

JRuby: Bringing Ruby to the JVM™

Thomas E. Enebo, Aandtech Inc.

Charles Oliver Nutter, Ventera Corp

<http://www.jruby.org>



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/>).



JRuby Presentation Goal

Learn what JRuby is and how Ruby and JRuby will improve your Java world



Agenda

What are Ruby and JRuby

Ruby Features

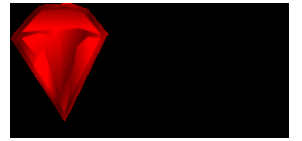
JRuby Features

Demonstrations

JRuby in the Wild

JRuby's Future

Conclusion



Agenda

What are Ruby and JRuby

Ruby Features

JRuby Features

Demonstrations

JRuby in the Wild

JRuby's Future

Conclusion



What is Ruby?

- Pure Object-Oriented Dynamically-typed Interpreted Language
- Open Source
- C Implementation is the “standard”
- Started in 1993 by Yukihiro ‘Matz’ Matsumoto
 - More powerful than Perl and more OO than Python
 - Guided by principle of least surprise
- www.ruby-lang.org



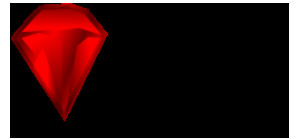
What is JRuby?

- 100% Java implementation of Ruby
- Open Source, GPL/LGPL/CPL licensed
- Not yet a Ruby-to-bytecode compiler
- Native-threaded
- Runs on Java SE versions 1.4.2 or higher
- Started in Fall of 2001 based on Ruby 1.6
- www.jruby.org



Java and Ruby Compared

```
public class Filter {
    public static void main(String[] args) {
        List list = new java.util.ArrayList();
        list.add("Tim"); list.add("Ike"); list.add("Tina");
        Filter filter = new Filter();
        for (String item : filter.filterLongerThan(list, 3)) {
            System.out.println( item );
        }
    }
    public List filterLongerThan(List list, int length) {
        List result = new ArrayList();
        for (String item : list) {
            if (item.length() <= length) { result.add( item ); }
        }
        return result;
    }
}
```

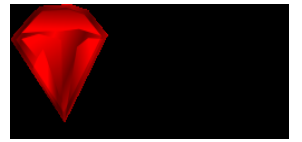


Java and Ruby Compared

Ruby!

```
list = ['Tim', 'Ike', 'Tina']  
list.select {|n| n.length > 3}.each {|n| puts n}
```

```
=> 'Tina'
```



Agenda

What are Ruby and JRuby

Ruby Features

JRuby Features

Demonstrations

JRuby in the Wild

JRuby's Future

Conclusion



Ruby Features (Blocks)

- Allows passing code around

- ```
def add_ten(base)
 yield(base + 10)
end
```

```
add_ten(5) { |num| puts num } => 15
```

```
add_ten(5) { |num| puts to_roman(num) } => XV
```

- Iteration done right

- ```
sum = 0;
collection = [1,2,4,3]
collection.each { |i| sum = sum + i }
puts sum => 10
```



Ruby Features (Duck-Typing)

- Type checking done at Runtime

```
class SlowList
  def find(criteria) ... end
end
```

```
class Tree
  def find(criteria) ... end
end
```

```
def search_with(search_type, criteria)
  search_type.find(criteria)
end
```

```
search_with(SlowList.new)
search_with(Tree.new)
```



Ruby Features (Open definitions)

- Never too late to add methods to a class

- ```
class Fixnum
 def prime?
 ...
 end
end
puts 13.prime? => true
```

- Or even change an existing one

- ```
class OnlyOnce
  def calc
    def calc; @cachedAnswer; end
    @cachedAnswer = expensive_one_time_calc()
  end
end
```



Ruby Features (Modules)

- Modules provide a namespace

```
module Color
  RED = [255,0,0];
end
puts Color::RED => [255,0,0]
```

- Modules provide mix-in inheritance

```
module Enumerable
  def sort; each {|i| ..sort logic..}; end
end
class Foo
  include Enumerable
  def each; ..yield all elements of Foo to block..; end
end
puts Foo.new.sort => sorted foo
```



Agenda

What are Ruby and JRuby

Ruby Features

JRuby Features

Demonstrations

JRuby in the Wild

JRuby's Future

Conclusion



JRuby Features (Java in Ruby)

- Import Java classes into Ruby

```
require 'java'  
include_class "java.util.Random"  
puts Random.new.nextInt() => 1064445193
```

```
include_class "java.lang.System"  
System.out.println("bar") => bar
```

```
include_class('java.lang.String') { |p, name|  
  "J#{name}" }  
JString.new('heh')
```



JRuby Features (Rubify Java)

- Automatic mapping of core types
 - Ruby Fixnum, Array, Hash, String... == long, Java List, Map, String...

```
include_class `java.util.Random`  
puts Random.new.nextInt() % 10 => 5
```
- Rubified method name aliases
 - ```
include_class `java.awt.Color`
puts Color::RED.getBlue => 0
puts Color::RED.blue => 0
puts Color::RED.get_blue => 0
```



# JRuby Features (Rubify Java Cont'd)

- Common Ruby methods added to core types
  - `java.lang.Comparable` defines '`<=>`' and includes Ruby's `Comparable` module
  - `java.util.List` defines '`each`', '`<<`' and includes `Enumerable` module
  - `java.util.Map`, `java.util.Set` define '`each`'
  - This list grows over time as we discover good fits



## JRuby Features (Java ↔ Ruby)

- Implement Java interfaces from Ruby

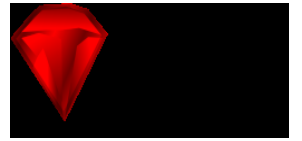
```
include_class "com.foo.MyInterface"
class MyImpl < MyInterface
 def bar # impl of public String
 bar();
 "hello"
 end
end
end
```
- Use Ruby from Java

```
MyInterface mine =
 (MyInterface) BSF.eval("MyImpl.new");
mine.bar(); => "hello"
```



## **JRuby Features (Miscellaneous)**

- Runs with Ruby's standard libraries
- Native threading: Ruby thread == Java thread



# Agenda

What are Ruby and JRuby

Ruby Features

JRuby Features

Demonstrations

JRuby in the Wild

JRuby's Future

Conclusion



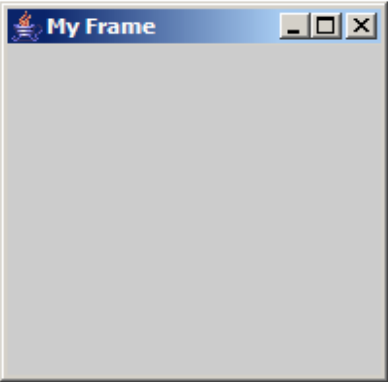
# IRB – Interactive Ruby

```
C:\JRubyWork\jruby>jirb
irb(main):001:0> x = 1
=> 1
irb(main):002:0> puts x + 1
2
=> nil
irb(main):003:0> def add_two(n)
irb(main):004:1> n + 2
irb(main):005:1> end
=> nil
irb(main):006:0> puts add_two(x)
3
=> nil
irb(main):007:0> class ThreeAdder
irb(main):008:1> def initialize(n)
irb(main):009:2> @n = n
irb(main):010:2> end
irb(main):011:1> def three_value
irb(main):012:2> @n + 3
irb(main):013:2> end
irb(main):014:1> end
=> nil
irb(main):015:0> ta = ThreeAdder.new(x)
=> #<ThreeAdder:0x11c2812>
irb(main):016:0> puts ta.three_value
4
=> nil
irb(main):017:0> _
```



# IRB cont'd – Java Integration

```
Command Prompt - jirb
C:\JRubyWork\jruby>jirb
irb(main):001:0> require 'java'
=> true
irb(main):002:0> include_class "javax.swing.JFrame"
=> ["javax.swing.JFrame"]
irb(main):003:0> f = JFrame.new("My Frame")
=> javax.swing.JFrame[frame0,144,0,0x0,invalid,hidden,layout=java.awt.BorderLayout,title=My Frame,resizable,normal,defaultCloseOperation=HIDE_ON_CLOSE,rootPane=javax.swing.JRootPane[,0,0,0x0,invalid,layout=javax.swing.JRootPane$RootLayout,alignmentX=null,alignmentY=null,border=,flags=385,maximumSize=,minimumSize=,preferredSize=],rootPaneCheckingEnabled=true]
irb(main):004:0> f.set_size(200, 200)
=> nil
irb(main):005:0> f.show
=> nil
irb(main):006:0> _
```

A screenshot of a Java Swing window titled "My Frame". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area of the window is empty and has a light gray background.




# IRB cont'd – Interactive Swing

```
C:\ Command Prompt - jrb
irb(main):003:0> f = JFrame.new("My Frame")
=> javax.swing.JFrame[frame0,144,0,0x0,invalid,hidden,layout=java.awt.BorderLayout,title=My Frame,resizable,normal,defaultCloseOperation=HIDE_ON_CLOSE,rootPane=javax.swing.JRootPane[,0,0,0x0,invalid,layout=javax.swing.JRootPane$RootLayout,alignmentX=null,alignmentY=null,border=,flags=385,maximumSize=,minimumSize=,preferredSize=],rootPaneCheckingEnabled=true]
irb(main):004:0> f.set_size(200, 200)
=> nil
irb(main):005:0> f.show
=> nil
irb(main):006:0> include_class "javax.swing.JButton"
=> ["javax.swing.JButton"]
irb(main):007:0> include_class "java.awt.event.ActionListener"
=> ["java.awt.event.ActionListener"]
irb(main):008:0> b = JButton.new "Push me!"
=> javax.swing.JButton[,0,0,0x0,invalid,layout=javax.swing.OverlayLayout,alignmentX=0.0,alignmentY=0.5,border=javax.swing.plaf.BorderUIResource$CompoundBorderUIResource@ecfe38,flags=296,maximumSize=,minimumSize=,preferredSize=,defaultIcon=,disabledIcon=,disabledSelectedIcon=,margin=javax.swing.plaf.InsetsUIResource[top=2,left=14,bottom=2,right=14],paintBorder=true,paintFocus=true,pressedIcon=,rolloverEnabled=false,rolloverIcon=,rolloverSelectedIcon=,selectedIcon=,text=Push me!,defaultCapable=true]
irb(main):009:0> class MyListener < ActionListener
irb(main):010:1> def actionPerformed(event)
irb(main):011:2> event.source.text = "Don't push me again!"
irb(main):012:2> end
irb(main):013:1> end
=> nil
irb(main):014:0> b.add_action_listener(MyListener.new)
=> nil
```



# IRB cont'd – Interactive Swing

```
Command Prompt - jrb
=> nil
irb(main):014:0> b.add_action_listener(MyListener.new)
=> nil
irb(main):015:0> f.content_pane.add(b)
=> javax.swing.JButton[, 0, 0, 0x0, invalid, layout=javax.swing.OverlayLayout, alignmentX=0.0, alignmentY=0.5, border=javax.swing.plaf.BorderUIResource$CompoundBorderUIResource@ecfe38, flags=296, maximumSize=, minimumSize=, preferredSize=, defaultIcon=, disabledIcon=, disabledSelectedIcon=, margin=javax.swing.plaf.InsetsUIResource[top=2, left=14, bottom=2, right=14], paintBorder=true, paintFocus=true, pressedIcon=, rolloverEnabled=false, rolloverIcon=, rolloverSelectedIcon=, selectedIcon=, text=Push me!, defaultCapable=true]
irb(main):016:0> f.show
=> nil
irb(main):017:0>
```



The image shows two side-by-side Swing windows, each titled "My Frame". The left window contains a single button with the text "Push me!". The right window contains a button with the text "Don't push me again!". This visualizes the state change of the button after an interaction.



# Ruby on Rails

The screenshot shows a web browser window titled "Online Cookbook - Mozilla Firefox" displaying a list of recipes. The browser's address bar shows the URL `http://127.0.0.1:8080/recipe/list`. The page content includes a table with two columns: "Recipe" and "Category". The table lists two recipes: "Fizzy Water (delete)" under "Beverages" and "Steak (delete)" under "Main Dishes". Below the table are links for "Create new recipe", "Show all recipes", and "Show all categories".

In the foreground, an Emacs editor window titled "emacs: list.rhtml" shows the source code for the page. The code uses ERuby to generate the table structure and includes a loop to iterate over recipes, displaying their titles and categories with delete links.

```
list.rhtml | RAILS DEMO | new.rhtml
<table border="1">
<tr>
 <td width="40%"><p align="c">
 <td width="20%"><p align="c">
 <td width="20%"><p align="c">
</tr>

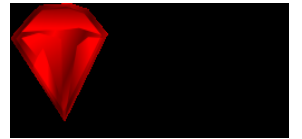
<% @recipes.each do |recipe|
 <% if (@category == nil) ||
 <tr>
 <td>
 <%= link_to recipe.title, :action =>
 :id => recipe

```



# Ruby on Rails

- JDBC activerecord adapter
- Deployment to Java servers an easier sell
- Possibilities to talk straight to Java services
- JRuby plans on supporting Rails around the end of this summer



# Agenda

What are Ruby and JRuby

Ruby Features

JRuby Features

Demonstrations

JRuby in the Wild

JRuby's Future

Conclusion



# JRuby in the Wild

- RDT
  - Ruby Development Tools for Eclipse
  - Uses JRuby's Ruby parser and AST
  - [www.rubypeople.org](http://www.rubypeople.org)
- Rad Rails
  - RDT-based Ruby on Rails IDE
  - [www.radrails.org](http://www.radrails.org)
- JEdit
  - A Programmer's Text Editor
  - Uses JRuby's Ruby parser and AST
  - [www.jedit.org](http://www.jedit.org)
- DataVision
  - Open Source Report Writer
  - Ruby as default formula language
  - [datavision.sourceforge.net](http://datavision.sourceforge.net)



# Agenda

What are Ruby and JRuby

Ruby Features

JRuby Features

Demonstrations

JRuby in the Wild

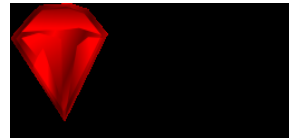
JRuby's Future

Conclusion



# New JRuby Design

- Iterative interpreter engine
- Heap-allocated Ruby stack frames, scopes
- M:N threading with thread, IO schedulers
- Compiled methods will “trampoline”
- Mixed-mode like HotSpot; dynamic optimization
- Pluggable core modules and subsystems
- Integrated Java/Ruby online debugging



# JRuby's Future

- Short Term
  - Better Compatibility with Ruby 1.8
  - Expanding Application Support
  - Continuation Support
  - Optimizations for Current Interpreter (Fast)
- Medium Term
  - Incorporating Ruby 2.0 features
  - M:N Threading
  - Multi-VM Support
  - “JRuby Bytecode” Interpreter (Faster)
- Long Term
  - Compilation to Java Bytecode (Fastest!)
  - Tail-Call Optimization



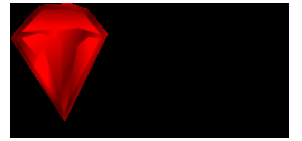
# Challenges for the Future

- Performance must be improved
  - Much slower than C Ruby currently
  - ...but not many optimizations yet
  - ...and no compilation yet
  - “Fast enough” is not always fast enough
- Keep JRuby Working
  - Refactored and redesigned code must not regress
  - Existing library of tests helps avoid regression
  - Slow process of encapsulating, reimplementing
  - Much more challenging



# Challenges Cont'd

- JVM and Ruby Incompatibilities
  - Continuations require stack manipulation (longjmp)
  - Ruby's Threads allow stop, kill, critical sections
  - Bindings, eval allow executing code in other contexts
  - Class, module defs always open, always mutable
  - System calls, signals, fork, symlinks, etc not possible
- JRuby isn't our day job
  - Contributors help immensely
  - Always more to do than time available



# Agenda

What are Ruby and JRuby

Ruby Features

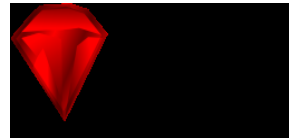
JRuby Features

Demonstrations

JRuby in the Wild

JRuby's Future

Conclusion



# Conclusion

- Ruby is an elegant, powerful language
- JRuby gives Ruby Java's capabilities
- JRuby gives Java Ruby's capabilities
- Another tool for the toolbox
- JVM, like .NET CLR, can support many languages
- Ready for use today, and great things in future



## For More Information

- Project Homepage: [www.jruby.org](http://www.jruby.org)
- Ruby Homepage: [www.ruby-lang.org](http://www.ruby-lang.org)
- Charles's Blog: [headius.blogspot.com](http://headius.blogspot.com)
- Tom's Blog:  
[www.bloglines.com/blog/ThomasEEnebo](http://www.bloglines.com/blog/ThomasEEnebo)
- *Programming Ruby*, by Dave Thomas
- jruby-user and jruby-devel mailing lists