Ruby:

“Orientación a Objetos” y algo más...

Ruby: “Orientación a Objetos” y algo más...

7mas Jornadas Regionales de Software Libre

Agosto de 2007, Córdoba, Argentina

Javier Smaldone
javier@smaldone.com.ar http://blog.smaldone.com.ar

Objetivos de esta charla

- Presentación del lenguaje

  no exhaustiva
  no es un curso

- Características llamativas

  para quien no conozca Lisp ni SmallTalk

- Algunos ejemplos

  comparación con otros lenguajes

Historia

- Desarrollado por Yukihiro “Matz” Matsumoto

liberado en 1993
“conocido” en 2000

- Perl demasiado “sucio”, Python no del todo OO

Smalltalk + sintaxis familiar + regexp + iteradores + ...

- Ruby on Rails lo llevó a la fama (2004)

excepto en Japón

Características

- 100% orientado a objetos

  objetos + métodos

- Tipado dinámico

  late binding, duck typing

- Sintaxis simple

  Principle Of Least Surprise

- Smalltalk + Perl

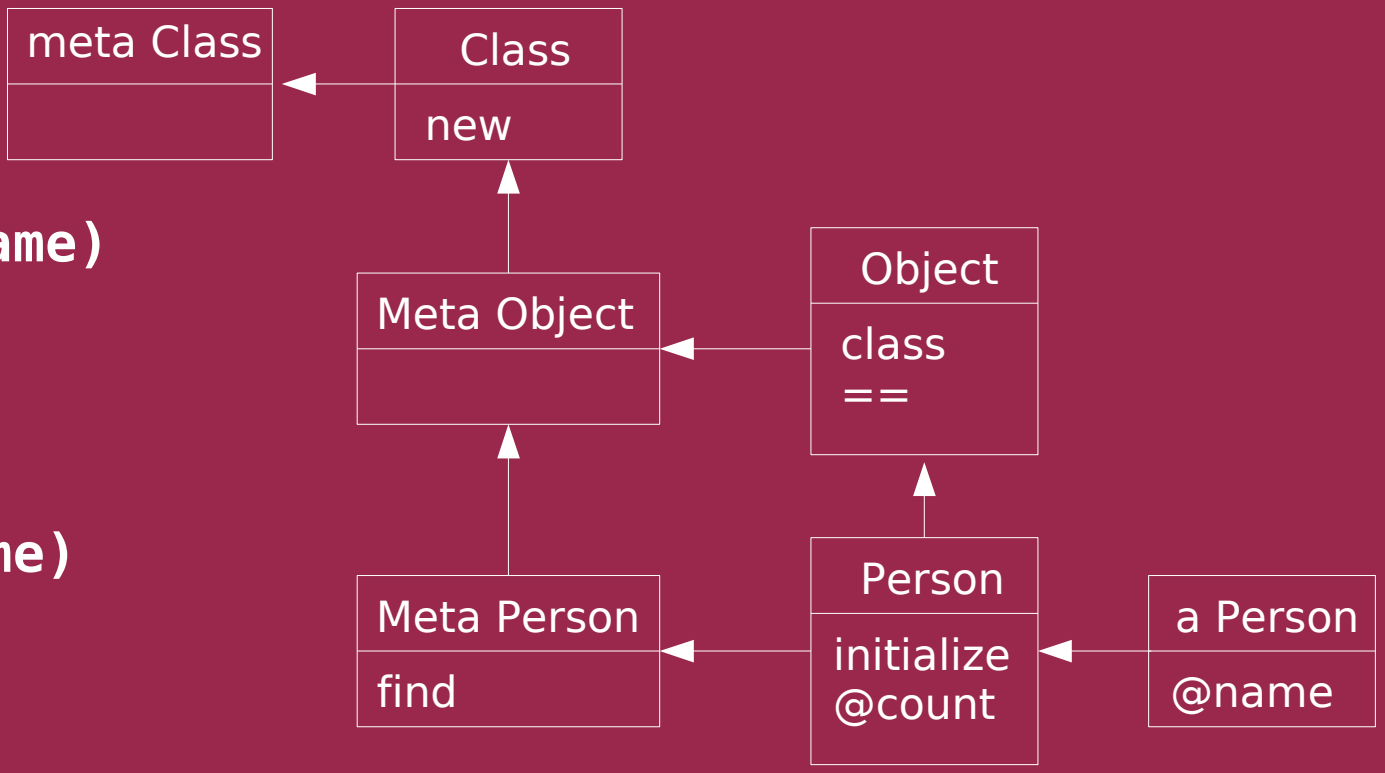
  Perl, pero bien hecho

Clases y meta-clases

```ruby
class Person
  @@count = 0

  def initialize(name)
    @name = name
    @@count += 1
  end

  def self.find(name)
    do_something
  end
end
```




```ruby
p = Person.new('Juan Perez')
q = Person.find('Pedro Lopez')
puts puts p == q ? 'Imposible' : 'Está bien'
```

Duck typing

```ruby
class Duck
  def quack
    'Quack!'
  end
end

class Goose
  def quack
    'Quaaack!'
  end
end

class Dog
  def bark
    'Arf!'
  end
end
```

Si camina como un pato
y grazna como un pato,
debe ser un pato.

```ruby
def make_it_quack(duck)
  puts duck.quack
end

a = Duck.new
b = Goose.new
c = Dog.new

make_it_quack a
make_it_quack b
# Error!
make_it_quack c
```

Herencia y mixins

```ruby
module Quack
  def make_it_quack
    puts quack
  end
end

class Bird
  include Quack
  def quack
    'I say'
  end
end
```

```ruby
class Duck < Bird
  def quack
    super + ' quack!'
  end
end

class Goose < Bird
  def quack
    super + ' quaaack!'
  end
end
```

```ruby
a = Duck.new
b = Goose.new

a.make_it_quack
b.make_it_quack
```

Getters y Setters

```ruby
class Person
  def name
   @name
  end

  def name=(name)
   @name = name
  end
end
```

```ruby
class Person
  attr_accessor :name
  # attr_writer ...
  # attr_reader ...
end
```

```ruby
p = Person.new

p.name = 'Pepe'
puts p.name
```

Introspección

```
'Hello world'.class              String
String.superclass                Object
String.is_a? Object              true
Duck.respond_to? :quack          false
Duck.new.respond_to? :quack      true
String.methods                   ['superclass', 'id', ..., 'is_a?']
'Hello'.methods                  ['to_a', '<<', ..., 'slice!']
'Hello'.private_methods          ['put_c', 'throw', ..., 'warn']
String.instance_methods          ['to_a', '<<', ..., 'slice!']
Class.class                      Class
Class.superclass                 Module
Module.superclass                Object
```

Clausuras (*closures*)

- También llamadas “funciones lambda” o “agentes”.
- Funciones y bloques como parámetros.
- Se pasa el bloque como parámetro, incluyendo las ligaduras al ámbito en donde fue definido.
- Permiten ocultar el entorno y definir funciones de orden superior.
- Existen en otros lenguajes, pero en Ruby son “naturales”.

Ejemplos de clausuras

Problema: Multiplicar por 2 los elementos de una lista

PHP

```php
function double($x) {
    $y = array();
    foreach ($x as $e)
        array_push($y, $e*2);
    return $y;
}
```

Haskell

```haskell
double = map (*2)
```

Ruby

```ruby
def double(a)
  a.map {|e| e*2}
end
```

Ejemplos de clausuras

Problema: Dado un arreglo de la forma `[["nombre", "apellido"]]`, obtener un arreglo de la forma `["apellido, nombre"]`

PHP

```php
function names($a) {
  $b = array();
  foreach ($a as $e)
    array_push($b, $e[1].', '.$e[0]);
  return $b;
}
```

Haskell

```haskell
join [x,y] = x ++ ", " ++ y
names = map (join . reverse)
```

Ruby

```ruby
def names(a)
  a.map{|e| e.reverse.join(', ')}
end
```

Ejemplos de clausuras

Problema: Calcular la edad promedio de las personas mayores de edad.

PHP

```php
$sum = 0;
$count = 0;
foreach ($people as $p){
  if ($p->age > 18){
    $sum += $p->age;
    $count++;
  }
}
if ($count)
  print $sum/$count;
```

Ruby

```ruby
adults = people.select{|p| p.age > 18}
sum = 0
adults.each {|p| sum += p.age }
puts sum.to_f/adults.size if adults.size > 0
```

Ejemplos de clausuras

Problema: Calcular la edad promedio de las personas mayores de edad.

Ruby

```
adults = people.select{|p| p.age > 18}
sum = 0
adults.each {|p| sum += p.age }
puts sum.to_f/adults.size if adults.size > 0
```

```
adults = people.select{|p| p.age > 18}
sum = adults.map{|p| p.age}.inject{|ac,e| ac+e}
puts sum.to_f/adults.size if sum
```

Ejemplos de clausuras

Problema: Encontrar la palabra más larga de una lista.

PHP

```php
function maxpal($a) {
$max = '';
foreach ($a as $e)
    if (strlen($e) > strlen($max))
        $max = $e;
return $max;
}
```

Haskell

```haskell
maxp xs ys = if (length xs) > (length ys) then xs
             else ys
maxpal = foldl maxp ""
```

Ruby

```ruby
def maxpal(a)
  a.inject {|m, e| e.length > m.length ? e : m }
end
```

Ejemplo: QuickSort

```
qs :: Ord a => [a] -> [a]
qs [] = []
qs (x:xs) = (qs (filter (<=x) xs))++[x]++(qs (filter (>x) xs))
```

```
class Array
  def head
    self[0]
  end
  def tail
    slice(1..-1)
  end
  def qs
    empty? ? [] : tail.select{|e| e<=head}.qs +
                  [head] +
                  tail.select{|e| e>head}.qs
  end
end
```

Referencias

- Sitio oficial de Ruby
  http://www.ruby-lang.org
- Ruby Central
  http://www.rubycentral.com
- Documentación (API completa)
  http://www.ruby-doc.org
- Ruby on Rails (framework para aplicaciones web)
  http://www.rubyonrails.org
- Ruby en tu navegador (tutorial interactivo)
  http://tryruby.hobix.com

Referencias

- Tutorial sobre Ruby y metaclases
  http://www.visibleworkings.com/little-ruby/

- Comparativa de performance
  http://www.antoniocangiano.com/articles/2007/02/19/ruby-implementations-shootout-ruby-vs-yarv-vs-jruby-vs-gardens-point-ruby-net-vs-rubinius-vs-cardinal

- Los lenguajes tienden a Lisp
  http://kapcoweb.com/p/static/docs/la-venganza-de-los-nerds/la-venganza-de-los-nerds.html
  http://www.paulgraham.com/icad.html

- Por qué Ruby es un Lisp aceptable
  http://www.randomhacks.net/articles/2005/12/03/why-ruby-is-an-acceptable-lisp

¿Preguntas?

Ruby: “Orientación a Objetos” y algo más...

¡Muchas gracias!

Javier Smaldone
javier@smaldone.com.ar http://blog.smaldone.com.ar