

# JRuby on Rails: A Taste of Honey (for the enterprise)

Charles Nutter and Thomas Enebo



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/>).

What will we cover:

- JRuby 101 – Fluff 'n Stuff
- Justification! (or: Dear god, why?)
- JRuby on Rails – The Basics
- Deployment Options – How Easy Can They Be?
- Migration – Steps, Gotchas, and Pitfalls
- Beyond Rails – Creating Your Own Chimera

# Your Humble Hosts

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

Off We Go!

# JRuby 101

# A Floor Wax and a Dessert Topping

JRuby is:

- Ruby (1.8ish)
- A non-Java language for the Java platform
- A new way to look at Ruby and the JVM
- Helping to expand Ruby's reach
- Helping the world better understand Ruby
- Really cool

# A Floor Wax and a Dessert Topping

JRuby is not:

- An attempt to pollute or fork Ruby
- An admission that Java sucks
- The answer to every problem with Ruby
- An attempt to alter Ruby or add incompatible features
- Slow

# But It's Java, Too

```
include Java
import java.lang.ArrayList
import javax.swing.JFrame

list = ArrayList.new
frame = JFrame.new("Ruby SWINGS!")
list << frame
list.each {|f| f.set_size(200,200) }
```

# Java When You Want It

include Java

```
import java.lang.ArrayList
```

```
import javax.swing.JFrame
```

```
list = ArrayList.new
```

```
frame = JFrame.new("Ruby SWINGS!")
```

```
list << frame
```

```
list.each {|f| f.set_size(200,200) }
```

# Libraries You Need

```
include Java
```

```
import java.lang.ArrayList
```

```
import javax.swing.JFrame
```

```
list = ArrayList.new
```

```
frame = JFrame.new("Ruby SWINGS!")
```

```
list << frame
```

```
list.each {|f| f.set_size(200,200) }
```

```
include Java
import java.lang.ArrayList
import javax.swing.JFrame

list = ArrayList.new
frame = JFrame.new("Ruby SWINGS!")
list << frame
list.each {|f| f.set_size(200,200) }
```

# The Rest Is Stuff You Know

- Obvious libraries and gems preinstalled
  - RubyGems (obviously)
  - Rake
  - Rspec
- Mongrel supports JRuby
  - No remote gem, download from jruby-extras project
- Rails “just works”
  - Major orgs now investing in JRuby on Rails

# What Hath Man Wrought?

Justification

or

Dear god, why would you do such a thing?

- Supporting the Status Quo
  - JRuby support's Ruby 1.8's String
  - JRuby works with ActiveSupport::MultiByte
- Exposing Java Unicode
  - Java strings are all unicode
  - Java libraries are all unicode-friendly
  - Ruby and Java strings go back and forth safely
- Ruby 2.0 String support on the way...

- JRuby supports Ruby's Thread API
  - Even the ones we hate: critical=, kill, raise
- But most JVMs are native-threaded
  - One JRuby thread = one system thread
  - Use multiple cores, spin off long-running jobs
- JRuby also supports thread pooling
  - Keep threads warm for quick hits
  - Typically-sited use for green threads

What's the most cited pain point for Rails?

**Now It Gets Interesting**

# JRuby on Rails: The Basics

Many (most?) gems just work.

Rails (of course)  
RedCloth, BlueCloth  
Hpricot

Basically, anything pure Ruby (or with a JRuby port).

Typical Rails commands just work.

```
rails myapp  
jruby script/generate controller test hello  
rake db:migrate
```

Command-line dev looks basically the same.

Basic server use is the same.

```
jruby script/server => WEBrick  
gem install mongrel-1.0.1-jruby.gem  
jruby script/server => Mongrel
```

So development-time serving works about the same.

But a few things are different.

(otherwise what would be the point?)

## Big Difference #1: Database Support

Default pure-Ruby MySQL driver works...

```
development:  
  adapter: mysql  
  database: testapp_development  
  username: testapp  
  password: testapp  
  host: localhost
```

Different and faster, but just as easy.

```
development:  
  adapter: jdbc  
  url: jdbc:mysql://localhost/testapp_dev  
  username: testapp  
  password: testapp  
  driver: com.mysql.jdbc.Driver
```

## JNDI for connection pooling

development:

adapter: jdbc

jndi: java:comp/env/jdbc/MyAppPool

driver: mysql

# Broad Database Support

4th Dimension (4D RDBMS), ADABAS, ALLBASE SQL, Advantage Database Server, Advantage Ingres 2.6, [Apache Derby](#), BASIS, Birdstep RDM Server, CA-IDMS, CISAM, Cache, Centura SQLBase, Clipper, Cloudscape, CodeBase, D3, DABroker, [DB2](#), DB2 IBM AS/400 UNIX, DB2 Linux, DB2 OS/390, DB2 Windows, DBMaker, DL/I, DMSII, DaffodilDB, DataFlex/PowerFlex, Datacom, Domino, ECB, EDA, Empress RDBMS, Enscribe, Essentia, FireBirdSQL, FirstSQL/J Embedded Mobile, FirstSQL/J Enterprise Server, FormWeb, FoxBase, FoxPro, FrontBase, H2 Database, [HSQLDB](#), IBM AS/400, IDMS, IMS, Image/Turboimage, ImageSQL, Informix, Ingres, InstantDB, InterBase, JDataStore, [Java DB](#), LDAP, [MS Access](#), [MS SQL Server](#), [Mimer SQL](#), [MySQL](#), News Server, Nonstop SQL/MP, ODBC, OS/390 Sequential Files, OleDB-Provider, OpenBase, OpenIngres, [Oracle](#), PICK, Paradox, Pervasive.SQL, PointBase, [PostgreSQL](#), Primebase SQL Database Server, Progress, Quadcap, RDB, RMS, Recital, Redbrick Warehouse, SAP DB, SAS, SESAM/SQL-Server, SOLID Embedded Engine, SOLID SynchroNet, SQL/DS, SQLite, SUPRA Server SQL, SearchServer, Sequential, Solid Server, Sybase, Teradata RDBMS, Text (CSV, Tab separated etc.), ThinkSQL, TinySQL, TurboIMAGE, UNIFY, VFP, VSAM, XML, YARD-SQL, dBase, kdb, mSQL, xbase, xbase

# Does JDBC Support Mean Rails Support?

- ActiveRecord-JDBC doesn't support all of them
  - JDBC has no schema management API
  - Different databases quote differently
  - Some databases are missing features
- We focused on databases close to Rails, JRuby
  - MySQL passes 100% of ActiveRecord tests (yay!)
  - Derby 34F, PostgreSQL, 17F (out of 1000+ tests)
  - Others at varying levels, but coming along

## Big Difference #2: No Native Extensions\*

\*At least, not until ports are available

# Native Extensions

- Some ports done, some in progress
  - Mongrel: done
  - Hpricot: done
  - Database support: some done, some in progress
  - RMagick: in progress
- We're looking for porters and recommendations

## Big Difference #3: Command-line Performance

# Same Old Story

No, Java isn't slow...  
...except at startup.

# JRuby on Java 6 Server VM

```
~/work/jruby $ jruby SERVER bench_fib_recursive.rb
```

```
2.071000 0.000000 2.071000 ( 2.070000)
```

```
1.234000 0.000000 1.234000 ( 1.234000)
```

```
1.372000 0.000000 1.372000 ( 1.373000)
```

```
1.372000 0.000000 1.372000 ( 1.372000)
```

```
~/work/jruby $ ruby -v
```

```
ruby 1.8.5 (2006-12-25 patchlevel 12) [i686-darwin8.8.1]
```

```
~/work/jruby $ ruby bench_fib_recursive.rb
```

```
1.670000 0.010000 1.680000 ( 1.680365)
```

```
1.660000 0.010000 1.670000 ( 1.675206)
```

```
...
```

# Same Old Story

So don't expect stellar CLI performance.

# Building, Running, and Deploying an App

Yup, that's about it.  
Other stuff should just work.

And now for something completely different...

## Migration

It can be done!  
(it's basically just Ruby, after all)

## Step One: Database Support

- MySQL: 100% working, included in CI
- Derby/JavaDB: Nearly 100%; in CI; used in Mingle
- HSQL: Coming along; not in CI
- PostgreSQL: Down to a few failures; in CI
- Oracle: Being improved by community, not in CI
- MS SQL, DB2, others: Community driven, not in CI

## Migrations

- Working well, because they're part of Rails tests
- Tricky on some DBs that don't have all features

## Fixtures

- Also working well, part of Rails tests
- Usually bugs are YAML rather than DB issues

## Step Two: Native Extensions

You have a few options...

- Option #1: Use something else
- Option #2: Use an equivalent Java library
- Option #3: Port the library yourself
  - More work than #4, but Java's pretty easy
- Option #4: Port by wrapping a Java library
  - Our favorite...least amount of work

# Example Extensions

- Hpricot – Rake generator supports Java
- Mongrel – Ditto
- RMagickJr – Wraps builtin Java graphics support
- JParseTree – Pure Ruby, but uses JRuby AST
- OpenSSL – Complete port of OpenSSL (really!)
- YAML – Complete implementation in Java

## Step Three: Choosing a Deployment Option

# Tried and True: Mongrel

## Mongrel

- You know the good reasons, they still apply
- Some community use, but little testing
- You thought  $n$  Ruby processes took up memory!
- Not utilizing best of JRuby and JVM

# A Taste of the Old World

## Web Archive in App Server

- Multi-app, multi-request concurrency
- Most servers support transparent clustering
- Resource pooling
  - Database connections, cached results
  - Execution threads
  - Request handlers
- Access to the “good” Java EE features

# Whoa, What “Good” Java EE Features?

Java EE has a lot of bathwater, but a lot of baby

- Java Message Service
- Java Persistence API (good riddance Entity Beans)
- Java Transaction API
- Easy deployment, clustering, failover, management
- Scales great
  - Though Java's complexity makes it tricky

## A Grizzly/GlassFish v3 option

- Lightweight, gem-installable like Mongrel
- Concurrency, pooling, multi-app like WAR
- Completes the deployment story for JRuby
- Looking for suggestions and help!

## Beyond Rails

Creating your own Chimera

# Calling Java Libraries from Rails

## To The Community

**To the JRuby community and contributors:** Thank you! None of this would have been possible without your support and help.

**To the rest of you:** We're ready to help you get involved! Join the mailing lists, or email me directly at [charles.nutter@sun.com](mailto:charles.nutter@sun.com). Run your apps, report bugs, submit patches, get involved!

**Thank You!**

JRuby: [www.jruby.org](http://www.jruby.org)  
JRuby Wiki: [www.headius.com/jrubywiki](http://www.headius.com/jrubywiki)  
NetBeans: [www.netbeans.org](http://www.netbeans.org)  
NetBeans Ruby: [wiki.netbeans.org/wiki/view/Ruby](http://wiki.netbeans.org/wiki/view/Ruby)  
JRuby Extras: [rubyforge.org/projects/jruby-extras](http://rubyforge.org/projects/jruby-extras)  
Charlie's Blog: [headius.blogspot.com](http://headius.blogspot.com)  
Tom's Blog:  
[bloglines.com/blog/ThomasEenebo](http://bloglines.com/blog/ThomasEenebo)