UNIVERSIDADE FEDERAL DE RORAIMA

CENTRO DE CIÊNCIAS DE TECNOLOGIA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Estudo sobre a Linguagem de Programação Ruby

Componentes:

Giovanna Monteiro de Azevedo

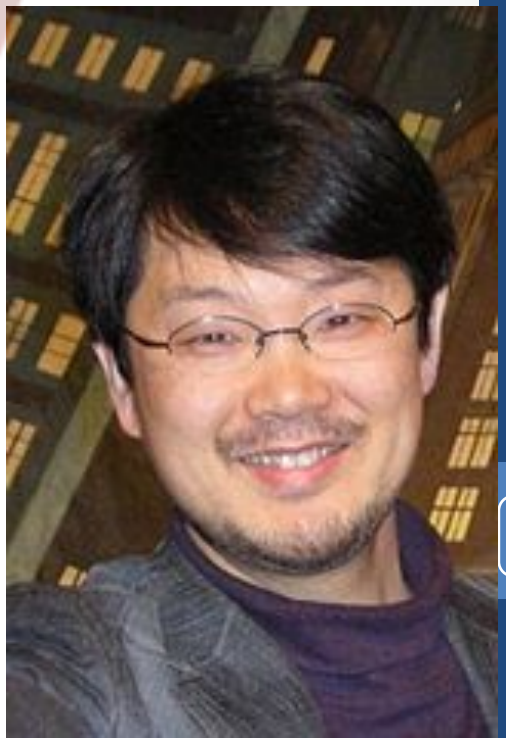
Ibukun Chife Didier Adjtche

Orientador: Herbert Oliveira Rocha

Boa Vista – RR, 2017

Histórico

- O que é Ruby?
- Criada em 1993, no Japão, por Yukihiro "Matz" Matsumoto;
- Com o intuito de ser usada como linguagem de script;
- Motivos para a criação;
- Uma linguagem mais poderosa do que Perl, e mais orientada a objetos do que Python;
- Ruby suporta programação funcional, orientada a objetos, imperativa e reflexiva;



Histórico

- Foi inspirada principalmente por Python, Perl, Smalltalk, Eiffel, Ada e Lisp, sendo muito similar em vários aspectos a Python;
- Foi conhecida em 2004 pelo seu famoso framework, Ruby On Rails, com características de meta-programação no desenvolvimento Web;
- Existem várias implementações alternativas da linguagem, como: *YARV*, *JRuby*, *Rubinius*, *IronRuby*, *MacRuby* e *HotRuby*, com uma abordagem diferente.

Domínios de Aplicação

- Aplicações Web:

  - BaseCamp: é um software com plataforma Web, que gerencia os projetos que podem ser de diversas áreas.

- Simulação

  - NASA Langley Research Center

- Negocio

  - Toronto Rehab : universidade

- Robótica

  - MORPH : grupo de Inteligência em Automação

- Redes

  - Open Domain Server

Domínios de Aplicação

- Telefonica
  - Lucent : empresa ligada ao Alcatel
- Administração de Sistemas
  - Level 3 Communications: empresa especialisada em Ti
- Plataformas:
  - Mac OS x
  - Linux
  - MS-DOS
  - Microsoft
  - Celulares Symbian Series 60
  - Plataformas que contém uma maquina virtual Java (Usando Jruby)

Paradigmas

- Ruby é uma linguagem Multi-paradigma;
  - Ao lado temos um exemplo do paradigma **Orientada Objetos:**


```
class Banco

    def initialize(contas)
        @contas = contas
    end

    def status
        saldo = 0
        for conta in @contas
            saldo += conta
        end
        saldo
    end
end
banco = Banco.new([200, 300, 400])
banco.status
```

Paradigmas

Funcional

```
 >> ens.to_s
=> "{ 1, 5, 3, 2 }"
?> ens.map { |x| x * 10 }
=> [10, 50, 30, 20]
>> ens.reject { |x| x.even? }
=> [1, 5, 3]
>> ens.find { |x| x >= 2 }
=> 5
```

Programação Imperativa

```
int soma = 0;
for(int i=0 ; i< array.length ; i++){
    soma += array[i];
}
return soma;
```

Variáveis e Tipos de Dados

- Exemplo de atribuição: x = 8;
- Variáveis que começam com uma letra maiúscula são chamadas Constantes;
- Ruby não possui tipos primitivos, mas sim todos tipos são classes, assim como todas variáveis são objetos;
- Ruby determina automaticamente o tipo de dados pelo valor atribuído à variável.

BEGIN	def	for	redo	undef
END	defined	if	rescue	unless
alias	do	in	retry	until
and	else	module	return	when
begin	elsif	next	self	while
break	end	nil	super	yield

Variáveis e Tipos de Dados

- 1) Integer
- 2) Float
- 3) String
- 4) Arrays
- 5) Hash
- 6) Symbol
- 7) Range

- Exemplos:

- x=42 #integer
- y = 1.58 #floating point value
- z = "Hello" #string
- w = ['elemento1', 'elemento2', 'elemento3'] #array
- t = {'objeto_chave1' => 23, 'objeto_chave2' => 30} #hash
- puts :simbolo #symbol
- a = (1..7).to_a
- puts a #[1,2,3,4,5,6,7] #range

Comandos de Controle

- As estruturas condicionais são estruturas que mudam pouco entre as linguagens de programação;
- Para utilizar estruturas de controle em Ruby, precisamos antes conhecer os operadores booleanos, true e false. Os operadores booleanos são: ==, >, <, >= e <=;
- Expressões booleanas podem ainda ser combinadas com os operadores && (and) e || (or);

- **1) <u>If:</u>**

```
num = 8
if num == 3
        puts “Número é igual a 3”
elseif num ==10
        puts “Número é igual a 10”
elsif num == 7
        puts “Número é igual a7”
else
        puts “Não encontrado”
end
```

Comandos de Controle

- **2) Unless:** a expressão unless é o oposto da expressão if. E executa o código quando a condição é falsa.

- unless i >= 10
- puts "menor que 10"
- else
- puts "maior igual que 10"
- end

- **3) Case:** é a opção mais simplificada e flexível, pq testa o valor nas afirmações de when.
- case i
- when 0..5
- puts " Esta entre 0 e 5"
- when 6..10
- puts " Esta entre 6 e 10"
- else
- puts i.to_s
- end

Comandos de Controle

- **4) While**: os loops são usados para executar no mesmo bloco de código um específico número de vezes. O loop While executa um bloco de código quando a condição for verdadeira.

- a = 0
- while a < 10
- puts a
- a+=1
- end

- **5) Until**: O loop until é o oposto de um loop while, ou seja, ele roda quando a condição é falsa.

- a = 0
- until a > 10
- puts “a = #{a}”
- a+=2
- end

Comandos de Controle

- **6) For**: O loop de for consiste em uma variável vazia e um range. Para cada interação do loop, a variável vazia recebe um valor correspondente ao elemento do range.

- for a in (1..6)
- puts a
- end

- **7) Loop Do**: permite que o código execute até que a condição do break seja atendida.

- X=0
- looop do
- puts x
- x+=1
- break if x>10
- end

Escopo (Regras de Visibilidade)

- Ruby tem uma estrutura de bloco aninhada, ou seja:
  - Declarações no nível mais externo têm escopo global (nível 1).
  - Declarações dentro de um bloco interno são locais ao bloco; cada bloco está dentro de outro bloco, com um nível a mais.

- O escopo (âmbito) de uma variável local em Ruby é um dos seguintes:
- proc{…}
- loop{…}
- def…end
- class…end
- module…end

Escopo (Regras de Visibilidade)

- Ruby implementa o escopo **estático** (as variáveis tem seu escopo determinado antes da execução do programa);

- Mas em Ruby todas as variáveis podem ser:
  - Local;
  - Global – $;
  - Atributos de objeto (de instância) – @ (a variável é acessível somente à classe);
  - Atributos de classe – @@ (a variável é acessível à classe e a todas que herdarem dela).

Escopo (Regras de Visibilidade)

Código 1

- *$s = 2;*
- *def scaled(d)*
- * d*$s;*
- *end*
- *def teste()*
- * s = 3;*
- * scaled(3);*
- *end*
- *puts teste();*

Código 2

- $s = 2;
- def scaled(d)
- d*$s;
- end
- def teste()
- $s = 3;
- scaled(3);
- end
- puts teste();

Escopo (Regras de Visibilidade)

- **Declaração:** é uma frase de um programa que ao ser elaborada produz ocorrências e vínculos. Ruby suporta os principais tipos de declarações, com exceção de declarações colaterais.

- **Definição** – é uma declaração cujo único efeito é produzir associações. Como exemplo, tem-se a declaração de constantes.
  - Ex: Constante = 10

- **Declaração de tipos** - uma declaração de um novo tipo cria um tipo e produz um vínculo. Como em Ruby TUDO é objeto, pode-se definir um novo tipo através de classes (TAD).
  - Ex:
```
classFoo
    @foo
end
```

Escopo (Regras de Visibilidade)

- **Declaração de variáveis** – uma declaração de variável nova cria uma variável e produz um vínculo. Como Ruby é dinamicamente tipada, a declaração de uma variável é feita quando se associa um valor à mesma:
  - Ex: foo = Foo.new

- **Declaração sequencial** – sintaxe semelhante aos comandos sequenciais:
  - Ex: @foo; @foo2

```
# declaracao da variavel @foo e da variavel @foo2
```

- **Declaração recursiva** – Uma declaração recursiva usa as próprias declarações que ela produz, pode ser, por exemplo, uma declaração de tipos recursivos ou definições de funções e procedimentos recursivos.
  - Ex:

```
def fib(n)
        return 1 if n <= 1
        return fib(n – 1) + fib(n – 2)
end
```

UBrTracker

Realiza o quantitativo de veículos do aplicativo Uber no Brasil e também executa o levantamento de percursos do centro de todas as capitais do país até os seus respectivos Aeroportos, contendo as variáveis de distância, tempo e valor da corrida.

UBrTracker

- Com o intuito de facilitar o desenvolvimento do projeto, utilizou-se os editores de texto Geany e SublimeText.

- Os testes e a execução foram executados na linha de comando através da seguinte entrada:
- env $(cat .env3 | xargs) ruby nome_do_arquivo.rb;

Conclusão

- A linguagem Ruby:
  - Aprendizado;
  - Funcionalidades;
  - Capacidade de migração e de comunicação com outras linguagens como C, HTML, PQP, SQL, CSS;
- API Uber:
  - Dificuldades;
  - Limitações.
- Futuras Aprimorações;
  - Site, Web, Mobile.

Referências Bibliográficas

- Caelum. Desenvolvimento Ágil para Web com Ruby on Rails 4. Disponível em: <www.caelum.com.br/apostilas/> Acesso em: 15 de julho 2017.

- Damasceno, H. **Paradigma Linguagem Ruby**. Curso ciência da computação, Faculdade Anglo Americano. Disponível em: <https://fr.slideshare.net/heverson/ruby-apresentacao/> Acesso em: 17 de julho 2017.

- **Ligação e Escopo**. Disponível em: <https://linguagemruby.wordpress.com/ligacao-e-escopo/> Acesso em 19 de julho 2017.

- Menegotto, A. B. e Mierlo F. **A Linguagem Ruby**. UNISINOS - Universidade do Vale do Rio dos Sinos Centro de Ciências Exatas e Técnológicas São Leopoldo - RS – Brasil.

Referências Bibliográficas

- **Ruby (linguagem de programação)**. Disponível em: <https://pt.wikipedia.org/wiki/Ruby_(linguagem_de_programa%C3%A7%C3%A3o) > Acesso em: 17 de julho 2017.

- **Ruby doc**. Disponível em: <http://ruby-doc.org/> Acesso em: 15 de julho 2017.

- **Site Oficial do Ruby**. Disponível em: <https://www.ruby-lang.org/pt/> Acesso em: 15 de julho 2017.

- **Tipos de dados em Ruby**. Disponível em:<http://www.devmedia.com.br/tipos-de-dados-em-ruby/33600>Acesso em: 15 de julho 2017.

OBRIGADO!