

# Your First Day with JRuby on Rails

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

# It Really Is Ruby

ruby 1.8.5 (2007-04-29 rev 3539) [i386-jruby0.9.9]

**Based on Ruby 1.8.x**

**ruby 1.8.5** (2007-04-29 rev 3539) [i386-jruby0.9.9]

# Just Another Implementation

ruby 1.8.5 (2007-04-29 rev 3539) [[i386-jruby0.9.9](#)]

## 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) }
```

## But Rubified!

```
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) }
```

## Download from JRuby.org

JRuby

Java powered Ruby implementation



[Home](#)

[Download!](#) | [Issue Tracker](#) | [Source \(anonymous, developer, browsable\)](#) | [Mailing Lists](#) | [IRC](#) | [Contribute](#)

# Setting Up JRuby

## Build it yourself

```
~/work $ svn co --quiet http://svn.codehaus.org/jruby/trunk/jruby
~/work $ cd jruby
~/work/jruby $ ant -q
    [javac] Note: /Users/headius/work/jruby/src/org/jruby/runtime/
a deprecated API.
    [javac] Note: Recompile with -Xlint:deprecation for details.

BUILD SUCCESSFUL
Total time: 13 seconds
~/work/jruby $ bin/jruby -e "puts 'Hello, JRuby!'"
Hello, JRuby!
~/work/jruby $ bin/gem -v
0.9.1
~/work/jruby $ bin/jirb
irb(main):001:0>
```

# The Rest Is Stuff You Know

- RubyGems 0.9.1 is preinstalled
  - `gem install rails -y --no-ri --no-rdoc`
  - `gem install ActiveRecord-JDBC`
- Mongrel supports JRuby
  - No remote gem, download from jruby-extras project
- Rails “just works”
  - Lots of effort put into making Rails run well
  - Major orgs now investing in JRuby on Rails

# What Hath Man Wrought?

Justification

or

Dear god, why would you do such a thing?

# Is Ruby Perfect?

Granted, we all love Ruby.

Except unicode support.

And threading.

Scaling Rails could be easier.

I don't want to write it in C to get performance.

Ruby 2.0 promises solutions...but what about today?

# Unicode Support

- 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

## True Concurrent Threading

But threading and unicode  
are implementation details.

What is the biggest reason for JRuby?

**Now It Gets Interesting**

# JRuby on Rails: The Basics

# Differentiation

Lots of things are the same.

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.

# Creating the App and Running the server

In general, it's just Ruby on Rails!

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

Make sure to require the “jdbc” adapter

...

```
require 'jdbc_adapter'
```

```
Rails::Initializer.run do |config|
```

...

# CLASSPATH Returns!

You thought you'd seen the last of it...

```
export CLASSPATH=mysql-driver.jar
```

But there's an easier way.

```
cp mysql-driver.jar ~/work/jruby/lib
```

# Configuring the database

# Almost Every Database Supports JDBC

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

# Including Yours...

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

# Native Extensions

More and more extensions will support JRuby

...but for now pure-Ruby versions are best

## Big Difference #3: Command-line Performance

# Same Old Story

No, Java isn't slow.

# Same Old Story

...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.

**That's It?**

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

And now for something completely different...

**How Easy Can It Be?**

**Deployment**

# The Old Way Works

Option #1: Just use Mongrel

# The Old Way Works

- Mongrel is supported by JRuby
- Some bits won't work
  - Forking off subprocesses
  - Process management
- Some folks are running JRuby/Mongrel
- Most typical Mongrel setups *should* work

(What, you don't know if they'll work?)

# The Old Way Works

Mongrel works great, no argument there.

# The Old Way Works

Capistrano makes it a lot easier.

# The Old Way Works

But haven't we solved this problem before?

Option #2: Deploy to an app server

## Introducing GoldSpike!

- Rails plugin for building WAR files
  - That's “Web application ARchive”
- One plugin, pure Ruby (even works in MRI)
- Out comes a deployable WAR file
- Server-agnostic
  - But GlassFish fairly rocks ([glassfish.dev.java.net](http://glassfish.dev.java.net))

```
~/work/jruby/testapp $ jruby script/plugin install \  
    svn://rubyforge.org/var/svn/jruby-extras/trunk/ \  
        goldspike/plugins/goldspike  
~/work/jruby/testapp $ rake war:standalone:create  
(in /Users/headius/work/jruby/testapp)  
Assembling web application  
...  
Adding Java library jruby-complete-0.9.9  
...  
Adding web application  
Adding Ruby gem rails version 1.2.3  
...  
Creating web archive  
~/work/jruby/testapp $ ls -l testapp.war  
-rw-r--r--  1 headius  headius 6386396 Apr 30 18:47 testapp.war
```

# Customizing Your Webapp

config/war.rb:

```
add_gem 'tzinfo'  
maven_library 'mysql', 'mysql-connector-java', '5.0.5'
```

# Using GoldSpike with GlassFish

**Even Newer?**

**Option #3: A “Pack” in a Box?**

# Bridging the Gap

It's well understood  
that Rails developers  
may not like app servers.

# Bridging the Gap

Is there a middle ground?

# The Newest Way

```
~/work/jruby/testapp $ gem install glassfish-rails
~/work/jruby/testapp $ jruby script/server glassfish
=> Starting GlassFish
=> Rails application at http://localhost:8080
=> Admin services at http://localhost:4848
=> Clustering enabled
=> Connection pooling enabled
=> Load balancing enabled
=> Server Ready.
```

**Tease!**

**Not Yet.  
But Soon :)**

## Migration

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

The areas of concern are  
the (currently) unsupported features.

## Case Study: Mephisto

Mephisto is a kick ass web publishing system. It's a blog engine with some simple CMS-ish concepts (sections, pages), a very flexible templating system, and an aggressive caching scheme that takes advantage of your web server's best traits.

## Step One: Database Support

## MySQL

- Well-supported
- No known issues
- ActiveRecord tests 100%
- Included in our continuous integration

## Derby (JavaDB) and HSQLDB

- Working well (fewer failures every week)
  - Embeddable and shippable
  - Good target for small apps
- Included in our continuous integration

## PostgreSQL

- Down to a few failures
- Included in our continuous integration

## Oracle

- Being used by some in community
  - Starting to get our attention
- May be in continuous integration now

MS SQL, DB2, others

- Used in community
- Have not started to investigate
- Not part of continuous integration
  - Looking for folks to help here

## 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  
(sometimes pretty easy)

Option #4 (our favorite!)  
Port by wrapping a Java library  
(often even easier)

# 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

Two real options, a third in development.

# Choosing a Deployment Option

You must balance features with convenience.

# Choosing a Deployment Option

And we want to help build a better way.

# 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!

Putting it all together with Mephisto

## Beyond Rails

Creating your own Chimera

# There's a Whole Platform Out There

- Billions of lines of Java code out there
- Tens of thousands of deployments
- Thousands of libraries
- The Java platform is mature, trusted, reliable
  - ...and often boring!
- So why not combine two greats?
  - Ruby as the programming language
  - Java for the platform and libraries

- Best of all worlds
  - Ruby or Rails as the application layer
  - Java libraries, alone or as ported gems
  - Java-based services...legacy integration
  - The JVM
- Acceptable to today's enterprise
  - “Enterprise-ready” without losing the Ruby Way
  - To them: “it's just Java”, to you: “It's just Ruby”

# Calling Java from Rails

# Building a Better Toolset

- You all have your tried and true tools
  - Textmate, Emacs, VI(M), plus command line
  - You know them and you love them
- But you're missing something
  - Code completion?
  - Jump to declaration?
  - Rename variables?
  - Pop-up RDoc?

# NetBeans Ruby Support

- Best Ruby IDE available
  - code completion in dozens of ways
  - smart syntax highlighting
  - more and more refactorings
  - jump to method, variable, class declarations
  - pop-up RDoc
  - full project support for Ruby or Rails apps
  - everything else IDEs do: version control, etc

# NetBeans Ruby and Rails Support

# A New Beginning

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)