Estudo sobre a Linguagem de Programação Ruby

João Paulo Parreira Peixoto e João Pedro Oliveira Silva

Departamento de Ciência da Computação – Universidade Federal de Roraima (UFRR)
Boa Vista – RR – Brasil

jpparreirap@gmail.com, joaopedro2195@gmail.com

Resumo: *Será apresentada a história da linguagem Ruby, várias áreas em que o Ruby se destacou sendo um sucesso nas áreas utilizadas, em empresas muito ricas como a NASA, os vários paradigmas que o Ruby suporta sendo ele nomeado multiparadigma. Sendo uma linguagem capaz de suportar a orientação a objeto foi capaz de criar um mini jogo de RPG, como um exemplo prático.*

1. História do surgimento da linguagem

Ruby se tornou reconhecida no meio especializado desde que Dave Thomas, conhecido como um dos "Programadores Pragmáticos" adotou-o como uma de suas linguagens preferidas e acabou por escrever um dos mais completos livros sobre a linguagem, o Programming Ruby. Com o advento desta publicação, a linguagem passou a contar com uma boa fonte de iniciação e referência em inglês, aumentando consequentemente o número de adeptos da linguagem no Ocidente.

Devido a grande exposição de um framework web feita em Ruby, o Ruby on Rails desenvolvido por David Heinemeier Hansson, a linguagem tem sido foco da mídia especializada justamente pela sua praticidade.

Esta mesma praticidade inclusive é um dos conceitos básicos desta linguagem. É possível fazer algoritmos que resolvam seus problemas, não necessitando se preocupar com as limitações da linguagem ou do interpretador.

2. Domínios de aplicação

Simulações

NASA Langley Research Center usa Ruby para realizar simulações.

Um grupo de pesquisa na Motorola usa Ruby para fazer scripts para um simulador, tanto para gerar cenários como para processar esses mesmos dados depois.

Modelagem 3D

O Google SketchUp é uma aplicação de modelagem 3D que utiliza o Ruby para sua macro-API de scripting.

Negócios

Toronto Rehab usa um programa baseado no RubyWebDialogs para gerir e acompanhar o suporte via telefone e pessoal das equipes de help desk de TI e operações de TI.

Robótica

No projeto MORPHA , Ruby foi usado para implementar a parte do controle reativo do robô de serviços da Siemens.

Redes

O Open Domain Server usa Ruby de forma a permitir que as pessoas usem clientes de DNS Dinâmicos para a atualização em tempo real das configurações de IP para que possam ser mapeadas em domínios estáticos.

Telefonia

Ruby está sendo utilizado na Lucent num produto de telefonia 3G wireless.

Administração de Sistemas

Ruby foi usado para escrever o componente de coleta de dados do sistema de Capacidade unix e Planejamento da Level 3 Communications, que recolhe estatísticas de performance de cerca de 1700 servidores Unix (Solaris e Linux) espalhados pelo mundo.

Aplicações Web

Basecamp, uma aplicação de gestão de projetos online desenvolvida pela 37signals é programada inteiramente em Ruby.

A List Arpart, uma revista para pessoas interessadas na criação de websites que existem desde 1997, foi recentemente renovada e usa uma aplicação personalizada construída em Ruby on Rails.

3. Paradigmas suportados pela linguagem

Orientada a objeto:

```
class Carro
    def initialize(marca,modelo,cor,tanque)
        @marca = marca
        @modelo = modelo
        @cor = cor
        @tanque = tanque
    end
end

gol = Carro::new(“Volkswagen”,”Gol”,:azul,50)
puts gol
```

Funcional:

```
soma = 0
for i in (0..array.length) do
    soma += array[i]
end
soma
```

Interpretada, pois possui um gerenciador de pacotes bastante avançado, flexível e eficiente: **RubyGems.** As *gems* podem ser vistas como bibliotecas reutilizáveis de código Ruby, que podem até conter algum código nativo (em C, Java, .Net). São análogos aos jars no ambiente Java, ou os assemblies do mundo .Net.

4. Variáveis e tipos de dados

Não existem tipos primitivos na linguagem Ruby, e todos os tipos são objetos. A seguir, alguns detalhes sobre os principais tipos.

- Integer representa valores numéricos inteiros, é uma classe abstrata

- Fixnum é uma das subclasses de Integer, e também representa números inteiros. Mas essa classe possui um limite de valor que ela pode armazenar dependendo da arquitetura do computador.
- Bignum também é uma subclasse de Integer e ela armazena valores inteiros maiores que o Fixnum. E o limite dos valores também depende nesse caso da memória disponível.
- Float é usada para representar valores de ponto flutuante, ou seja, números com valores fracionados, independentemente do tamanho desse número.
- String é uma classe para textos, são delimitadas por aspas simples ou aspas duplas.
- Symbol tem um comportamento parecido com a String, porém são declaradas com dois pontos antes do nome do objeto (ex: **:nome**, **:idade**) e não possuem tanta flexibilidade, não suportando acentos nem espaços em branco, mas sempre ocupam o mesmo espaço de memória e sempre utilizada em Hash.
- Array é a classe para trabalhar com arrays
- Hash é a classe para trabalhar com hash maps, que se parecem com os arrays, porém seus índices podem ser qualquer tipo de objeto, sendo comumente utilizado para servir de índices os objetos do tipo String e Symbol.
- Ranges é a classe que armazena intervalo de dados, por exemplo, "numnaturais = 1..10" ou "alfabeto = 'a' .. 'z' ".

4. Comandos de controle

Condicionais:

- **IF**

```
[taq@~]irb
irb(main):001:0> i = 10
=> 10
irb(main):002:0> if i > 10
irb(main):003:1>     puts "maior que 10"
irb(main):004:1> elsif i == 10
irb(main):005:1>     puts "igual a 10"
irb(main):006:1> else
irb(main):007:1*     puts "menor que 10"
irb(main):008:1> end
```

```
igual a 10
=> nil
irb(main):009:0>
```

- **UNLESS –** Pode usar como uma forma negativa do **IF**

```
[taq@~]irb
irb(main):001:0> i = 10 => 10
irb(main):002:0> unless i >= 10 # se i N~AO FOR maior ou igual a 10
irb(main):003:1>      puts "menor que 10"
irb(main):004:1> else
irb(main):005:1*      puts "maior igual que 10"
irb(main):006:1> end maior igual que 10
=> nil
irb(main):007:0> puts "menor que 11" unless i > 11 # se i N~AO FOR maior que 11,
imprima! menor que 11
=> nil
irb(main):008:0>
```

- **CASE**

```
[taq@~]irb
irb(main):001:0> i = 10
=> 10
irb(main):002:0> case i
irb(main):003:1>      when 0..5
irb(main):004:1>          puts "entre 0 e 5"
irb(main):005:1>      when 6..10
irb(main):006:1>          puts "entre 6 e 10"
irb(main):007:1>      else
irb(main):008:1*          puts "???"
irb(main):009:1> end
entre 6 e 10
=> nil
irb(main):010:0>
```

Loops

Há quatro maneiras de interagir com o conteúdo do loop:

break sai do loop completamente

next vai para a próximo iteração

return sai do loop e do método onde o loop está contido

redo restarta o loop

- **WHILE**

```
[taq@~]irb
irb(main):001:0> i = 0
=> 0
irb(main):002:0> while i < 5
irb(main):003:1>     puts i
irb(main):004:1>     i += 1
irb(main):005:1> end
0
1
2
3
4
=> nil
irb(main):006:0>
```

- **FOR**

```
[taq@~]irb
irb(main):001:0> for i in (0..5)
irb(main):002:1>     puts i
irb(main):003:1> end
0
1
2
3
4
5
=> 0..5
irb(main):004:0>
```

- **UNTIL**

```
[taq@~]irb
```

```
irb(main):001:0> i = 0
=> 0
irb(main):002:0> until i == 5
irb(main):003:1>      puts i
irb(main):004:1>      i += 1
irb(main):005:1> end
0
1
2
3
4
=> nil
irb(main):006:0>
```

- **BEGIN**

```
[taq@~]irb
irb(main):001:0> i = 0
=> 0
irb(main):002:0> begin
irb(main):003:1*      puts i
irb(main):004:1>      i += 1
irb(main):005:1> end while i < 5
0
1
2
3
4
=> nil
irb(main):006:0> begin
irb(main):007:1*      puts i
irb(main):008:1>      i -= 1
irb(main):009:1> end until i == 0
5
4
3
2
1
```

```
=> nil
irb(main):010:0>
```

- **LOOP**

```
[taq@~]irb
irb(main):001:0> i = 0
=> 0
irb(main):002:0> loop do
irb(main):003:1*      break unless i < 5
irb(main):004:1>      puts i
irb(main):005:1>      i += 1
irb(main):006:1> end
0
1
2
3
4
=> nil
irb(main):007:0>
```

5. 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

O programa inteiro (se não há nenhum dos itens anteriores)

Ruby implementa o escopo estático (as variáveis tem seu escopo determinado antes da execução do programa, ou seja, não é determinado durante o tempo de execução, como seria o escopo dinâmico). Mas em Ruby todas as variáveis podem ser local, global – $, atributos de objeto (de instância) – @ (a variável é acessível somente à classe) ou atributos de classe – @@ (a variável é acessível à classe e a todas que herdarem dela). O que pode confudir na hora de determinar qual escopo a linguagem tem. Por exemplo:

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();
```

No código 1 será impresso 6 e no código 2 será impresso 9. Nesses códigos exemplificam duas diferentes formas de acesso a uma variável, mas o escopo continua sendo estático.Não está relacionado com o TIPO de escopo, mas com o escopo em si (escopo = faixa de instruções no qual uma variável é visível). Ou seja, está relacionado com a forma de acesso a ambientes locais ou não locais.

Declarações

Uma 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:

```
class Foo

@foo

end
```

Declaração de variáveis – uma declaração de variável nova cria uma variável e produz um vínculo. Como foi dito ateriormente, em Ruby, todas as variáveis podem ser local, global ou atributos de classe. 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 seqüenciais:

Ex:

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

obs: o '#' significa início de comentário

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
```

6. Exemplo prático de uso da linguagem de programação

O paradigma utilizado para fazer o jogo de RPG foi o Orientado a Objeto, pois deixa o código com uma fácil compreensão e deixa o código fonte muito mais organizado. O Ambiente de desenvolvimento que foi utilizado foi o NetBeans 8.0.

O Código fonte do jogo de RPG está disponível em: https://github.com/SephirothZX/ProjetoFinalLP2016JPauloJPedro/blob/master/CodigoFonte.

7. Conclusões

Após algumas pesquisas para este projeto, podemos concluir que a linguagem de programação Ruby se assemelha muito com Python, ambas fazem com que o desenvolvimento de um código se torne mais prático. Apesar do Ruby ser uma linguagem que ainda esta aumentando o numero de adeptos da linguagem, ainda sim é bem poderosa e usada em diversas areas e situações da nossa vida, como por exemplo: simulações, modelagem 3D, suporte de telefonias, róbotica, redes, administração de sistemas e aplicações web.

Nessas Aplicações web a ferramenta principal do ruby que é utilizada é o framework on rails(ruby on rails). Com algumas pesquisas feitas também podemos concluir que diferente de Python, o Ruby possui mais oportunidades de emprego devido a sua surpreendente velocidade de criar websites com o "on rails", diferente do Python que a maioria das oportunidades de emprego são por indicação.

Porém, a vantagem do Python é que como ele está a mais tempo no mercado existe muito mais casos de sucesso, possui uma comunidade de programadores ativa e colaborativa maior, diversas fontes de estudo e boa documentação e exemplos de códigos, o que definitivamente o Ruby ainda não é tao forte quanto a isso, por ter pouco tempo no marcado, não possui uma biblioteca tao grande de apoio para estudo ou consultas quanto Python.

8. References

Oficina da Net. RUBY, O QUE É?, 2008. Disponível em: <https://www.oficinadanet.com.br/artigo/1072/ruby_o_que_e>. Acesso em: 07/08/2016.

HIDEKI, Eric. PYTHON OU RUBY: QUAIS SÃO AS DIFERENÇAS E O MERCADO DE TRABALHO, 2013. Disponível em: <https://ericstk.wordpress.com/2013/11/11/python-ou-ruby-quais-sao-as-diferencas-e-o-mercado-de-trabalho/>. Acesso em: 10/08/2016.

STONE, Mike. IS RUBY A FUNCIONAL LANGUAGUE?, 2008. Disponível em: <http://stackoverflow.com/questions/159797/is-ruby-a-functional-language>. Acesso em: 09/08/2016.

Code Academy. VISÃO GERAL E UMA PEQUENA AMOSTRA. Disponível em: <https://www.codecademy.com/pt-BR/courses/ruby-beginner-pt-BR/0/1?curriculum_id=535951905d81961e16000001>. Acesso em: 06/08/2016.

RANGEL, Eustáquio. TUTORIAL DE RUBY, 2005. Disponível em: <https://leanpub.com/conhecendo-ruby>. Acesso em: 07/08/2016.

Mississippi College. RUBY EXAMPLE CODE. Disponível em: <http://sandbox.mc.edu/~bennet/ruby/code/>. Acesso em: 06/08/2016.

Caelum. A LINGUAGEM RUBY. Disponível em: <https://www.caelum.com.br/apostila-ruby-on-rails/a-linguagem-ruby/#2-1-a-historia-do-ruby-e-suas-caracteristicas>. Acesso em: 08/08/2016.

SANTANA, Carlos. TIPOS DE DADOS EM RUBY, 2013. Disponível em: <https://luaderuby.wordpress.com/2013/02/01/tipos-de-dados/>. Acesso em: 04/08/2016.

MANTOVANI, Ricardo. RUBY-CLASSES-ESCOPOS-LINUX, 2014. Disponível em: <https://desenvolvimentoaberto.org/2014/12/28/ruby-classes-escopos-linux/>. Acesso em: 06/08/2016.

DANIEL, José. TIPOS DE DADOS EM RUBY. Disponivel em: <http://www.devmedia.com.br/tipos-de-dados-em-ruby/33600>. Acesso em: 06/08/2016.

CRUZ, Thiago. CONHECENDO A LINGUAGEM RUBY, 2015. Disponível em: <http://www.devmedia.com.br/conhecendo-a-linguagem-ruby/8226>. Acesso em: 06/08/2016.

BRIZENO, Marcos. CONCEITOS NA PRÁTICA: PROGRAMAÇÃO FUNCIONAL COM RUBY, 2013. Disponível em: <https://brizeno.wordpress.com/2013/12/06/conceitos-na-pratica-programacao-funcional-com-ruby/>. Acesso em: 09/08/2016.

CORREIA, Fábio. RUBY – VARIÁVEIS, 2008. Disponível em: <https://wmagician.wordpress.com/2008/01/26/ruby-variaveis/>. Acesso em: 09/08/2016.

MENEZES, Alexandre P., GOIS, Ely L. LIGAÇÃO E ESCOPO, 2008. Disponível em: <https://linguagemruby.wordpress.com/ligacao-e-escopo/>. Acesso em: 07/08/2016.

SOBRE O RUBY. Disponível em: <https://www.ruby-lang.org/pt/about/>. Acesso em: 06/08/2016.

HISTÓRIAS DE SUCESSO. Disponível em: <https://www.ruby-lang.org/pt/documentation/success-stories/>. Acesso em: 03/08/2016.

PARA RUBY A PARTIR DE C E C++. Disponível em: <https://www.ruby-lang.org/pt/documentation/ruby-from-other-languages/to-ruby-from-c-and-cpp/>. Acesso em: 04/08/2016.