Pourquoi :ruby, :ruby_on_rails sont géniaux ?

D'un point de vue technique!

9 juillet 2010

Maxime Menant

Qui parle?

- Maxime Menant
- 25 ans
- Sophia Antipolis
- http://blog.maximemenant.fr
- maxime_menant
- +2 an de dev Ruby on Rails
- +3 ans de dev Web (PHP, Java, ...)

Ruby

:ruby

- Libre
- 100% Objet
- Interprété
- Multi-paradigme
- Syntaxe proche du langage naturel
- Code compact
- Méta-programmation

Les bases de

Ruby

:variables

:variables

Variables :

locale

@instance

@@de_classe

:variables

Constante : COLORS

Variables :

locale

@instance

@@de_classe

:variables

Constante : COLORS

Variables :

locale

@instance

@@de_classe

Accesseur :

attr_accessor
attr_reader
attr_writer

:conditions

:conditions

```
if value == 3 then
   ...
elsif value > 3
   ...
else
   ...
end
```

:conditions

```
if value == 3 then
    ...
elsif value > 3
    ...
else
    ...
end
```

```
unless  value < 20
    ...
end
```

:conditions

```
if value == 3 then
    ...
elsif value > 3
    ...
else
    ...
end
```

```
unless  value < 20
    ...
end
```

```
thresold = true if a < 100

thresold = true unless a >= 100
```

:boucles

:boucles

```
10.times do |i|
    puts i
end
```

:boucles

```
10.times do |i|
    puts i
end
```

```
books.each do |book|
    puts book.name
end
```

:boucles

```
10.times do |i|
    puts i
end
```

```
books.each do |book|
    puts book.name
end
```

```
a *= 2 while a < 100
a *= 2 until a >= 100
```

:symbole

:symbole

```
'symbol'.object_id                              2159861560
```

:symbole

```
'symbol'.object_id          2159861560
```

```
:symbol.object_id           329788
```

:symbole

```
'symbol'.object_id                2159861560
'symbol'.object_id                2159845100
```

```
:symbol.object_id                     329788
```

:symbole

```
'symbol'.object_id          2159861560
'symbol'.object_id          2159845100
```

```
:symbol.object_id           329788
:symbol.object_id           329788
```

:array, :hash

:array, :hash

```ruby
array = []
array << 'one'
array << 'two'
array << 'three'
```

:array, :hash

```
array = []
array << 'one'
array << 'two'
array << 'three'
```

```
['one', 'two', 'three']
```

:array, :hash

```
array = []
array << 'one'
array << 'two'
array << 'three'
```

```
['one', 'two', 'three']
```

```
hash = {}
hash[:one]    = 1
hash[:two]    = 2
hash[:three]  = 3
```

:array, :hash

```
array = []
array << 'one'
array << 'two'
array << 'three'
```

```
['one', 'two', 'three']
```

```
hash = {}
hash[:one]    = 1
hash[:two]    = 2
hash[:three]  = 3
```

```
{:one => 1, :two => 2, :three => 3]
```

:classes

```
class Wizard
  def initialize(name)
    @name = name
  end
end
```

:classes

```ruby
class Wizard
  def initialize(name)
    @name = name
  end
end
```

```ruby
merlin = Wizard.new 'Merlin'
puts merlin.inspect
```

:classes

```
class Wizard
  def initialize(name)
    @name = name
  end
end
```

```
merlin = Wizard.new 'Merlin'
puts merlin.inspect
```

```
#<Wizard:0x100124070 @name="Merlin">
```

:classes

```
class Wizard
  attr_reader :name

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

:classes

```
class Wizard
  attr_reader :name

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

```
merlin =  Wizard.new 'Merlin'
puts merlin.name
```

:classes

```
class Wizard
  attr_reader :name

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

```
merlin =  Wizard.new 'Merlin'
puts merlin.name
```

```
Merlin
```

:modules, :mixins

```ruby
module Spell
  def cast(spell)
    puts "#{self.name} invokes #{spell}!"
  end
end
```

:modules, :mixins

```ruby
module Spell
  def cast(spell)
    puts "#{self.name} invokes #{spell}!"
  end
end
```

```ruby
class Wizard
  include Spell
end
```

:modules, :mixins

```
module Spell
  def cast(spell)
    puts "#{self.name} invokes #{spell}!"
  end
end
```

```
class Wizard
  include Spell
end
```

```
merlin =  Wizard.new 'Merlin'
merlin.cast 'Thunder Bolt'
```

:modules, :mixins

```ruby
module Spell
  def cast(spell)
    puts "#{self.name} invokes #{spell}!"
  end
end
```

```ruby
class Wizard
  include Spell
end
```

```ruby
merlin =  Wizard.new 'Merlin'
merlin.cast 'Thunder Bolt'
```

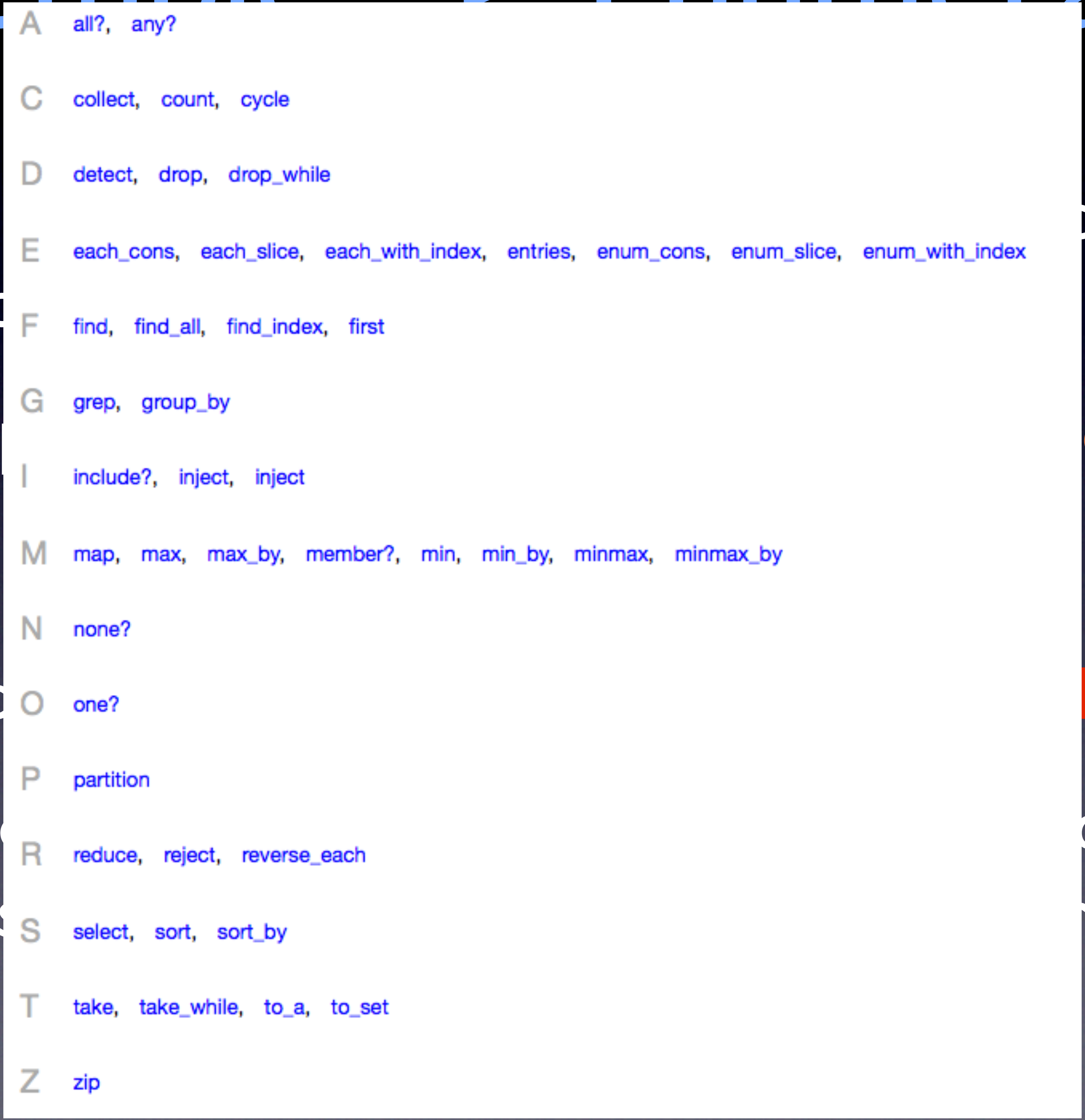
```
Merlin invokes Thunder Bolt!
```

:exemple => Enumerable

- Soit une classe A contenant une collection d'objets B
- A possède la méthode de parcours each
- et B la méthode de comparaison <=>
- alors en incluant le module Enumerable
- A obtient plus de 45 nouvelles méthodes basées sur les comparaisons et les parcours

:exemple => Enumerable

- So
  d'c
- A
- et
- alc
- A
  ba



A	all?, any?
C	collect, count, cycle
D	detect, drop, drop_while
E	each_cons, each_slice, each_with_index, entries, enum_cons, enum_slice, enum_with_index
F	find, find_all, find_index, first
G	grep, group_by
I	include?, inject, inject
M	map, max, max_by, member?, min, min_by, minmax, minmax_by
N	none?
O	one?
P	partition
R	reduce, reject, reverse_each
S	select, sort, sort_by
T	take, take_while, to_a, to_set
Z	zip

ction
ch
le
des
rcours

:implémentation

- Plusieurs versions de la machine virtuelle ruby :
    - jRuby : Java
    - IronRuby : .Net
    - MacRuby : Objective C
    - Rubinius : Ruby lorsque c’est possible, C++ sinon

:exemple => jRuby

:exemple => jRuby

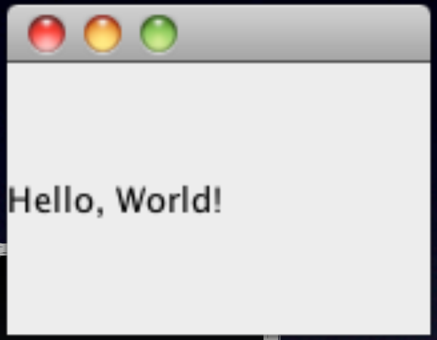
```
include Java

frame = javax.swing.JFrame.new()
frame.getContentPane().add(javax.swing.JLabel.new('Hello, World!'))
frame.setDefaultCloseOperation(javax.swing.JFrame::EXIT_ON_CLOSE)
frame.pack()
frame.set_visible(true)
```

:exemple => jRuby

```
include Java

frame = javax.swing.JFrame.new()
frame.getContentPane().add(javax.swing.JLabel.new('Hello, World!'))
frame.setDefaultCloseOperation(javax.swing.JFrame::EXIT_ON_CLOSE)
frame.pack()
frame.set_visible(true)
```


Hello, World!

:gems

- Equivalent des packages .deb pour Ruby
- S'installe avec leurs dépendances
- Plus de 14000 gems à ce jour
- Un repository de ces gems :
  - http://rubygems.org/

RAILS

:ruby_on_rails

- 2 Principes :
  - DRY : Don’t Repeat Yourself
  - Convention over Configuration
- REST : Representational state transfer
- MVC : Modèle - Vue - Contrôleur

:REST

- l’URI identifie clairement une ressource
- Operations HTTP :
    - GET
    - POST
    - PUT
    - DELETE
- Stateless - chaque opération est auto-suffisante

Démonstration

It’s gonna be Legend...ary :)

:idée

Mise en place d’une application de gestion de tâches pour des projets

:awesome

- Création d’un prototype d’application web et de son API en quelques minutes
- Scaffolding des ressources
- Abstraction de très haut niveau
- Des gems et des plugins couvrant tous les usages courant
- 1er déploiement en moins d’une minute sur Heroku

?

menant.maxime@gmail.com

maxime_menant