

# JRuby on Rails

Rich Manalang

# Who?

- Rich Manalang
- Oracle Applications Product Strategist/Hacker
- Rubyist through Rails
- JRubyist because of my day job

# What?

- Oracle Mix – you guessed it, a social network
- JRuby on Rails
  - Rails 1.2.6
  - JRuby 1.1 RC2
  - Loads of plugins
  - Deployed on “The Red Stack”
    - Oracle Enterprise Linux
    - Oracle Database 10g
    - Oracle Application Server 10g
    - Oracle Internet Directory
    - Oracle SSO

# Why?

- Bring customers closer to the employees who build the products they buy
- Solicit ideas from customer
- Allow for special interest networks and groups to form
- ... on Jruby?
  - Oracle is a Java shop. MRI wasn't really an option.

# Deployment Options

- Ruby
  - Mongrel clusters, Evented Mongrels, Thin?
  - Load balancer (Nginx, Apache, Lighttpd, Pound, HAProxy)
- JRuby
  - Mongrel clusters (load balanced just like with MRI)
  - Glassfish Gem – nice for small apps
  - WAR file deployment – optimal for production

**Live Demo**

# WAR File Deployment

- Mongrel clusters == JRuby runtimes
  - *n*-JRuby runtimes in one JVM instance

```
<context-param>
  <param-name>jruby.pool.minIdle</param-name>
  <param-value>2</param-value>
</context-param>

<context-param>
  <param-name>jruby.pool.maxActive</param-name>
  <param-value>4</param-value>
</context-param>
```

# WAR File Deployment

- How?
  - Warbler – Rails plugin

```
rich@rich-mac:~/dev/beast$ warble -T
rake config          # Generate a configuration file to customize your wa...
rake pluginize      # Unpack warbler as a plugin in your Rails application
rake version        # Display version of warbler
rake war            # Create beast.war
rake war:app        # Copy all application files into the .war
rake war:clean      # Clean up the .war file and the staging area
rake war:gems       # Unpack all gems into WEB-INF/gems
rake war:jar        # Run the jar command to create the .war
rake war:java_classes # Copy java classes into the .war
rake war:java_libs  # Copy all java libraries into the .war
rake war:public     # Copy all public HTML files to the root of the .war
rake war:webxml     # Generate a web.xml file for the webapp
```

# WAR File Deployment

- Pros
  - JRoR app becomes just another Java web app
    - Easy deployment, hot deployment if container supports it
    - Easy monitoring under most Java EE servers
    - Easy clustering under most Java EE servers
  - WAR file can contain other Java apps too!
    - Servlets, JSP, etc.
    - Servlet filters – for gzipping, header manipulation, caching, etc.
  - Full deployment easily script-able with Capistrano

# Ruby Issues

- Ruby 1.8: Green threading
  - No scaling across processors
  - C libraries won't/can't yield
  - One-size-fits-all scheduler
- Ruby 1.9: Native, non-parallel execution
- JRuby:
  - Ruby threads are Java threads
  - World class scheduler with tunable algorithms

# Ruby Issues

- Ruby 1.8: Partial Unicode
  - Internet connection applications MUST have solid Unicode
  - Ruby 1.8 provides very partial support
  - App devs roll their own: Rails Multi-byte
- Ruby 1.9
  - Drastic changes to interface and implementation
  - Performance issues
  - Each string can have its own encoding
- JRuby: Java Unicode

# Ruby Issues - Performance

- Ruby 1.8: Slower than most languages
  - 1.8 is usually called “fast enough”
  - ... but routinely finishes last
  - ... and no plans to improve in 1.8
- Ruby 1.9: Improvement, but not scalable
  - New engine about 1.5x for general applications
  - Only implicit AOT compilation
  - No JIT, no GC or threading changes
- JRuby: Compiler provides better performance

# Ruby Issues - Memory

- Ruby 1.8: Memory management
  - Simple design
  - Good for many apps, but not scalable
  - Stop-the-world GC
- Ruby 1.9: No change
  - Improved performance => more garbage
  - GC problems could multiply
- JRuby: World class Java GC's

# Ruby Issues - C

- Ruby 1.8 & 1.9: C language extensions
  - C is difficult to write well
  - Badly-behaved extensions can cause large problems
  - Threading and GC issues relating to extensions
  - Portable, but often with recompilation
  - No security restrictions in the system
- JRuby
  - Java extensions
  - GC and threading no problem

# Ruby Issues – Politics (which is why I'm here)

- Politics
  - “You want me to switch to what?”
  - “... and it needs servers/software/training?”
  - Potentially better with time (e.g., 1.9)
- Legacy
  - Lots of Java apps in the world
  - Extensive amount of Java frameworks
- JRuby solves both of these by running on top of Java
  - “Credibility by association”

# Performance

- No, it's not all that important
  - Until it is!
- JRuby 1.0 was about 2x slower than Ruby 1.8.6 (June 2007)
- JRuby 1.1 Beta 1 was about 2x faster
- JRuby trunk is 5x faster, often faster than 1.9
  - As a result, the JRuby team has stopped working with perf for now
  - ... but targeting Java performance next

# JRuby is used by...

- [mix.oracle.com](http://mix.oracle.com)
- [mediacast.sun.com](http://mediacast.sun.com)
- ThoughtWorks' Mingle

# References

- Ola Bini's Google Tech Talk
  - <http://www.youtube.com/watch?v=PfnP-8XbJao>
- JRuby on Rails Benchmarking Fun
  - <http://blog.erichsen.net/2008/02/17/benchmarking-fu>

# References

