Ruby 2.0: ¿Qué hay de nuevo, viejo?

Edgardo E. Hames

ehames

5 de abril de 2013

Ruby 1.8 tiene los días contados.

Múdense a Ruby 2.0 ya!

Parámetros con nombre (*keyword arguments*)

Ruby 2.0: parámetros con nombres

- Podemos asociar nombres a parámetros de un método

Ruby 2.0: parámetros con nombres

- Podemos asociar nombres a parámetros de un método
- Más descriptivos y fáciles de recordar

Ruby 2.0: parámetros con nombres

- Podemos asociar nombres a parámetros de un método
- Más descriptivos y fáciles de recordar
- El orden no es relevante

Parámetros con nombres: ejemplo

```
def wrap(string, before: "<", after: ">")
  "#{before}#{string}#{after}"
end

wrap("hello")                              #=> "<hello>"
wrap("hello", after:"]", before:"[") #=> "[hello]"
```

Ruby 1.8: simulados usando un hash

Usado en Rails y otros DSLs

```
some_method :foo => "bar"
```

Claves desconocidas pasaban inadvertidas

```
some_method :unknown => "never used"
```

Ruby 2.0: mejor que un simple hash

```ruby
begin
  wrap("hello", unknown: "]")
rescue ArgumentError => e
  e.message
end
```

Refinamientos
(*refinements*)

Ruby 1.8: las clases pueden modificarse

Permite hacer *monkey patching*.

```
class String
  def reverse
    # hacer otra cosa
    self.capitalize!
  end
end
```

Contra: ensucia el entorno de las clases clientes.

Ruby 2.0: tienen visibilidad acotada

```ruby
module IncompatibleStrings
  refine String do
    def reverse
      # hacer otra cosa
      self.capitalize!
    end
  end
end

"abc".reverse                          #=> "cba"
using IncompatibleStrings; "abc".reverse    #=> "Abc"
```

Iteración perezosa
(*lazy iteration*)

Ruby 1.8: iteraciones sobre enumerados

```
(1..1_000_000).map{|i|  2*i}.first          #=> 2
```

Ruby 2.0: iteración perezosa

La iteración puede ser postergada usando #lazy

```
(1..1_000_000).lazy.map{|i| 2*i}.first #=> 2
```

Nuevos literales

Literal de hash

```
{:on => 1, :off => 2}
```

se puede escribir como

```
{on: 1, off: 2}
```

Arreglos de símbolos %i y %I

Podemos crear arreglos constantes de símbolos

```
%i{on off}    #=> [:on, :off]
```

incluso con interpolación de variables

```
s1, s2 = %w{on off}
%I{#{s1} #{s2}}    #=> [:on, :off]
```

Otras mejoras

- Convención `to_h` para convertir a hash

- Convención `to_h` para convertir a hash
- Nuevo motor de expresiones regulares

- Convención `to_h` para convertir a hash
- Nuevo motor de expresiones regulares
- Module#prepend para extender clases

- Convención `to_h` para convertir a hash
- Nuevo motor de expresiones regulares
- Module#prepend para extender clases
- UTF-8 por defecto

- Convención `to_h` para convertir a hash
- Nuevo motor de expresiones regulares
- Module#prepend para extender clases
- UTF-8 por defecto
- DTrace/TracePoint para depurar

- Convención `to_h` para convertir a hash
- Nuevo motor de expresiones regulares
- Module#prepend para extender clases
- UTF-8 por defecto
- DTrace/TracePoint para depurar
- Optimizaciones en la VM

- Convención `to_h` para convertir a hash
- Nuevo motor de expresiones regulares
- Module#prepend para extender clases
- UTF-8 por defecto
- DTrace/TracePoint para depurar
- Optimizaciones en la VM
  - menor uso de memoria gracias a bitmap GC

- Convención `to_h` para convertir a hash
- Nuevo motor de expresiones regulares
- Module#prepend para extender clases
- UTF-8 por defecto
- DTrace/TracePoint para depurar
- Optimizaciones en la VM
  - menor uso de memoria gracias a bitmap GC
  - `require`

- Convención `to_h` para convertir a hash
- Nuevo motor de expresiones regulares
- Module#prepend para extender clases
- UTF-8 por defecto
- DTrace/TracePoint para depurar
- Optimizaciones en la VM
  - menor uso de memoria gracias a bitmap GC
  - `require`
  - operaciones de punto flotante

- Convención `to_h` para convertir a hash
- Nuevo motor de expresiones regulares
- Module#prepend para extender clases
- UTF-8 por defecto
- DTrace/TracePoint para depurar
- Optimizaciones en la VM
  - menor uso de memoria gracias a bitmap GC
  - `require`
  - operaciones de punto flotante
  - llamadas a métodos

¡gracias por venir!

Referencias

- Ruby 2.0.0-p0 is released
- NEWS for Ruby 2.0.0
- Ruby 2.0 (en)
- Ruby 2.0.0 in detail
- Ruby 2.0.0 by example
- Why you should be excited about garbage collection in Ruby 2.0