



Programming Ruby

Formation Ruby - Actiane
Samedi 16 Février 2008



1

D'où qu'il vient ?

- Japon, 1993, un homme : Yukihiro Matsumoto
- Have fun in programming !
- 1er version officielle en Janvier 1995
- Dernière version stable : 1.8.6 - Mars 2007

2

松本行弘

Yukihiro Ma
Né le 14 Av
Japonais
Diplômé de
Membre de
Saints des D



3

We need to focus on humans, on how humans care about doing programming or operating the application of the machines. We are the masters. They are the slaves.

Yukihiro Matsumoto

4

Les principes Ruby

- Une syntaxe très simple : principe de moindre surprise.
- Des variables non typées, dont la déclaration est facultative.
- Une gestion interne de la mémoire.
- Une application stricte de la règle "tout est objet".

5

The principle of least surprise is not for you only.

The principle of least surprise means principle of least my surprise.

And after two years of C++ programming, it still surprises me.

And it means the principle of least surprise after you learn Ruby very well.

Yukihiro Matsumoto

6

Tout objet.

- Toute donnée est un objet, y compris les types
- Toute fonction est une méthode et ce sont des objets
- Toute variable est une référence à un objet

7

Les interpreteurs

- Machine Virtuelle YARV (l'héritière)
- JVM de Sun (JRUBY)
- CLR de Microsoft (IronRuby et Ruby.NET)
- Machine Virtuelle Smalltalk (Rubinius)

8

Need ! Need !



Ruby One-click Installer :
<http://rubyforge.org/projects/rubyinstaller>



Paquet pour chaque distribution (deb, rpm...)
\$ wget ftp://ftp.ruby...tar.gz; tar -xf ruby*
\$./configure; make; make install



Ruby en standard (mais 1.8.2) depuis la 10.3
<http://homepage.mac.com/discord/Ruby/>
<http://locomotive.raum.org>

9

Jamais seul.

- Console interactive : irb (Interactive Ruby Shell)
- Debugger : ruby -r debug
- Profiler : ruby -r profile
- Couverture de code : rcov
- Gestionnaire de paquet : RubyGem

10

Jette ton oeil...

```
5.time do
  print "chaussette !"
end

return unless books.include? :harry_potter

['pain', 'nuttella', 'lait'].each do |food|
  print food.capitalize
end
```

11

Le terreau.

```
String :          Integer (BigNum,      Float :
"Dude ?"         FixNum)              3.5
"Ruby rox\n"     3.5                  23.2E-6
String.new('leaul') 5**20
                  4_344_566_322
Array :          Range :
[1, "deux", 9 / 3, Hash :             10.100
Person.new ]    {'I' => 1, 'II' => 2}  -3..5

FalseClass  TrueClass  NilClass
```

12

Comment t'enfile tes Strings ?

```
a = "\nThis is a double quoted string\n"
a = %Q{\nThis is a double quoted string\n}
a = <<BLOCK
This is a multi-line double quoted string
BLOCK
a = %/\nThis is a double quoted string\n/

a = 'This is a single quoted string'
a = %q{This is a single quoted string}
```

13

L'habit fait le moine.

```
NomDeClasse
nom_methode nom_variable
methode_qui_questionne?
methode_dangereuse!
@variable_d_instance
@@variable_de_classe
$variable_globale
$VARIABLE_GLOBALE
CONSTANTE AutreConstante
:symbole
```

14

Tout un symbole !

- Enregistre une référence mémoire pour une chaîne donnée
- Unique : une chaîne correspond toujours au même symbole
- Sert de clés, pour les noms de méthodes...

15

Bon à savoir...

- Tous les paramètres sont passés par référence sur les objets.
- Une valeur par défaut peut être spécifiée pour les paramètres.
- Parenthèses facultatives lors de l'appel d'une méthode.
- La valeur retournée est la dernière valeur évaluée (souvent absence de return)
- Les méthodes sont public, protected ou private

16

Les modules.

- Pseudo classe non instanciable
- Permet les espaces de nommage (::)
- Sert à étendre un classe
- Héritage multiple grace au mixin

17

Le bloc.

- C'est un fragment de code et un contexte d'exécution
- Classe Proc

```
bloc = Proc.new { |x, y| x+y }  
bloc.call(55, 45)
```

18

Bloc::Transaction

```
File.open("/tmp/bigfile.txt") do |fh|  
  fh.each_line { |l| puts l if l.size > 40 }  
end
```

19

Bloc::Closure

```
transfer = FileTransfer.new  
b1 = Button.new("Start") { transfer.start }
```

20

Passage de bloc, execution

```
def bis_repetita(&action)
  yield
end

bis_repetita { puts "Coco!"}

def bis_repetita2(&action)
  @bis_repetita = &action
end

bis_repetita2 { |value| puts "Salut " + value }
@bis_repetita.call "Coco!"
```

21

Getter - Setter

```
def create_set_and_get(initial_value = 23)
  closure_value = initial_value
  return Proc.new {|x| closure_value = x},
         Proc.new {closure_value}
end

setter, getter = create_set_and_get

setter.call(42)
getter.call # => 42
```

22

Bloc::Yield

```
class Fixnum
  def upto(max, &block)
    for i in self..max do
      yield i
      i = i.succ
    end
  end
end

1.upto(5) { |step| puts "Step #{step}!"}
```

23

If it walks like a duck and quacks like a
duck, I would call it a duck.

James Whitcomb Riley

In other words, don't check whether it IS-a duck:
check whether it QUACKS-like-a duck, WALKS-like-a
duck, etc, etc, depending on exactly what subset of
duck-like behaviour you need to play your language-
games with.

Alex Martelli

24

Exception, règle, toussa

Une exception est lancée avec le mot clé raise (avec un message optionel)

```
raise "This is a message"
```

25

Exception, règle, toussa

On peut aussi spécifier un type d'erreur

```
raise ArgumentError, "Name can't be blank"
```

Ou une instance d'exception

```
raise ArgumentError.new("Illegal arguments!")
```

26

Exception, règle, toussa

Faire une exception personnalisée

```
class ParseError < Exception
  def initialize input, line, pos
    super "Could not parse '#{input}' at
line #{line}, position #{pos}"
  end
end

raise ParseError.new("Foo", 3, 9)
```

27

Exception, règle, toussa

On peut attraper les exceptions pour les traiter comme on le souhaite

```
begin
  # Do something
rescue Exception
  # Handle exception
end

begin
  # ...
  rescue RuntimeError => e
    # handling, possibly involving e
  end
```

28

The primary duty of an exception handler is to get the error out of the lap of the programmer and into the surprised face of the user.

Provided you keep this cardinal rule in mind, you can't go far wrong

Verity Stob

29

Affectation parallèles

```
a = [1, 2, 3, 4]
```

```
b, c = a          # b == 1, c == 2
b, *c = a         # b == 1, c == [2, 3, 4]
b, c = 99, a      # b == 99, c == [1, 2, 3, 4]
b, *c = 99, a     # b == 99, c == [[1, 2, 3, 4]]
b, c = 99, *a     # b == 99, c == 1
b, *c = 99, *a    # b == 99, c == [1, 2, 3, 4]
```

30

Affectation imbriquée

```
b, (c, d), e = 1, 2, 3, 4
# b == 1, c == 2, d == nil, e == 3
```

```
b, (c, d), e = [1, 2, 3, 4]
# b == 1, c == 2, d == nil, e == 3
```

```
b, (c, d), e = 1, [2, 3], 4
# b == 1, c == 2, d == 3, e == 4
```

```
b, (c, d), e = 1, [2, 3, 4], 5
# b == 1, c == 2, d == 3, e == 5
```

```
b, (c,*d), e = 1, [2, 3, 4], 5
# b == 1, c == 2, d == [3, 4], e == 5
```

31

Classe de base de Ruby

Types de base

– Object
– String, Array, Hash,

Range, Regexp

– Integer, Fixnum, Bignum,
Float, Numeric

Fichiers

– IO, File, FileTest, Dir, Errno

Math

– trigo, log, exp...

Autres

– Kernel
– Exception
– Process, Signal
– Thread, ThreadGroup
– Time
– GC
– Marshal
– TrueClass, FalseClass,
NilClass

32

Bibliothèques Standards

Database	Concurrence/Distribution
- DBM, GDBM, SDBM	- dRuby, Rinda
Protocoles réseau	- Thread, Mutex, Monitor,
- Net (http, ftp, imap, pop, smtp, telnet), openSSL	Math
Web/XML	- Complex, Rational, mathn, Matrix, ...
- CGI, URI, REXML, RSS,	Lib. externes
SOAP, XMLRPC, YAML,	- DL
Webrick	Et beaucoup d'autres...

33

Ressources

<http://www.ruby-lang.org/>
<http://ruby-doc.org/>
<http://ruby-doc.org/docs/ProgrammingRuby/>
<http://ruby-doc.org/docs/UsersGuide/rg/>
<http://qa.pojignantguide.net/>
<http://www.math.umd.edu/~dcarrera/ruby/0.3/>
<http://ruby-doc.org/core/>

34

Coding in Ruby makes me happy because it's one of the shortest paths between my brain and a computer

Dave Thomas

35

Merci !
Des questions ?



36