Introduction à la programmation Ruby sous GNOME

Libr'East 2004

Laurent Sansonetti - lrz@gnome.org

Plan

- Ruby
- GNOME
- Ruby-GNOME2
- Conclusions

Ruby / Introduction

Ruby est un langage de programmation:

- orienté humain ;
- orienté objet ;
- interpreté ;
- libre (double licence GPL/Ruby) ;
- (de plus en plus) populaire ;
- (de plus en plus) utilisé.

La dernière version stable de Ruby est 1.8.1 (01-2004).

Ruby / Principes

Deux principes:

1. Tout est un objet

```
123                             # -> Fixnum
1_000_000_000_000               # -> Bignum
3.14159265                      # -> Float
"Hello Ruby!"                   # -> String
[ 1, 2 ]                        # -> Array
{ 1 -> "un", 2 -> "deux" }      # -> Hash
/(\w+):(\d+)/                   # -> Regexp
Fixnum                          # -> Class
```

2. Toute procédure est une méthode

Ruby / Bonjour Monde

```ruby
# ichi
def bonjour
  puts 'Bonjour Monde'
end
bonjour

# ni
Kernel.puts("Bonjour Monde")

# san
$stdout.puts String.new("Bonjour Monde")
```

Ruby / Blocs de code

Un bloc de code:

- est une suite d'expressions délimitée par `do...end` ou `{...}` ;
- peut accepter un ou plusieurs arguments ;
- renvoie une valeur ;
- est utilisé pour créer des:
  - procédures ;
  - itérateurs ;
  - transactions.

Ruby / Blocs de code / Procédures

```ruby
bonjour = Proc.new do |nom|
  puts "Bonjour " + nom
end
bonjour.call("Monde")
```

```ruby
def trois_fois
  yield
  yield
  yield
end
trois_fois { puts "fuga" }
```

Ruby / Blocs de code / Itérateurs

```ruby
10.times { puts "Bonjour Monde" }
```

```ruby
noms = [ "Sophie", "Marc", "Christophe", "Jean", "Denis" ]
noms.sort.each do |nom|
  puts "bonjour " + nom
end
```

Ruby / Blocs de code / Transactions

```ruby
File.open("foo") do |fichier|
  fichier.each_line do |ligne|
    puts "#{fichier.lineno}: #{ligne}"
  end
end
```

```ruby
# Methode traditionelle
fichier = File.open("foo")
...
fichier.close
```

Ruby / Dynamicité

```ruby
class Integer
  def minutes
    self * 60
  end
end

sleep 10.minutes # :-)
```

Ruby / Bibliothèques

Bibliothèque standard (stdlib), plus de 100 modules:

dl accès dynamique à l'éditeur de liens ;

drb développement distribué ;

net/* protocoles FTP, HTTP, IMAP, POP, SMTP, TELNET, . . . ;

openssl cryptographie ;

racc générateur d'analyseur (compatible YACC) ;

rdoc générateur de documentation ;

rexml analyseur / générateur XML ;

syck implémentation YAML ;

test/unit tests unitaires ;

webrick bibliothèque spécialisée services Web ;

xmlrpc client / serveur XMLRPC ;

etc...

Ruby / Outils

Outils standards:

ruby interpreteur Ruby:

```
$ ruby -e "puts 'Bonjour monde'"
Bonjour monde
$ ruby mon_programme.rb
```

irb interpreteur de commandes interactif:

```
$ irb --simple-prompt
>> "Bonjour monde"
=> "Bonjour monde"
>> 1 + 2 + 3
=> 6
>>
```

ri affiche la documentation:

```
$ ri Array
...
$ ri each
...
```

Ruby / Communauté

- Sites:

  http://www.ruby-lang.org Site officiel ;

  http://www.ruby-doc.org Documentation et articles en ligne ;

  http://www.rubycentral.com Livre complet en ligne ;

  http://www.rubyforge.org Projets libres/opensource.

- Discussions:

  ruby-talk@ruby-lang.org Liste de diffusion ;

  comp.lang.ruby Groupe de discussion ;

  #ruby-lang @ irc.freenode.net Canal IRC.

Plan

- Ruby
- GNOME
- Ruby-GNOME2
- Conclusions

GNOME / Plateforme

La plateforme de développement GNOME est un ensemble de bibliothèques C:

- libres (LGPL) ;
- documentées ;
- garanties de compatibilité ascendante ;
- dont le cycle de développement est planifié.

GNOME / Bibliothèques

Quelques bibliothèques:

GTK Construction d'interfaces graphiques ;

Libglade Chargement dynamique d'interfaces graphiques (XML) ;

GStreamer Développement multimédia (audio et vidéo) ;

GConf Enregistrement des préférences d'une application ;

GNOME-DB Accès aux bases de données ;

...

Toutes ces bibliothèques sont basées sur GLib/GObject.

GNOME / GLib

GLib apporte:

- des types basiques portables ;
- des threads ;
- le chargement dynamique de bibliothèques partagées ;
- des conteneurs (listes liées, hachages. . .) ;
- . . .
- *un framework orienté objet pour le langage C (GObject)*.

GNOME / GObject

GObject apporte:

- la notion de classe et d'interface aux structures ;
- un héritage simple aux classes ;
- un typage dynamique ;
- des signaux ;
- des propriétés aux classes ;
- des drapeaux et énumérations typés (*objets*).

Plan

- Ruby
- GNOME
- Ruby-GNOME2
- Conclusions

Ruby-GNOME2 / Introduction

Le projet Ruby-GNOME2:

- existe depuis 2002 ;
- est formé de 16 développeurs ;
- supporte 17 bibliothèques ;
- est traduit en Allemand, Anglais, Français, Italien, Japonais et Portugais.

La dernière version est 0.9.1 (03-2004).

Ruby-GNOME2 / Bibliothèques

Bibliothèques supportées:

Nom	Avancement	Tutoriel(s)?	Référence de l'API?	Exemples?
ATK	***	Non	Non	Oui
GLib	***	Non	Non	Oui
GDK	***	Non	Oui	Oui
GdkPixbuf	***	Non	Non	Oui
GTK	***	Oui	Oui	Oui
Libglade	***	Non	Non	Oui
GtkGLExt	***	Non	Oui	Oui
GConf	***	Oui	Oui	Oui
GStreamer	***	Oui	Oui	Oui
GnomeCanvas	***	Non	Non	Oui
Libgnome	***	Non	Non	Oui
Libgda	***	Oui	Oui	Oui
GnomeVFS	**	Non	Non	Oui
Libart	**	Non	Non	Oui
Pango	**	Non	Non	Oui
GtkHtml2	*	Non	Non	Oui
GtkSourceView	*	Non	Non	Oui

Ruby-GNOME2 / RBBR

RBBR (RuBy BRowser, Navigateur Ruby):

- inspecte les classes et modules:
  - Couche native: méthodes, constantes, héritage...
  - Couche GLib: signaux, propriétés, énumérations et drapeaux...
- affiche la référence de l'API ;
- parcourt les icônes et éléments prédéfinis de GNOME ;
- est ergonomique (HIG) ;
- est traduit en Allemand, Anglais, Biélorusse, Coréen, Espagnol, Français, Gallois, Italien, Japonais et Portugais.

La dernière version de RBBR est 0.6.0 (03-2004).

Ruby/GTK en action

```
require 'gtk2'

Gtk.init

bouton = Gtk::Button.new("Bonjour Monde")
bouton.signal_connect('pressed') { puts "Bonjour Monde" }

fenetre = Gtk::Window.new
fenetre.title = "Bonjour Monde"
fenetre << bouton
fenetre.signal_connect('destroy') { Gtk.main_quit }
fenetre.show_all

Gtk.main
```

Ruby/Libglade en action

```ruby
require 'libglade2'

class MonApplication
  def initialize
    xml = GladeXML.new('app.glade') { |conn| method(conn) }
    xml['label'].text = Time.now.to_s
  end

  def on_button_pressed
    puts "Bonjour Monde"
  end
end

Gtk.init; MonApplication.new; Gtk.main
```

Ruby/GConf en action

```
require 'gconf2'

NOM = "Ordinateur de " + ENV['USER']
BASE = '/apps/nautilus/desktop/'

client = GConf::Client.new
client[BASE + 'icon_name'] = NOM
client[BASE + 'home_icon_visible'] = true
```

Ruby/GStreamer en action

```
require 'gst'
Gst.init

source = Gst::ElementFactory.make("filesrc")
source.location = ARGV.first
decodeur = Gst::ElementFactory.make("mad")
sortie = Gst::ElementFactory.make("osssink")

source >> decodeur >> sortie

pipeline = Gst::Pipeline.new
pipeline.add(source, decodeur, sortie)
pipeline.play
while pipeline.iterate do end
pipeline.stop
```

Ruby/Libgda en action

```
require 'libgda'
Gda.init(__FILE__, "0.1")
Gda::Client.new.open_connection("ma_source") do |conn|
  conn.signal_connect('error') do |err|
    $stderr.puts err.description
    exit 1
  end
  cmde = Gda::Command.new("SELECT * FROM test", Gda::Command::TYPE_SQL
  modele = conn.execute(cmde)
  puts modele.columns.join(' | ')
  modele.each_row do |row|
    puts row.values.map { |val| val.to_s }.join(' | ')
  end
end
```

Plan

- Ruby
- GNOME
- Ruby-GNOME2
- Conclusions

Conclusions

Ruby:

1. permet de facilement coller ensemble des composants complexes ;
2. est suffisament mature pour du vrai développement ;
3. évolue constament ;
4. est sexy :-)

Questions

Des questions?...
Merci!

```
http://ruby-gnome2.sourceforge.jp
#ruby-gnome2 @ irc.gnome.org
lrz@gnome.org
```