JRuby for the Win

Ola Bini
computational metalinguist
ola.bini@gmail.com
http://olabini.com/blog

tisdag den 17 maj 2011
Logistics and Demographics
Your host

From Sweden
Language geek at ThoughtWorks, in Chicago
JRuby core developer
Creator of some languages (Ioke, Seph)
Author of Practical JRuby on Rails, coauthor of Using JRuby
Member of the JSR 292 EG
What is Ruby?

Dynamic, strongly typed, pure object oriented language

Interpreted

Open Source

Default implementation in C (called MRI)

Current versions: 1.8.7 and 1.9.2

Created in 1993 by Yukihiro ‘Matz’ Matsumoto

“More powerful than Perl and more object oriented than Python”

“The principle of least surprise”
Object orientation

Everything is an object:

```
Circle.new(4)   # => instance of Circle
"abc".length   # => 3
2.to_s         # => "2"
```

All objects are instances of classes:

```
1.class        # => Fixnum
(9**99).class  # => Bignum
"abc".class    # => String
/a/.class      # => Regexp
true.class     # => TrueClass
nil.class      # => NilClass
String.class   # => Class
Class.class    # => Class
```
Ruby - Blocks

Attach code to any invocation:

```ruby
# two syntaxes - curly brackets and do-end
[1, 2, 3].each { |n| puts "Number #{n}" }

[1, 2, 3].each do |n|
  puts "Number #{n}"
end

def foo(num)
  yield num
end
foo(1) { |e| puts "I got number #{e}" }
```
Ruby - Blocks

Removes dangerous repetition:

No need for external iterator:

```java
Iterator<String> iter = list.iterator();
while(iter.hasNext()) {
    System.out.println(iter.next());
}
```

The Ruby version shifts the responsibility:

```ruby
list.each do |element|
    puts element
end
```
Ruby - Blocks

Internalizes transactions:

```ruby
open(filename) do |file|
  # ... do something with the file
end

transaction do
  # some inserts
  # some selects
  # some updates
end
```
Ruby - Modules

As namespaces

```ruby
module Foo
  class Bar; end
  def self.hello    # a module method
    puts "Hello"
  end
end

Foo::Bar.new      # :: is used for nested levels
Foo::hello        # invoking the hello method
Foo.hello         # the same thing
```
Ruby - Modules

As mixins:

```ruby
module ShapeStuff
  def diameter
    2 * @radius
  end
end

class Circle
  include ShapeStuff
end

Circle.new(4).diameter #=> 8
```

Ruby - Enumerable

Implement `each`

Include `Enumerable`

You get:

- `all?`
- `any?`
- `collect`
- `detect`
- `each_with_index`
- `entries`
- `find`
- `find_all`
- `grep`
- `include?`
- `inject`
- `map`
- `max`
- `member?`
- `min`
- `partition`
- `reject`
- `select`
- `sort`
- `sort_by`
- `to_a`
- `zip`
Ruby - Comparable

Implement <=>

Include Comparable

You get:
<
<=
==
>
>=
between?
Ruby - Conclusions

Language succinctness

Malleability of language
  Agile way of working
  Quick turnaround
  Support for writing domain specific languages

Sources of innovation
  Rails
  Testing

Language power
JRuby

Implementation of the Ruby language

Java 1.5+

1.8.7 compatible (1.9.2 about 80%)

Open Source

Created 2001

Embraces testing (~50,000 tests)

Current version: 1.6.1

EngineYard

ThoughtWorks
Differences

Most compatible alternative implementation
Native threads vs Green threads
No C extensions (well, some)
No continuations
No fork
ObjectSpace disabled by default
Why JRuby?

Threading
Unicode
Performance
Memory
Explicit extension API and OO internals
Libraries and legacy systems
Politics
Simple JRuby
Java integration

Java types == Ruby types
Call methods, construct instances
Static generation of classes
camelCase or snake_case
.getFoo(), setFoo(v) becomes .foo and .foo = v
Interfaces can be implemented
Classes can be inherited from
Implicit closure conversion
Extra added features to Rubyfy Java
Ant+Rake
Maven+Gems
Swing

Swing API == large and complex
  Ruby magic simplifies most of the tricky bits

Java is a very verbose language
  Ruby makes Swing fun (more fun at least)

No consistent cross-platform GUI library for Ruby
  Swing works everywhere Java does
Swing - the direct approach

```java
import javax.swing.JFrame
import javax.swing.JButton

frame = JFrame.new("Swing is easy now!")
frame.set_size 300, 300
frame.always_on_top = true

button = JButton.new("Press me!")
button.add_action_listener do |evt|
  evt.source.text = "Don't press me again!"
  evt.source.enabled = false
end

frame.add(button)
frame.show
```
Swing - Cheri (builder)

include Cheri::Swing

frame = swing.frame("Swing builders!") { |form|
  size 300, 300
  box_layout form, :Y_AXIS
  contentPane { background :WHITE }

  button("Event binding is nice") { |btn|
    on_click { btn.text = "You clicked me!" }
  }
}

frame.visible = true
class ProfligacyDemo
  java_import javax.swing.*
  include Profligacy

  def initialize
    layout = "[<translate][*input][>result]"
    @ui = Swing::LEL.new(JFrame, layout) { |cmps, ints|
      cmps.translate = JButton.new("Translate")
      cmps.input = JTextField.new
      cmps.result = JLabel.new

      translator = proc { |id, evt|
        original = @ui.input.text
        translation = MyTranslator.translate(original)
        @ui.result.text.text = translation
      }

      ints.translate = { :action => translator }
    }
  end
end
Swing - MonkeyBars (tools)

GUI editor friendly (e.g. NetBeans “Matisse”)

Simple Ruby MVC based API

Combines best of both worlds
Testing

Ruby frameworks

Cucumber

JtestR
Google AppEngine

JRuby runs on it

JRuby-rack supports it

Google gems

Startup time

Merb, Ramaze and Sinatra easy options

Rails works
Mobile
Clojure STM
Using JRuby
Bringing Ruby to Java

Charles O Nutter,
Nick Steger,
Thomas Enebo,
Ola Bini, and
Ian Dees

Edited by Jacquelyn Carter

The Facets of Ruby Series
Questions?