Introducción a Ruby on Rails

Funciones y Métodos

Overview

- Funciones/Métodos
  - Definiciones
  - Cómo invocarlos?
  - Qué y cómo retornan?
  - Argumentos por defecto
- Cómo hacer a los métodos mas expresivos
- Qué es “splat”.

Funciones y Métodos

- Técnicamente, una función es definida fuera de una clase, y un método es definido dentro de una clase.
- En ruby, toda función/método tiene al menos una clase a la que pertenece
  - Pero..no siempre está escrito dentro de una clase

Toda función es en realidad un método en ruby.

Métodos

- Los paréntesis son opcionales
  - Tanto cuando se define como cuando se invoca al método.
- Son utilizados por claridad.

```ruby
def metodo
  puts "sin paréntesis"
end

def metodo_()
  puts "con paréntesis"
end

metodo() # => no parens
metodo # => no parens
metodo_ # => yes parens
```

Retorno / Return

- No se necesita declarar el tipo de los parámetros.
- Se puede retornar cualquier cosa.
- El keyword **return** es opcional
  - La última línea ejecutada es retornada.

```
def sumar(uno, two)
  uno + two
end

def dividir(uno, two)
  return "No puedo hacer esta división!" if two == 0
  uno / two
end

puts sumar(2, 2) # => 4
puts dividir(2, 0) # => No puedo hacer esta división!
puts dividir(12, 4) # => 3
```

Métodos expresivos

- Los métodos pueden terminar con:
  - ‘?’ - Indica un predicado
  - ‘!’ - Indica que pueden existir efectos peligrosos

```ruby
def puedo_dividir_por(number)
  return false if number.zero?
  true
end

puts puedo_dividir_por 3 # => true
puts puedo_dividir_por 0 # => false
```

Parámetros por defecto

- Se pueden tener argumentos por defecto.
  - Si el valor es pasado se utiliza
  - Caso contrario: se utiliza el valor por defecto provisto en la firma del método

```
def factorial (n)
  n == 0? 1 : n * factorial(n - 1)
end

def factorial_con_parametro_defecto (n = 5)
   n == 0? 1 : n * factorial_con_parametro_defecto(n - 1)
end

puts factorial 5 # => 120
puts factorial_con_parametro_defecto # => 120
puts factorial_con_parametro_defecto(3) # => 6
```

Splats

- Se utiliza * como prefijo del parámetro cuando se quiere indicar que es un array
- Se puede aplicar inclusive a parámetros del medio, no solamente al último.

```ruby
def max(one_param, *numbers, another)
  # Variable length parameters passed in
  # become an array
  numbers.max
end

puts max("something", 7, 32, -4, "more") # => 32
```

Entonces..

- No existe necesidad de declarar el tipo de dato de un argumento o el retorno. Es un lenguaje dinámico.
- **return** es opcional - Se retorna la última línea evaluada.
- Se pueden construir métodos con un número variable de argumentos o argumentos por defecto.