# JRuby: Ready for ACTION!

**Charles Oliver Nutter and Thomas Enebo**
The JRuby Guys
Sun Microsystems

# Agenda

- JRuby overview
- JRuby enhancements and additions
- Swing programming
- Demo: Swing in Ruby
- Web applications
- Demo: JRuby on Rails
- Graphics and applets (that don't suck)
- Demo: Pretty graphics!

# The JRuby Guys

- Charles Oliver Nutter and Thomas Enebo
- Longtime Java developers (10+ yrs each)
- Engineers at Sun Microsystems for 1 yr
- 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

# JRuby

- Java implementation of Ruby language
  - "It's just Ruby!"
- Started in 2002, open source, many contributors
  - Ola Bini, Marcin Mielzinsky, Nick Sieger, Bill Dortch, Vladimir Sizikov, MenTaLguY
- Aiming for compatibility with current Ruby version
  - Ruby 1.8.6 patchlevel 111 (114 was just released)
- Improvements on Ruby
  - Native threading, better performance, many libraries

# JRuby 1.1 Released!

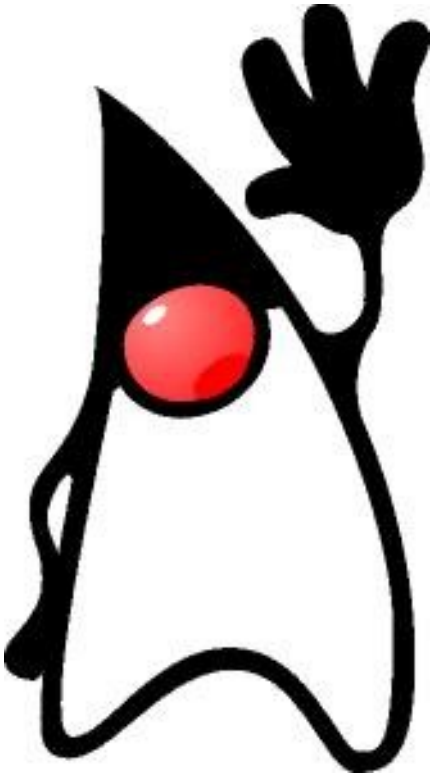
- Announced today: JRuby 1.1 is finished!
- Massive improvements over 1.0
  - Full compiler for Ruby code to JVM bytecode
  - Performance is many times better across the board
  - New Regexp impl with full Oniguruma features
  - New rewritten IO subsystem to parallel Ruby behavior
  - Reduced memory footprint
  - Best compatibility level ever
- Ready for production use (and already being used)
- Quick series of maintenance releases coming

# Compatibility

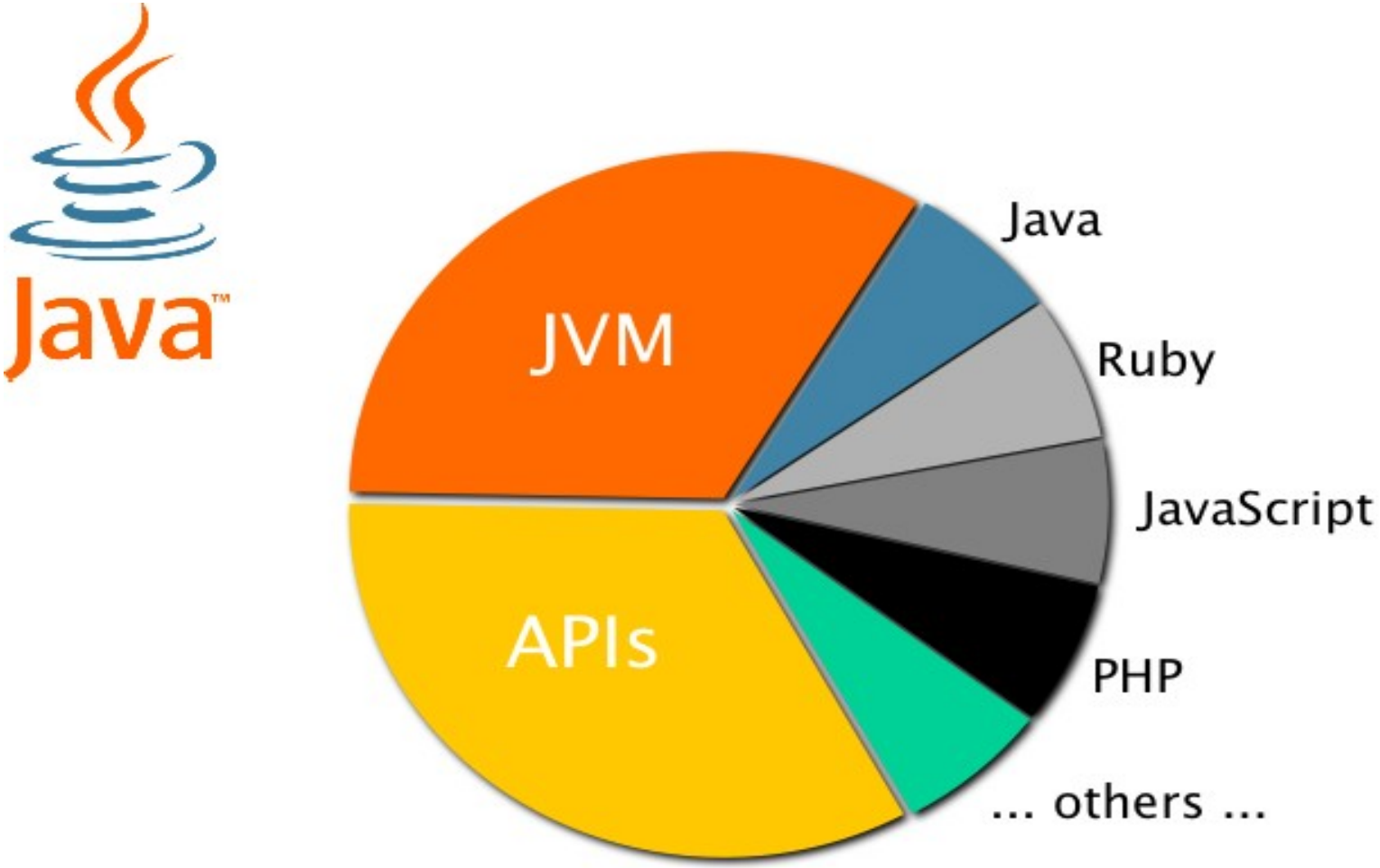
- Applications are king
  - Rails, Rubygems, Rake, Rspec, ...
- Testing rulez (~42,000 expectations/assertions)
  - Prevents regressions
  - Helps to better define Ruby
- Prevents fragmenting a community
  - Like Sapphire or ...Sapphire???

# Java == A Dirty Word

- “The answer is Java.  What is the Question?”



# Java != A Dirty Word



# Java != A Dirty Word

- Fantastic Virtual Machine
  - > Tuned for over a decade by an army
  - > Runs on virtually all os/hardware combos
  - > Dynamic optimizations (Hotspot)
  - > Keeps getting faster:

| | Java 5 | Java 6 | |
|---|---|---|---|
| Rexml | 10.9s | 7.41s | %32 |
| Hpricot | 4.06s | 2.59s | %36 |

# Java != A Dirty Word

- Fantastic Garbage Collectors
  - Compacting
  - Concurrent
  - Many tunables and choices

# Java != A Dirty Word

- Native threading
- Tools
  - IDEs (refactoring, debugging)
  - Profilers (instrumenting, sampling)
  - JMX (ask VM for stats)
- Libraries
  - Anything you can think of...
  - Write image_science in 60 lines of Ruby using Java 2D

# Where is JRuby being used?

- Swing GUI development
  - Makes Swing much nicer to use, easier to handle
- Ruby on Rails
  - Better deployment options, better performance
- Tooling for IDEs
  - JRuby's parser enables NetBeans, Eclipse, IntelliJ
- Graphics
  - Ruby + Processing = cool demos

# Swing 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

# Option 1: Direct approach

```
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
```

# DEMO
# Swing in Ruby

# Option 2: Cheri (builder approach)

```
include Cheri::Swing

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

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

frame.visible = true
```

# Option 3: Profligacy (targeted fixes)

```
class ProfligacyDemo
  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 = translation
      }

      ints.translate = {:action => translator}
    }
  end
end
```

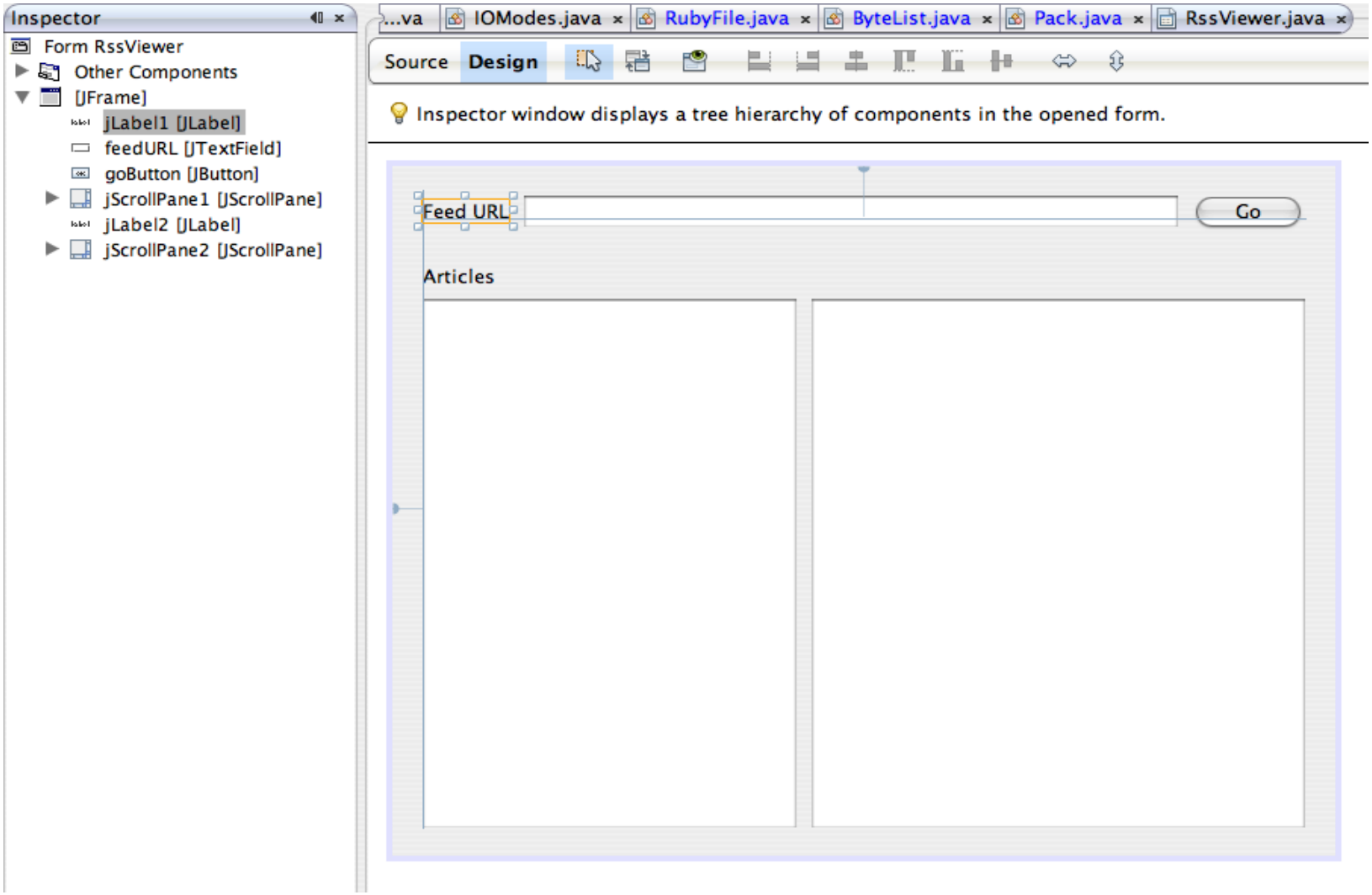
Profligacy
the world needs less swing

# Option 4: MonkeyBars (tool-friendly)

- GUI editor friendly (e.g. NetBeans “Matisse”)
- Simple Ruby MVC-based API
- Combines best of both worlds

monkeybars

# MonkeyBars + NetBeans Matisse



# MonkeyBars Controller

```
class RssController < Monkeybars::Controller
  set_view "RssView"
  set_model "RssModel"

  close_action :exit
  add_listener :type => :mouse,
     :components => ["goButton", "articleList"]

  def go_button_mouse_released(view_state, event)
    model.feed_url = view_state.feed_url
    content = Kernel.open(model.feed_url).read
    @rss = RSS::Parser.parse(content, false)

    model.articles = @rss.items.map {|art| art.title}
    model.article_text =
        CGI.unescapeHTML(@rss.items[0].description)
    update_view
  end
  ...
```

# Web applications

- Classic Java web dev is too complicated
  - Modern frameworks follow Rails' lead
- Over-flexible, over-configured
  - Conventions trump repetition and configuration
- Rails deployment is still a pain
  - You shouldn't need N processes!
- Rails performance should be better
  - JRuby has potential to be much faster

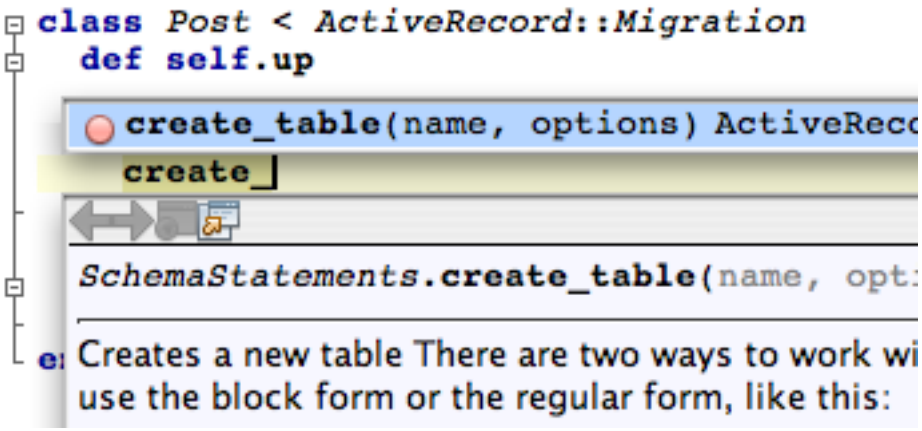
# Production JRuby on Rails

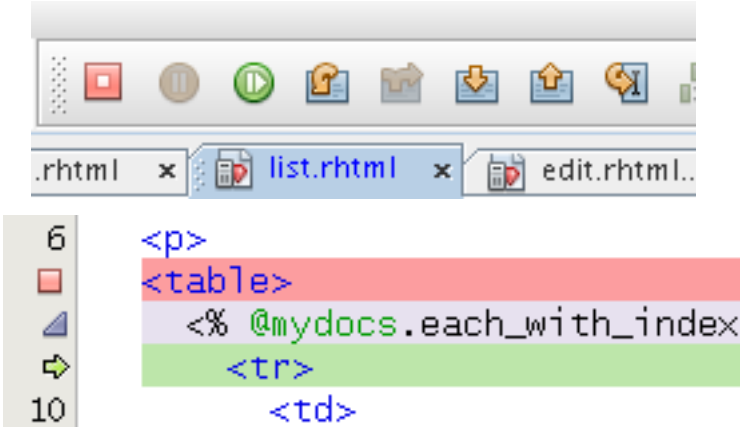
- Oracle's Mix – digg-like social customer site
  - mix.oracle.com
- Sun's MediaCast – file distribution portal
  - mediacast.sun.com
- ThoughtWorks' Mingle – collaborative project mgmt
  - mingle.thoughtworks.com
- Sonar – code/project analysis tool
  - sonar.hortis.ch
- More on the way!

# DEMO
# GlassFish Gem

# JRuby Enables Tooling

- JRuby's parser used by most Ruby IDEs
  - > NetBeans Ruby Support
  - > Eclipse RDT/RadRails/Aptana, DLTK, 3rd Rail
  - > IntelliJ
  - > Jedit
- Roman Strobl's NetBeans session at 13:00



# Graphics with 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

# DEMO
# Pretty Graphics!

# Thank you!

- Main JRuby page: www.jruby.org
- JRuby Wiki: wiki.jruby.org
- Charles Nutter
  - charles.nutter@sun.com
  - headius.blogspot.com
- Tom Enebo
  - thomas.enebo@sun.com
  - www.bloglines.com/blog/ThomasEEnebo

# JRuby: Ready for ACTION!

- **Charles Oliver Nutter and Thomas Enebo**
- The JRuby Guys
- Sun Microsystems