

JRuby

Charles Oliver Nutter



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/>).

Part One

The Demon

Java

Terror

But why?

The power of Ruby compels you!

The Demon

I don't like the Java language.

The Demon

I don't like the Java language.

So don't use it.

The Demon

The Demon

I tried a Java application and it sucked.

The Demon

I tried a Java application and it sucked.

You can write crap in any language.

The Demon

The Demon

Java is slow.

The Demon

Java is slow.

No, it's not. Welcome to 2008.

The Demon

Java is not FOSS.

The Demon

Java is not FOSS.

Yes, it is. Welcome to 2008.

The Demon

Two words: Enterprise JavaBeans.

The Demon

Two words: Enterprise JavaBeans.

Ok, I'll give you that one.

The Demon

I can't get away from fscking Java!

The Demon

I can't get away from fscking Java!

Ahh, now we're getting somewhere :)

The Demon

The Demon

Don't throw out the baby.

Part Two

The Baby

The Baby

- The JVM
 - Highly tuned native JIT
 - Runs everywhere
 - Dynamic optimization at runtime
 - Faster every release

The Baby

The Baby

- Legendary garbage collector
 - Compacting avoids memory fragmentation
 - Various levels of concurrency
 - Infinitely tunable with solid defaults
- Fast allocation
 - Thread-local allocation buffers (no malloc)
 - Escape analysis moves data to stack

The Baby

- Native, fully-parallel threads
 - Writing a native-threaded VM is hard
 - No, I mean **really, really** hard.
 - Writing a native-threaded GC is harder
- Tooling
 - Refactoring, debugging, profiling in IDEs
 - Monitoring and management
- Libraries
 - Anything you can think of

JVM Optimizations 101

The HotSpot VM (Sun's JVM)

JVM Optimizations 101

- HotSpot is mixed-mode
 - Alternately interpreted or native
 - Code can flip back and forth
 - In both modes, gathers profile data
 - Profile data used to reoptimize

JVM Optimizations 101

- HotSpot can dynamically optimize
 - Compile JVM bytecode to native
 - Inline calls into one another
 - Optimistically optimize across calls
 - When guards fail, deopt and try again

JVM Optimizations 101

- HotSpot can eliminate code
 - Uncontended lock elimination
 - Promote unescaped heap to stack
 - Don't calculate unused results
 - Don't inline uncalled methods

Java is fast.

Java is fast.

Java's VM is fast,
and will continue to get faster.

JRuby runs on Java.

JRuby runs on Java.

JRuby runs on Java's VM.

JRuby runs on Java.

JRuby runs on Java's VM,
and uses it more and more efficiently.

JRuby bytes of bytecode:
Around 7.7MB total.

JRuby bytes of bytecode:
Around 7.7MB total.

Almost 1MB generated.

JRuby is fast.

JRuby is fast.

JRuby is fast because the JVM is fast.

JRuby is fast.

JRuby is fast because the JVM is fast,
and will get faster even if we end development.

JRuby is fast.

JRuby is fast because the JVM is fast,
and will get faster even if we end development,
and we're working our asses off anyway.

We write Java.

We write Java.

...So you don't have to.

Part Three

Janus

Janus

The two faces of JRuby

Java

Ruby

Janus

Is JRuby for the Java world or the Ruby world?

Janus

Is JRuby for the Java world or the Ruby world?

JRuby is a Ruby and a JVM language.

Part Four

Meat

Meat

- Cool stuff
 - JVM bytecode DSL
 - Java inline
 - FFI (call C libraries)
 - Debugging Ruby
 - Memory/perf profiling
 - image_voodoo
 - Duby (a Ruby-like static-typed language)
- Practical stuff
 - GUI development
 - 2D graphics
 - Rich web clients
 - Ruby tool support
 - Dead simple Rails deployment
 - Ruby 1.9 support
 - Testing Java

GUI Applications

GUI Applications

- A plague of choices
 - Cheri (a Swing builder)
 - Profligacy (fixes binding and layout)
 - MonkeyBars (tool-supported)
 - Swiby (web-targeted)
 - Rubeus (another Swing builder)
 - LimeLight (rich GUI framework)
 - Glimmer (SWT wrapper)
 - Qt::JRuby (Qt Jambi wrapper)
 - Ruby-processing (2D/3D graphics)

GUI Applications

But why?

GUI Applications

- GUI APIs are complicated
 - Ruby makes them much more palatable
- Java, C, C++ are too verbose
 - Ruby makes GUI dev actually fun
- No consistent cross-platform Ruby GUI
 - JVM-based APIs work (basically) everywhere
- No fire-and-forget distribution
 - Pack it up and ship it...even in a browser!

MonkeyBars

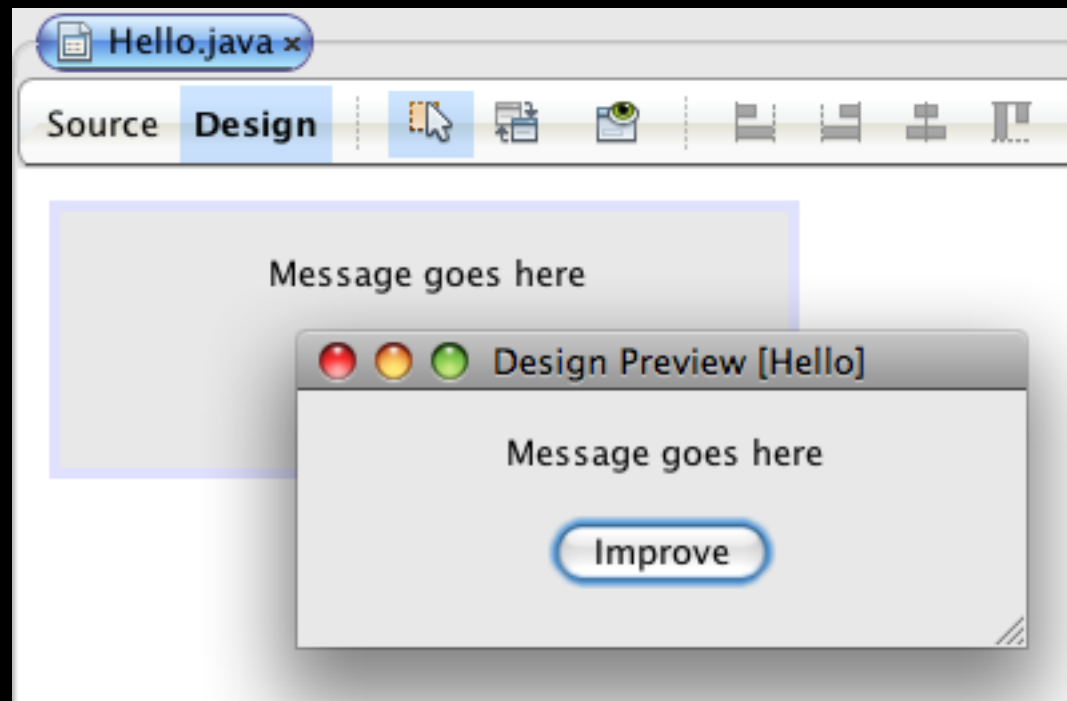
- Use Ruby for the application logic

```
class GreeterController < ApplicationController
  set_model 'GreeterModel'
  set_view 'GreeterView'
  set_close_action :exit

  def improve_button_action_performed
    model.message =
      "<html>But Swing with Monkeybars<br>is awesome!</html>"
    update_view
  end
end
```

MonkeyBars

- Use a layout tool for the GUI



Demo

Ruby-Processing

- Leverage Java libraries for graphics, etc

Demo

Web Applications

Web Applications

- Obviously, Java web dev is teh suck
 - Who would choose that over a Ruby framework?
- If you get past the suck, death by config
 - Ruby frameworks learned from Rails
- Ruby deployment is so 1995
 - How many processes? Are you serious?
- Rails and friends could be faster
 - The other languages are laughing at us

Rails

- Write Ruby...only Ruby...like you want
 - JRuby is “just Ruby” ...your app may “just work”
 - Caveats: native extensions, bad threading code
- Bundle your app in a single file
 - WAR file: “Web Archive”
 - Also WARless options like GlassFish/Jetty gems
- Deploy

Rails

What, you want more?

Rails

- Uniform database API
 - JRuby supports more DBs than Ruby
 - And they all work on all platforms
- Thread and connection pools
 - Scale up or down your app as needed
 - Thread-safe Rails? (or others like Merb)
- Full management and monitoring
 - It's the enterprise...they live for this stuff.