

The Road to Babel: Bringing the JVM Language Implementers Together

Charles Oliver Nutter
JRuby Core Developer
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

- Introduction
- Language Survey
 - > Overview, example, status and how you can help
- Case Study: JRuby
- Collaboration
- Your Future

Who Am I

- Charles Oliver Nutter <charles.nutter@sun.com>
- JRuby co-lead
 - > Compiler and runtime work
- Language enthusiast
 - > C, C++, Pascal, VB, Java, Ruby...
- Bringing implementers together
- Helping to drive JVM changes

Language Survey

- The JVM is multilanguage today
 - > 200+ implementations: more than any other VM
 - > Dozens of languages: nearly all major ones
 - > Some quality, some not, some alive, some dead
- No language is impossible
 - > Functional languages most “different” from Java
 - > Dynamic languages hardest to optimize
- No language is too weird
 - > LOLCODE – Programming the LOL way.
 - > NestedVM – MIPS assembly to JVM bytecode

Java Example

```
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.GridLayout;
import java.awt.event.ActionListener;

public class MyApp {
    public static void main(String[] args) {
        JFrame frame = new JFrame("My Frame");
        final JButton button = new JButton("Press Me!");

        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                button.setText("Hello Java!");
            }
        });

        frame.add(button);
        frame.setLayout(new GridLayout(2, 2, 3, 3));
        frame.setSize(300, 80);
        frame.setVisible(true);
    }
}
```

Functional Languages

Clojure

- <http://clojure.sourceforge.net/>
- A dynamic, compiled, functional dialect of Lisp for the JVM, with strong concurrency support.
 - > Dynamic typing, with optional type hints
- “substantially easier to read and write”
 - > Vector, list, map literals
 - > “far fewer parentheses”
- “designed to tackle the problems for which you might ordinarily use Java”
 - > Uses Java's type system, integrates well

Clojure Example

```
(import '(javax.swing JFrame JLabel JTextField JButton)
        '(java.awt.event ActionListener)
        '(java.awt GridLayout))
(let [frame (new JFrame "My Frame")
      my-button (new JButton "Press me!")]
  (. my-button
    (addActionListener
      (proxy [ActionListener] []
        (actionPerformed [evt]
          (setText "Hello Clojure!"))))))
(doto frame
  (setLayout (new GridLayout 2 2 3 3))
  (add my-button)
  (setSize 300 80)
  (setVisible true)))
```

Clojure Status

- 2.5 yrs in development, released for 6 months
- Releases every 4-6 weeks; trunk is stable
- Lots of documentation, screencasts, tutorials
- “If you are interested in functional programming, Lisp, software transactional memory, agents, multimethods and other language features, in an efficient, dynamic environment, with great access to the Java frameworks, the best way to help Clojure is to try it out, and join the Clojure community on the Google group.

Kawa

- <http://www.gnu.org/software/kawa/>
- A framework for implementing high-level and dynamic languages
 - > An implementation of Scheme, with “very useful” Java integration
 - > Implementations of other programming languages, including XQuery (Qexo) and Emacs Lisp (Jemacs)
- One of the oldest, fastest JVM compilers (1996)

Kawa Example

```
(define-alias JFrame javax.swing.JFrame)
(define-alias JButton javax.swing.JButton)
(let ((frame (JFrame
  title: "My Frame"
  layout: (java.awt.GridLayout 2 2 3 3)))
  (my-button (JButton "Press me!")))
  (my-button:addActionListener
  (object (java.awt.event.ActionListener)
    ((actionPerformed evt)
      (my-button:setText " Fahrenheit")))))
  (frame:add my-button)
  (frame:setSize 300 80)
  (frame:setVisible #t))
```

Kawa Status

- Current release is 1.9.1 (early 2007)
- Summer 2008 should see 1.10
- Multiple projects using Kawa
- “a bit of an open secret”
 - > Trying to get the word out, get people interested
 - > With updates, good potential for new languages

OCaml-Java

- <http://ocamljava.x9c.fr/>
- OCaml for the JVM
 - > Very expressive language (Caml)
 - > Functional, imperative, and object-oriented
 - > Developed and distributed by INRIA, France's national CS research institute since 1985
- OCaml for Java
 - > Expressiveness of OCaml vast number of Java libraries
- Not funded or endorsed by the INRIA

OCaml-Java Example

```
open CadmiumObj
open Swing
```

```
let () =
  let frame = new JFrame (`String "My Frame") in
  let button = new JButton (`String "Press Me!") in
  button#addActionListener (new ActionListener (`Cd'wrap (object
    method actionPerformed (_ : jobject) =
      button#setText "Hello OCaml!"
    end)));
  ignore (frame#add (button :> jobject));
  frame#setLayout (new GridLayout (`IntIntIntInt (2|, 2|, 3|, 3|)) :>
    jobject);
  frame#setSize 300| 80|;
  frame#setVisible true
```

OCaml-Java Status

- OCaml 1.0
 - > Currently in beta; 1.0 final in May
 - > High compatibility level with Caml
 - > Self-hosting: can compile itself and Caml compilers
- Future releases
 - > 1.1: move from Java 1.5 to 1.6 (summer)
 - > 2.x branch: focus on performance issues
- One-developer free-time project
 - > Please contribute bug reports, doc errors

Scala

- <http://www.scala-lang.org/>
- Object-oriented: classes, traits, and mixins
- Functional: function objects, currying, pattern matching, case classes
- Statically-typed: type inference, rich type system
- Extensible: nice-looking DSLs, closure creation
- Fully interoperable with Java

Scala Example

```
import javax.swing.{JFrame, JLabel, JTextField, JButton}
import java.awt.event.{ActionListener,(ActionEvent)}
import java.awt.GridLayout
```

```
val frame = new JFrame("MyFrame")
val myButton = new JButton("Press me!")
myButton.addActionListener(
  new ActionListener {
    def actionPerformed(evt: ActionEvent) {
      myButton.setText("Hello Scala!")
    }
  }
)
```

```
def chain[T](what: T)(actions: T => Unit*) = actions.foreach(_(what))
```

```
chain(frame)(_.setLayout(new GridLayout(2,2,3,3)),
  _.add(myButton),
  _.setSize(300,80),
  _.setVisible(true))
```

Scala Status

- 2.7.0-final is current; 2.7.1.RC2 is latest
- 5000 downloads per month
- 20 emails per day
- 'Lift' web framework gaining traction
- Scala Lift-off event this Saturday

Dynamic Languages

Groovy

- <http://groovy.codehaus.org/>
- “agile and dynamic language for the JVM”
- Many features from Ruby, Python
 - > Closures, mutable classes, literal syntax
- Syntax very similar to Java
 - > Some Java will “just run”
 - > Idiomatic Groovy can be as terse as Ruby
- Many libraries built in
 - > GUI builder, testing, mocking

Groovy Example

```
import java.awt.event.ActionEvent
import javax.swing.JFrame
import javax.swing.JButton
import java.awt.GridLayout
import java.awt.event.ActionListener

def frame = new JFrame("My Frame")
final button = new JButton("Press Me!")

button.addActionListener( {
    button.text = "Hello Java!"
} as java.awt.event.ActionListener )

frame.add button
frame.layout = new GridLayout(2, 2, 3, 3)
frame.setSize(300, 80)
frame.visible = true
```

Groovy Status

- Groovy 1.5 production version
 - > In maintenance, 1.5.6 is current
 - > 1.5 focused on adding Java 5 features
 - > 35 bug fixes in 1.5.6 over 1.5.5 just a week later
- Groovy 1.6 in development
 - > Beta release in late April
 - > Performance improvements
 - > AST transformation
 - > Define annotations in Groovy; use from Groovy or Java

JRuby

- <http://www.jruby.org>
- Ruby for the JVM
 - > Dynamic typing, terse syntax, operator overloading, metaprogramming, very active and growing community
 - > Rails and apps like it run great
 - > Better performance, scaling, than all other impls
 - > Interesting case for JVM improvements
- Ruby for Java
 - > Simplifies complex libraries
 - > Integrates well with Java classes
 - > Rails for Java shops

JRuby Example

```
require 'java'  
import javax.swing.JFrame  
import javax.swing.JButton  
import java.awt.GridLayout  
  
frame = JFrame.new("My Frame")  
button = JButton.new("Press me!")  
  
button.add_action_listener { button.text = "Hello Ruby!" }  
frame.layout = GridLayout.new(2, 2, 3, 3)  
frame.add button  
frame.set_size 300, 80  
frame.visible = true
```

JRuby Status

- JRuby 1.1
 - > Production version, in maintenance
 - > 1.1.1 is current, 1.1.2 in mid-May
 - > Great performance, getting better in maintenance
 - > Most compatible with Ruby 1.8
- JRuby future
 - > More speed, more compatibility
 - > Redesigned Java integration for perf and completeness
 - > Sharing more libraries with other lang projects
 - > Support advanced JVM “in development” features early

Jython

- Python for the JVM
 - > Dynamic typed, simple class structure, some metaprogramming, whitespace-based structure
 - > Well structured, well-designed, “honest” language
 - > One of the larger communities
 - > Wide range of libraries and applications
 - > A better VM for Python?
- Python for Java
 - > Scripting Java libraries, etc
 - > Bring new apps, frameworks to the table (e.g. Django)

Jython Example

```
from javax.swing import JFrame, JLabel, JTextField, JButton
from java.awt import GridLayout

frame = JFrame('My Frame')
button = JButton('Press me!',
                 actionPerformed=lambda evt: button.setText('Fahrenheit'))

frame.setLayout(GridLayout(2, 2, 3, 3))
frame.add(button)
frame.setSize(300, 80)
frame.setVisible(True)
```

Jython Status

- Jython 2.2
 - > = Python 2.2, pretty far behind current Python 2.5/2.6
 - > Stable, surprisingly large userbase
 - > In maintenance now
- Jython trunk
 - > Roughly around 2.3 compatibility and moving fast
 - > New compiler, parser coming soon
 - > Native library ports in progress
 - > Already sharing a POSIX library with JRuby

Rhino

- JavaScript for the JVM
 - > Dynamic typed (like the others)
 - > Functional-like language you love to hate
 - > Very flexible function/type structures
 - > Pervasive on the client; gaining strength on server
 - > Excellent performance, one of the best
- JavaScript for Java
 - > Same basic scripting capabilities
 - > New solutions for old problems (Phobos web framework)

Rhino Example

```
importClass(java.awt.GridLayout)
importClass(javax.swing.JButton)
importClass(javax.swing.JFrame)
```

```
var frame = new JFrame("My Frame")
var button = new JButton("Press Me!")
```

```
button.addActionListener(function() {
    button.text = "Hello Java!"
})
```

```
frame.add(button)
frame.setLayout(new GridLayout(2, 2, 3, 3))
frame.setSize(300, 80)
frame.visible = true
```

Rhino Status

- Rhino 1.6R2 included in Java 6
 - > Interpreted by default in Java 6
 - > No E4X in Java 6
- Rhino 1.7
 - > R1
 - JavaScript 1.7 compatibility: list comprehensions, generators...
 - > R2
 - Support HtmlUnit and ECMA 262
 - More performance work
 - Parse tree API
- ECMAScript 4 / JavaScript 2 is being watched

Case Study: JRuby

Ruby for the JVM

- Why it's hard:
 - > Dynamic typing makes optimization tough
 - > Brings along its own type system
 - > Real-world dependencies on C extensions
 - > Difficult features: “eval”, continuations, byte-based String
- Why it's rewarding:
 - > Beautiful, elegant language
 - > New blood, new ideas for the Java world
 - > A better runtime
 - > Pushing the bounds of the JVM

JRuby

- Complex
 - > Mixed-mode: interprets, then compiles later
 - > Custom String, Array (List), Hash (Map), Regexp
 - > Generated code to avoid reflection overhead
 - Lots and lots
 - > Multiple VM: run many Ruby apps on one JVM
- What would make it easier
 - > Lightweight code generation and loading
 - > Faster-than-reflection method objects
 - > Toolchain for parsing, interpreting, compiling
 - > More flexible JVM type system

Collaboration

Making Babel a Reality

- Many common goals and challenges
 - > Why keep reinventing the wheel?
- We're all growing the platform
 - > We want the Java platform to succeed
 - > We're working hard to make sure it does
- We're all extending the platform
 - > Challenges are opportunities
 - > The platform isn't perfect, but it's Open
 - > It's up to you and us

JVM Languages Group

- URL
- Implementers from most major languages
- NUMBER OF MEMBERS, POSTS
- Discussions on parsing, compiling, threads, more
- Sharing information, ideas
- Discussing future plans for languages and JVM
- Great fun to read, participate
 - > Even if you're not a language person!

JVM Language Runtime

- URL
- DLR-inspired, just getting started
- Gathering the best libraries together
- Provide a fast track for implementation
 - > You might make a language someday
- Solve the hard problems once
 - > You might hate making a language some day
- The pieces are already out there!

Possible Bits for JRL

- Toolchain for language parsing, interp, compilation
 - > Kawa has some nice compilation bits
 - > Rhino has an optimizing compiler
 - > Some kind of ANTLR++?
- Language-agnostic type inference
 - > Repurpose Scala or Haskell?
 - > Roll our own for dynamic language optimization?
- Metaobject protocol (flexible class definition)
 - > Attila Szgedi's Dynalang MOP
 - > Groovy's MOP and lessons learned

Possible Bits for JRL

- Reflection-free method handles
 - > JRuby's “Invoker” generation; almost free handles
 - > John Rose's backported DVM handles
- Specialized types
 - > Marcin Mielzynski's JOni regex engine port
 - > Ola Bini's JvYAML parser
- So much to choose from...
 - > ...but the JRL is really out there already!

The Da Vinci Machine Project

- <http://openjdk.java.net/projects/mlvm/>
- OpenJDK Multi-language VM
- Feature testbed for future JDKs
 - Anonymous classloading (prototype working)
 - Lightweight method handles (prototype almost done?)
 - Optimized dynamic invocation (waiting on handles)
 - Continuations (proof-of-concept working, prototype coming)
 - Tail call optimizations (under research)
 - Tuples
 - <YOUR FEATURE HERE>
- Crazy cool stuff!

Your Future

Get Involved!

- All of these languages are open source
 - > Each would die without community
 - > You are the community
- You are not as dumb as you think
 - > You can learn, you can help!
 - > Don't be afraid to try, ask questions, read code
- You'll be a better person as a result
 - > Companies hire OSS developers
 - > Conferences pay OSS developers to travel, speak
 - > OSS is educational and fun, fun, FUN!

Get Involved!

- Languages are both easy and difficult
 - > Core types are easy
 - Run tests, fix bugs, write better methods (easy)
 - Vast bulk of work for JRuby is in core types
 - Mostly “just Java”, sometimes the language itself (easy, fun!)
 - > Interpreters are easy
 - Given a tree...walk it (easy)
 - Do stuff for each node (easy, maybe obscure)
 - > Compilers are difficult, but fun
 - Given a tree...walk it (easy)
 - Generate bytecode (tricky, obscure)
 - Generate optimized bytecode (hard, even more obscure)

Next Steps for You!

- It's a polyglot world out there
 - > Attend the sessions for these languages!
 - > Try them out, report bugs and feedback!
 - > Rewrite a piece of code or library of yours!
 - > Get on the mailing lists, and see how you can help!
- Q/A
 - > Thank you for coming!