

# JRuby: Up and Running!

**Charles Oliver Nutter and  
Thomas Enebo**

The JRuby Guys

Sun Microsystems



Except where otherwise noted, the content of this presentation is licensed under

the Creative Commons Attribution-Share Alike 3.0 United States License (<http://creativecommons.org/licenses/by-sa/3.0/us/>).

# Agenda

- JRuby overview
- Real-world JRuby
  - > Graphics and applets (that don't suck)
  - > Web applications
  - > GUI programming
- Interactive: what do you want to know?

# The JRuby Guys

- Charles Oliver Nutter and Thomas Enebo
- Longtime Java developers (10+ yrs each)
- Engineers at Sun Microsystems for 2 years
- Full-time JRuby developers
- Also working on JVM dynlang support
- Wide range of past experience
  - > C, C++, C#, Perl, Python, Delphi, Lisp, Scheme
  - > Java EE and ME, JINI, WS

# What Is Ruby

- Dynamic-typed, pure OO language
  - > Interpreted
  - > Open source, written in C
  - > Good: easy to write, easy to read, powerful, “fun”
  - > Bad: green threads, unicode support, libraries, “slow”
- Created 1993 by Yukihiro “Matz” Matsumoto
  - > “More powerful than Perl and more OO than Python”
- Very active community, wide range of apps
- Ruby 1.8.x is current, 1.9 is in development

# Ruby Growth (Gartner Projection)

# JRuby

- Java implementation of Ruby language
  - > “It's just Ruby!”
- Started in 2002, open source, many contributors
  - > Ola Bini, Marcin Mielzynsky, Nick Sieger, Vladimir Sizikov, MenTaLguY, Wayne Meissner
- Aiming for compatibility with current Ruby version
  - > Ruby 1.8.6
- Improvements on Ruby
  - > Native threading, better performance, many libraries

# Ruby Quick Tour: Pure OO

- Everything is an Object
  - > `Circle.new(4)` => instance of Circle
  - > `"abc".length` => 3
  - > `1.to_s` => "1"
- All Objects are instances of Classes
  - > `1.class` => Fixnum
- Single-Inheritance
- Object is base class of all classes

# Ruby Quick Tour: Basics

- Literals

- > Fixnum: 1

- > Float: 1.0

- > Bignum: 12345678987654321

- > String: "one" 'one' %Q[one] %q[one] ...

- > Multiline string ("here doc"):

- x = <<EOS

- extend across two  
lines

- EOS

- > Symbol: :one, %s[one]

- > Regexp: /^foo\w+.\*bar\$/, %r[^foo\$]

- > Array: [1, "ein", :ichi]

- > Hash: {:one => 1, :two => 2}

# Ruby Quick Tour: Basics

- String interpolation

```
> a = "foo"
> b = "bar#{a}baz" => "barfoobaz"
```
- Operator overloading

```
> def +(arg); ...
```
- Attributes

```
> class Foo
  attr_accessor :a
end
x = Foo.new
x.a = "hello"
puts x.a => "hello"
```

# Ruby Quick Tour: Duck Typing

- Dynamic typing
- “If it waddles like a duck and it quacks like a duck...”

```
def make_it_waddle(waddler)
  waddler.waddle
end
```

```
make_it_waddle(Duck.new)
make_it_waddle(Penguin.new)
make_it_waddle(Octopus.new)
```

- Runtime errors rarely happen
  - > Unit testing helps prevent them

# Ruby Quick Tour: A Simple Class

```
class Hello
  # initialize is Ruby's constructor
  def initialize(message)
    @message = message
  end

  def print
    # insert the @message into a string
    puts "Hello #{@message}"
  end
end

# construct a Hello object
hello = Hello.new("Devvxx!")
hello.print
```

# Ruby Quick Tour: Blocks/Closures

```
# two formats: braces {} and do .. end
[1, 2, 3].each {|number| puts "I see #{number}" }
[1, 2, 3].each do |number|
  puts "I see #{number}"
end
```

```
# methods that accept blocks
def foo
  yield "hello"
end
def foo2(&block)
  block.call("hello")
end
```

# Ruby Quick Tour: Modules

```
# A collection of objects
class MyProducts
  # Enumerable provides iteration methods
  include Enumerable

  # define an 'each' method that iterates
  def each
    # yield each element in turn
  end
end
```

```
list = MyProducts.new
list.select {|item| item.price > 5.00}
list.sort {|a, b| a.name <=> b.name}
list.max
```

# Ruby Quick Tour: RubyGems

- Ruby's packaging system
  - > Think CPAN, Maven, apt, rpm
- Shipped with JRuby
  - > Step 1: unpack JRuby
  - > Step 2 (optional): add 'bin' to PATH
  - > Step 3: bin/gem install <whatever\_you\_desire>
  - > You're ready to go!
- All major Ruby projects are in gems
  - > Look for 'gem install ....' in upcoming slides

# Ruby-Processing

- “Processing is an open source programming language and environment for people who want to program images, animation, and interactions.”
  - > Basically a cool Java library for 2D graphics
- Ruby-Processing wraps Processing with JRuby
  - > Cool, rubified 2D graphics environment for you
  - > Eye-candy demos for us
  - > Thanks to Jeremy Ashkenas for putting these together

# JMonkeyEngine

- JMonkeyEngine: 3D Scenegraph library
  - > OpenGL, Used Commercially

# **DEMO**

# **Pretty Graphics!**

# Web Applications: Ruby on Rails

- A Full-stack MVC web development framework
- Open Source (MIT), Many Contributors
- Written entirely in Ruby
- First released in 2004
- Growing popularity
  - > RailsConf attendance: 500, 1000, 2500 since 2006
  - > Four Rails books downstairs (and more Ruby books)
  - > Hundreds of job postings and growing fast

# Rails Precepts

- Convention over Configuration
  - > Why punish the common cases?
  - > Encourages standard practices
  - > Everything simpler and smaller
- Don't Repeat Yourself (DRY)
  - > Framework written around minimizing repetition
  - > Repetitive code harmful to adaptability
- Agile Development Environment
  - > No recompile, deploy, restart cycles
  - > Simple tools to generate code quickly
  - > Testing built into framework

# The Rails Way: Controllers

```
# app/controllers/person_controller.rb
```

```
class PersonController < ApplicationController
  verify :method => :post,
         :only => [:create, :update, :delete]

  def list
    @all_people = Person.find :all
  end
  alias :index :list

  def update
    @entry = Person.find(params[:id])
    @entry.update_attributes(params[:person])
    redirect_to :action => 'list'
  end
end
```

```
...
```

Rails  
Example

# The Rails Way: Views

```
<-- app/ views/ person/ list.rhtml -->
```

```
<table>
```

```
<tr>
```

```
<% for column in Person.content_columns %>
```

```
<th><%= column.human_name %></th>
```

```
<% end %>
```

```
</tr>
```

```
<% for person in @people %>
```

```
<tr>
```

```
<% for column in Person.content_columns %>
```

```
<td><%=h person.send(column.name) %></td>
```

```
<% end %>
```

```
<td><%= link_to 'Show', :action => 'show', :id => person %></td>
```

```
<td><%= link_to 'Edit', :action => 'edit', :id => person %></td>
```

```
<td><%= link_to 'Destroy', { :action => 'destroy', :id => person },  
:confirm => 'Are you sure?', :method => :post %></td>
```

```
</tr>
```

```
<% end %>
```

```
</table>
```

```
<%= link_to 'Previous page', { :page => @person_pages.current.previous } if  
@person_pages.current.previous %>
```

```
<%= link_to 'Next page', { :page => @person_pages.current.next } if  
@person_pages.current.next %>
```

Rails  
Example

# The Rails Way: Persistence

```
# connect to the database
```

```
ActiveRecord::Base.establish_connection(  
  :adapter => "mysql", :database => "mydb",  
  :host => "localhost", :username => "mydb_user",  
  :password => "foo" )
```

```
# create a model object
```

```
class Contact < ActiveRecord::Base  
end
```

```
# persist!
```

```
Contact.create "name" => "Charles Nutter",  
              "title" => "JRuby Developer"
```

```
Contact.create "name" => "Thomas Enebo", "title" => "JRuby Developer"
```

```
# query
```

```
Contact.find(:all).each {|c| puts c.name}  
nutter = Contact.find_by_name("Charles Nutter")
```

```
# update
```

```
nutter.title = "Dark Overlord of the Universe"  
nutter.save
```

Rails  
Example

# ActiveRecord Migrations

```
# db/migrate/001_add_user_table.rb
```

```
class AddUserTable < ActiveRecord::Migration
  def self.up
    create_table :users do |t|
      t.column :first_name, :string
      t.column :last_name, :string
      t.column :birthday, :date
    end
  end

  def self.down
    drop_table :users
  end
end
```

Rails  
Example

# **DEMO**

## **Walkthrough and Deploy**

# Production JRuby on Rails

- Sun's Kenai.com – project hosting site
  - > [www.kenai.com](http://www.kenai.com)
- Oracle's Mix – digg-like social customer site
  - > [mix.oracle.com](http://mix.oracle.com)
- ThoughtWorks' Mingle – collaborative project mgmt
  - > [mingle.thoughtworks.com](http://mingle.thoughtworks.com)
- Trisano – infectious disease tracking for US gov'ts
  - > [www.trisano.org](http://www.trisano.org)
- Many others in government, large biz, telecom

# Where is JRuby being used?

- Graphics and Games
  - > Ruby + graphics = cool
- JRuby on Rails
  - > Better deployment options, better performance
- GUI development
  - > Makes Swing much nicer to use, easier to handle

# GUI Programming

- Swing API is very large, complex
  - > Ruby magic simplifies most of the tricky bits
- Java is a very verbose language
  - > Ruby makes Swing actually fun
- No consistent cross-platform GUI library for Ruby
  - > Swing works everywhere Java does (everywhere)
- No fire-and-forget execution
  - > No dependencies: any script works on any JRuby install

# GUI Library Options

- Rubeus - gem install rubeus
  - > Builder-like DSL syntax
- Profligacy - gem install profligacy
  - > Rubified layout expression language
  - > Trivial event binding without listeners
- **MonkeyBars** - gem install monkeybars
  - > Leverage GUI builders
  - > MVC structure
- ...and 5+ others for Swing, SWT, and Qt

# DEMO

## Swing in Ruby

# Thank you!

- JRuby - [www.jruby.org](http://www.jruby.org)
  - > [wiki.jruby.org](http://wiki.jruby.org)
- GlassFish - [glassfish.dev.java.net](http://glassfish.dev.java.net)
  - > `gem install glassfish`
  - > Looking for bug reports, feature requests!
- Charlie's blog: [blog.headius.com](http://blog.headius.com)
- Tom's blg:  
[www.bloglines.com/blog/ThomasEEnebo](http://www.bloglines.com/blog/ThomasEEnebo)