Curso Completo

Aula 034

Block, Proc e Lambdas

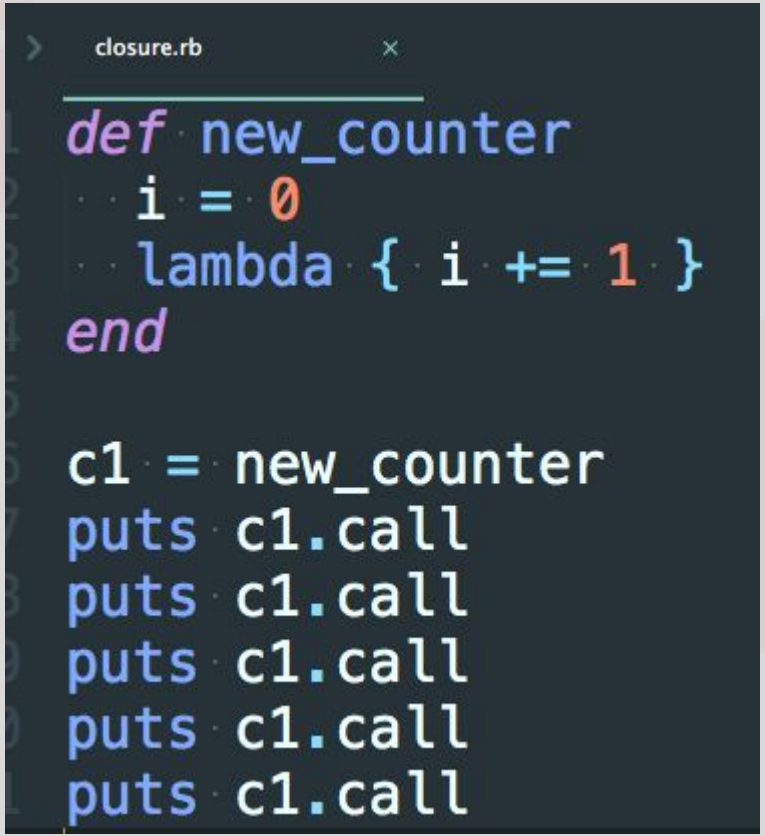
Closures

https://pt.wikipedia.org/wiki/Clausura_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o)

Closures

"Uma closure lembra-se do contexto onde foi criada."

Closures


closure.rb
def new_counter
i = 0
lambda { i += 1 }
end
c1 = new_counter
puts c1.call
puts c1.call
puts c1.call
puts c1.call
puts c1.call

Três variedades de Closure

`&block`

`Proc.new`

`lambda`

Block

Techo de código compreendido entre **do end** ou **{}**

Block

Trechos grandes - **do end**
Trechos de única linha - **{}**

Exemplo

```
{ puts “olá” }

do
  puts “olá”
  puts “mundo”
end
```

Block

Qualquer método pode ser chamado com um bloco como argumento implícito, ou seja, dentro do método você pode chamar o bloco usando a palavra-chave **yield** (ceder) com um valor.

Exemplo

block_1.rb

```ruby
def saudacao
  puts "olá"
  puts yield
  puts "seja bem-vindo!"
end

saudacao { puts "Jackson" }
```

block_given?

Quando não passamos um bloco esperado uma exceção é levantada.

block_given?

Contorne com um **block_given?**

Exemplo

block_2.rb

```ruby
def saudacao
  if block_given?
    puts "olá"
    puts yield
  else
    puts "O bloco não foi passado!"
  end
end

saudacao { puts "Jackson" }
```

Passando argumentos para Bloco

Você pode prover parâmetros para a chamada a yield: estes serão passados para o bloco. Dentro do bloco, você lista os nomes dos argumentos para receber os parâmetros entre barras verticais (|).

Exemplo

block_3.rb

```ruby
def saudacao
  if block_given?
    puts yield("Olá",3)
  else
    puts "O bloco não foi passado!"
  end
end

saudacao { |str,num| puts "#{str} Jackson! " * num }
```

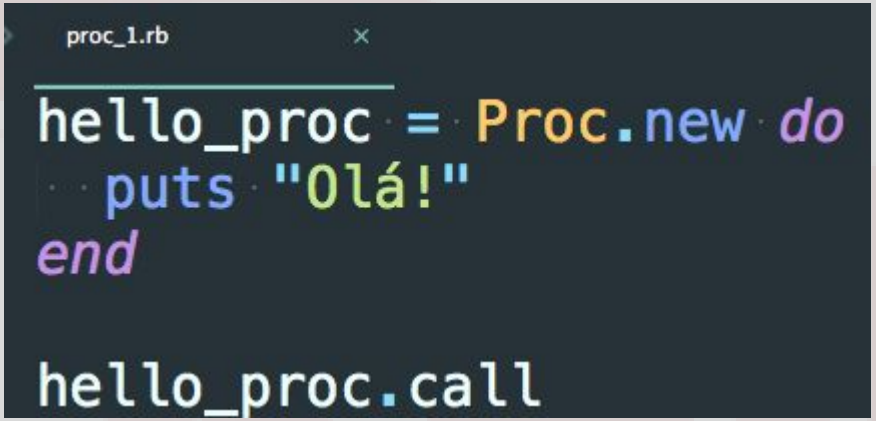
Proc

http://ruby-doc.org/core-2.2.0/Proc.html

Proc / Lambda

http://ruby-doc.org/core-2.2.0/Proc.html

Exemplo Proc



Exemplo Proc

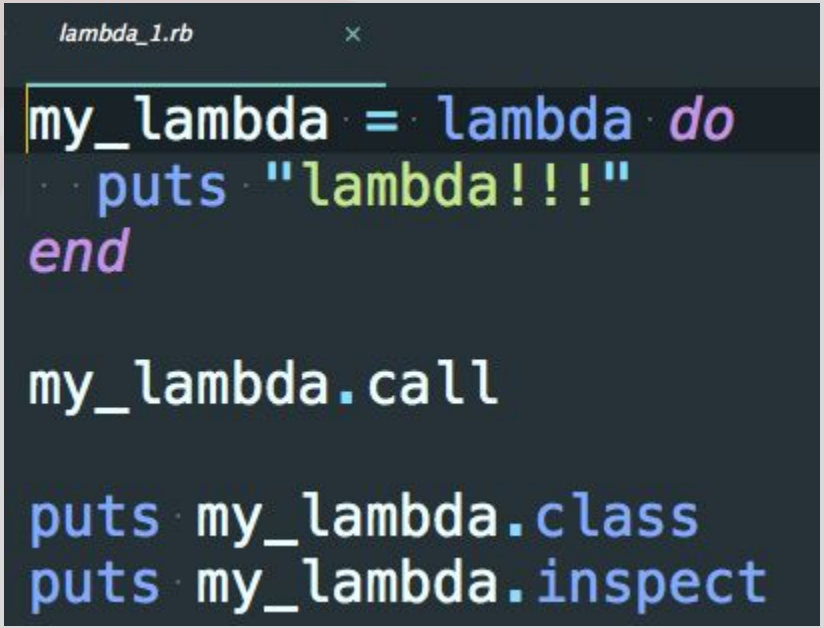
proc_2.rb

```ruby
hello_proc = proc do
  puts "Olá!"
end

hello_proc.call

puts hello_proc.class
puts hello_proc.inspect
```

Exemplo Lambda



Exemplo Lambda

lambda_2.rb

```ruby
my_lambda_arrow = -> do
  puts "Arrow lambda!!!"
end

my_lambda_arrow.call

puts my_lambda_arrow.class
puts my_lambda_arrow.inspect
```