGUAL
!=	DIFERENTE

Sintaxe – if e Unless

```
if (cond)
    Comandos
else
    comandos
end
```

```
unless (cond)
    Comandos
else
    comandos
end
```

Exemplo

```
if (i == 5)
    puts(i)
end
```

ou

```
puts(i) if (i == 5)
```

Sintaxe - Case

```
case ( variavel)
when valor
      comandos
when valor2
      comandos
else
      comandos
end
```

Exemplo1

```
case (i)
when 1..6
      puts(“esta entre 1 e 6”)
when 8
      puts (“i igual a 8”)
else
      puts (“nao sei”)
end
```

Exemplo2

```
case (i)
when 1..6
      puts(“esta entre 1 e 6”)
when “string”
      puts (“i não e inteiro”)
else
      puts (“nao sei”)
end
```

Sintaxe – While and Until

```
while (teste logico)
    comandos
end
```

```
until (teste logico)
    comandos
end
```

Exemplo

```
while (i < 10)
    puts i
    i = i +1
End

i = 5
puts i while (i -= 1) > 0
```

Exemplo

```
until (i > 10)
    puts i
    i = i +1
End

i = 5
puts i until (i -= 1) < 0
```

Sintaxe - Métodos

```
def nome_da_funcao(parametro1, parametro2, ...)
    <bloco de comandos>
end
```

Exemplo:

```
def alo(nome)
    puts “Oi #{nome}.”
end
```

Parâmetro Variável:

```
def alo(*args)
    args.each {|a| puts “Oi #{a}”}
end
```

O Comando return:

```
def soma(a,b)
    a + b
end
```

```
def dict(a,b)
    return a
    puts b
end
```

Problema - 1

Fatorial

Sintaxe - String

string = “qualquer string”

“1 + 1 = #{ 1+ 1}” # “1 + 1 = 2”
'1 + 1 = #{ 1 + 1}' # “1 + 1 = \#{ 1 + 1}”
a = “<h1>” # “<h1>”
a += “</h1>” # “<h1></h1>”

a.sub(“h1”,”h2”) # “<h2></h1>”
a.gsub(“h1”,”h2”) # “<h2></h2>”

Sintaxe - Regex

```
regx =  /\.\w+$/


regx.match(“arquivo.txt”)   # #<MatchData ".txt">
“arquivo.txt”.sub(regx,””) # “arquivo”
```

Sintaxe - Classe

```
class Nome_da_classe
    comandos
end
```

Exemplo 1:

```
class Pessoa
    def set_nome(nome)
        @nome = nome
    end
    def get_nome
        @nome
    end
end
```

Instanciando 1:

```
pessoa = Pessoa.new
```

Exemplo 2:

```
class Pessoa
    def initialize(nome)
        @nome = nome
    end
    def set_nome(nome)
        @nome = nome
    end
    def get_nome
        @nome
    end
end
```

Instanciando 2:

```
pessoa = Pessoa.new(“Exemplo”)
```

Visibilidade:

Public, private e protected

Observação:

O nome das classes devem começar com uma letra maiúscula

Sintaxe - Herança

```
class Nome_da_classe < Nome_da_classe_pai
    comandos
end
```

Exemplo 1:

```
class Estudante < Pessoa
    def set_ra(ra)
        @ra = ra
    end
    def get_ra
        @ra
    end
end
```

Instanciando 1:

```
estudante = Estudante.new
> Estudante.superclass
= Pessoa
```

Exemplo 2:

```
class Estudante < Pessoa
    def initialize(nome,ra)
        @nome = nome
        @ra = ra
    end
    def set_ra(ra)
        @ra = ra
    end
    def get_ra
        @ra
    end
end
```

Instanciando 2:

```
estudante=Estudante.new(“Exemplo”, 001)
```

Problema - 2

Extendendo a classe String

Sintaxe – Setters e Getters

Gera, em tempo de execução, os métodos setters e getters dos atributos.

Exemplo:

```
class Pessoa
      attr_accessor :nome
end
```

→ **Para gerar somente os setters**

```
attr_writer :nome
```

→ **Para gerar somente os getters**

```
attr_reader :nome
```

Sintaxe - Módulos

```
module NomeDoModulo
    comandos
end
```

Exemplo:

```
module Lobo
    def self.uivar
        “ahuuu”
    end
end
```

Saída:

```
irb(main):036:0> Lobo.uivar
=> "ahuuu"
```

Sintaxe – Mixin's

Alternativa a herança múltipla

```
class NomeDaClasse < NomeDaClassePai
    include nome_do_modulo
end
```

Exemplo:

```
class Lobisomen < Pessoa
    include Lobo
end
```

Métodos Singletons

Consiste em definir um metodo para uma instancia que a classe pai não possui.

Exemplo:

```
class Mamifero
    def produz_leite
        puts “usa como alimento para seus filhotes”
    end
    def possui_pelos
        puts “possui pelos”
    end
end

morcego = Mamifero.new
def morcego.voa
    puts “um mamifero que voa”
end
```

Duck Typing

“If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck.” - Duck Test

```
class Music
  attr_accessor :title
end

class Book
  attr_accessor :title
end

def print_title(media)
  klass = case media.class.to_s
  when "Music"
    "Múscia"
  when "Book"
    "Livro"
  end
  puts "#{klass}: #{media.title}"
end
m = Music.new; m.title = "Alive"
b = Book.new; b.title = "O Rei do Inverno"

print_title(m) # "Alive"
print_title(b) # "O Rei do Inverno"
```

Sintaxe - Array

Construtor:
nome_array = []

Atribuindo Valores:

nome_array[0] = 5
nome_array[1] = “String”
nome_array[2] = Pessoa.new

nome_array.each { |i| puts i}

Sintaxe - Hash

Construtor:
nome_hash = {}

Atribuindo Valores:

nome_hash= {1 => 2, “a” => “b” }
nome_hash[“palavra”] = “word”
nome_hash[“classe pessa”] = Pessoa.new

Exemplo:

Saída do exemplo acima no irb.

irb(main):014:0> nome_hash
=> {5=>10, "pessoa"=>#<Pessoa:0xb7c853f4>, "a"=>"b", 1=>5}

Sintaxe – Bloco de Comandos

```
do
    <comandos>
end
```

Ou

```
{
    <comandos>
}
```

```
do |args|
    <comandos>
end
```

Ou

```
{ |args|
    <comandos>
}
```

Implementação:

```
def metodo
    yield
end
```

Utilização:

```
metodo { <comandos> }
```

Com Parâmetros:

```
def metodo(n)
    yield(n)
end
```

Utilização:

```
metodo(2) {|i| i + 3}
```

Sintaxe - Proc

```
p = Proc.new do
    <comandos>
end
```

Ou

```
proc do
    <comandos>

end
```

Implementação:

```
def metodo
    yield
end
```

Utilização:

```
metodo &p
```

```
p = Proc.new do |args|
    <comandos>
end
```

Ou

```
proc do |args|
    <comandos>

end
```

Com Parâmetros:

```
def metodo(n)
    yield(n)
end
```

Utilização:

```
metodo(2) &p
```

Sintaxe – For

for i **in** expression
  <comandos>
end

Utilização:

```
for i in 1..10
  puts i
end
```

n**.times do |i|**
  <commandos>
end

Utilização:

```
10.times  {|i| puts i}
```

Problema - 3

Criação da classe Texto

Sintaxe – Rescue

```
begin
    Comandos
rescue
    comandos
end
```

```
begin
    Comandos
rescue RuntimeError => e
    Comandos
rescue
    comandos
end
```

Sintaxe – Retry

```
begin
    Comandos
rescue RuntimeError => e
    Comandos
    retry
rescue
    comandos
end
```

Sintaxe – Raise

raise “deu erro”

e = Exception.new(“deu problema”)
raise e

Sintaxe - Require

Importa arquivos de código.
Serve para importar arquivos em código ruby ou código objeto na forma de libs .

Exemplo:

require “arquivo_de_classes.rb”
require “libmysql”
require “rexml/document”
require “myclass.so”

Sintaxe – Include

Importa os métodos de instancia de um módulo para a classe. Os métodos importados serão métodos de instancia.

Exemplo:

```
class Pessoa
  include Estudante
end
```

Sintaxe – Extend

Importa os métodos de instancia de um módulo para a classe. Os métodos importados serão métodos de classe.

Exemplo:

```
class Pessoa
  extend Estudante
end
```

Method Missing

Quando um método é invocado e ele não está definido o method_missing é executado.

Exemplo:

```
class Qualquer
  def method_missing(method, *args)
  end
end
```

Problema - 4

Method missing e singleton

Testes

- A Comunidade Ruby gosta
- Diversas ferramentas
- Test/Unit

Links

- http://dl.dropbox.com/u/1482800/eustaquiorangel.com/tutorialruby.pdf
- http://www.ruby-doc.org
- http://why.nomedojogo.com/