

Sleeping with the enemy

Integrating Java and Ruby: A primer

Murray Steele

10 Good Reasons

10 Good Reasons

- Java is entrenched

10 Good Reasons

- Java is entrenched
- Java is the “enterprise”

10 Good Reasons

- Java is entrenched
- Java is the “enterprise”
- Java is the town bike:
 - everyone knows “her”
 - “she’s” done everything you can think of and what you’ve not even heard of yet

Some 10 Good Reasons

- Java is entrenched
- Java is the “enterprise”
- Java is the town bike:
 - everyone knows “her”
 - “she’s” done everything you can think of and what you’ve not even heard of yet

I'm sold, what now?

- Several Choices:

I'm sold, what now?

- Several Choices:
 - JRuby - <http://www.jruby.org/>

I'm sold, what now?

- Several Choices:

- JRuby - <http://www.jruby.org/>

- Hessian - <http://www.caucho.com/hessian/>

I'm sold, what now?

- Several Choices:

- JRuby - <http://www.jruby.org/>

- Hessian - <http://www.caucho.com/hessian/>

- RJB - <http://arton.no-ip.info/collabo/backyard/?RubyJavaBridge>

What I use: YAJB

- [http://www.cmt.phys.kyushu-u.ac.jp/
~M.Sakurai/cgi-bin/fw/wiki.cgi?page=YAJB](http://www.cmt.phys.kyushu-u.ac.jp/~M.Sakurai/cgi-bin/fw/wiki.cgi?page=YAJB)

What I use: YAJB

- <http://www.cmt.phys.kyushu-u.ac.jp/~M.Sakurai/cgi-bin/fw/wiki.cgi?page=YAJB>
- (or, y'know, google)

- It's pure Ruby and pure Java

- It's pure Ruby and pure Java
- Works by:
 - Forking a JVM (not embedding or ruby running inside a JVM)

- It's pure Ruby and pure Java
- Works by:
 - Forking a JVM (not embedding or ruby running inside a JVM)
 - Communicates over XMLRPC or Binary protocol

Running Blackboxes

- Ruby telling Java what to do (simple version)

Running Blackboxes

```
#!/usr/bin/env ruby
require 'yajb/jbridge'
include JavaBridge

if !defined?(JBRIDGE_OPTIONS) then JBRIDGE_OPTIONS = {} end

JBRIDGE_OPTIONS[:classpath] = '$CLASSPATH:' +
                               '/Developer/Examples/Java/JFC/SwingSet2/SwingSet2.jar'

at_exit { JavaBridge.break_bridge }

jimport 'SwingSet2'
jimport 'java.awt.GraphicsEnvironment'

swingset =
  :SwingSet2.jnew(nil,
                 :GraphicsEnvironment.jclass.getLocalGraphicsEnvironment().
                 getDefaultScreenDevice().
                 getDefaultConfiguration())

# Need to do *something* to keep the main thread alive
loop {}
```

Playing with libraries

- Ruby telling Java what to do (complex version)

Playing with libraries

- Ruby telling Java what to do (complex version)
- It looks like the freak-child of Ruby and Java code ... *shiver*

Playing with libraries

```
# This is the standard xml+xsl-t pattern for using FOP.
# http://xmlgraphics.apache.org/fop/0.92/embedding.html
fopFactory = :FopFactory.jclass.newInstance

# the output file for the pdf
out = :BufferedOutputStream.jnew(:FileOutputStream.jnew(:File.jnew(ARGV.shift)))
fop = fopFactory.newFop(:MimeConstants.jclass.MIME_PDF, out)
factory = :TransformerFactory.jclass.newInstance

# the xsl-t file to transform the xml into xsl-fo
xslt = :StreamSource.jnew(:File.jnew(ARGV.shift))
transformer = factory.newTransformer(xslt)

# the xml file you want to generate a pdf from
src = :StreamSource.jnew(:File.jnew(ARGV.shift))
res = :SAXResult.jnew(fop.getDefaultHandler)
transformer.transform(src, res)
out.close

# Of course ... instead of just args on the command line for filenames
# to read / output to you could use java's StringReader objects to read
# from XML documents you've created with builder or some other ruby xml
# generation mechanism...
```

Lets get Dirrty

- Playing with complex libraries and extending Java with Ruby code

Lets get Dirrty

```
button1 = jnew :JButton, "send message to ruby"
action1 = jextend :ActionListener
2 action1.jdef(:actionPerformed) do |event| ## implementation by block
  puts "called by Java : #{text.getText}"
3 end
button1.addActionListener(action1)

button2 = jnew :JButton, "exit"
action2 = jextend :ActionListener
2 def action2.actionPerformed(event) ## implementation by singleton method
  wakeup_thread
3 end
button2.addActionListener(action2)

frame = jnew(:JFrame)
2 frame.addWindowListener(:WindowListener.jext() do |method, args|
  puts "#{method} called with #{args.nil? or args.empty? ? 'no args' : arg
  if method == 'windowClosing'
    wakeup_thread
  end
3 end)
```

**That last example
wasn't very
interesting was it?**

What about...

What about...
script/server tomcat?