# JRuby
## Java and Ruby together under one VM

Cliff Moon
Chariot Solutions

po-ru.com


# CRUSH THE PYTHON!

PRODUCED BY THE PEOPLE'S COMMITTEE FOR REVOLUTIONARY RE-EDUCATION

# What is JRuby?

- A 100% Java implementation of the Ruby 1.8.5 runtime.
- An embeddable scripting engine compatible with the BSF and the JSR 223 (Java 6) scripting frameworks.

# JRuby is not...

- intended to displace MRI, YARV, or Rubinius.
- interested in changing the syntax of the ruby language.

# How does JRuby differ from MRI?

- Threading
  - MRI uses green threads (due to change in ruby 2)
  - JRuby uses Java threads
- Continuations - Kernel.callcc
  - MRI supports them (possibly to change in ruby 2)
  - JRuby does not
- fork()
  - MRI supports this on posix platforms
  - JRuby does not

# How Do Ruby and Java Get Along?

# The Nuts and Bolts of Integration

- Java calling into Ruby
  - Ruby objects are just AST's
  - The JVM does not recognize Ruby classes
  - Proxies enable invocation
- Ruby calling into Java
  - Load Java objects via scripting framework
  - JRuby converts camelCase to under_score
  - Getters and setters are converted to accessors
  - Require can load JAR's

# Loading Jars

```
if RUBY_PLATFORM =~ /java/
  require 'find'

  Find.find("/my/lib/folder/") do |jar|
    if File.basename(jar) =~ /(.*\.jar)$/
      require jar
    end
  end
end
```

# A Ruby Message Listener

```
include Java
import javax.jms.MessageListener
class MyRubyMDB
    include MessageListener

    def onMessage(message)
        if message.respond_to? :text
            puts “I has message says: #{message.text}”
        end
    end
end
```

# Message Listener Driver

```
include Java
import javax.naming.InitialContext

conn_factory = InitialContext.new.lookup('java:comp/env/jms/ConnFact')
connection = conn_factory.create_connection
session = connection.create_session(false, 1)
consumer = session.create_consumer(session.create_topic('myTopic'))
consumer.message_listener = MyRubyMDB.new
connection.start
```

# Proxying Ruby Objects ( < Java 6)

```
Ruby runtime = Ruby.getDefaultInstance();

runtime.loadFile(new File("path/to/my_ruby_mdb.rb"));

Object rubyListener = runtime.evalScript("MyRubyMDB.new");

MessageListener listener = (MessageListener)
  JavaEmbedUtils.rubyToJava(runtime, (IRubyObject)
  rubyListener, MessageListener.class);
```

# Proxying Ruby Objects (Java 6)

```
ScriptEngineManager mgr = new ScriptEngineManager();

ScriptEngine engine = mgr.getEngineByName("jruby");

engine.eval(new FileReader("path/to/my_ruby_mdb.rb"));

Object rubyListener = engine.eval("MyRubyMDB.new");

MessageListener listener = ((Invocable)
  engine).getInterface(rubyListener,
  MessageListener.class);
```

# Passing in Objects (BSF)

```
BSFManager.registerScriptingEngine("ruby",
  "org.jruby.javasupport.bsf.JRubyEngine", new String[]
  { "rb" });

BSFManager mgr = new BSFManager();

mgr.declareBean(“passed_in”, “World”, String.class);

mgr.exec("ruby", "(java)", 1, 1, "puts \”Hello
  #{$passed_in}\”");


=> Hello World
```

# Passing in Objects (Java 6)

```
ScriptEngineManager mgr = new ScriptEngineManager();

ScriptEngine engine = mgr.getEngineByName("jruby");

engine.getContext().setAttribute("passed_in", “World”,
  ScriptContext.ENGINE_SCOPE);

engine.eval("puts \”Hello #{$passed_in}\””, context);


=> Hello World
```

# Arrays and Collections

- Java arrays are distinct from Ruby arrays
  - Conversion is not automatic
  - Convenience methods are provided
- Java collections are distinct from Ruby arrays
  - Conversion are automatic
  - Convenience methods are provided, though mostly unnecessary

# Array Conversions

```
irb(main):004:0> java = [10, 22, 33].to_java(:int)
=> #<#<Class:01x171cdc>:0x2c065 @java_object=[I@a531a9>
irb(main):005:0> java[2]
=> 33
irb(main):006:0> java.to_a
=> [10, 22, 33]
```

# Collection Conversions

```
irb(main):009:0> list = ArrayList.new([1, 2, 3, 4])
=> #<Java::JavaUtil::ArrayList:0x1d3edd @java_object=[1,
  2, 3, 4]>
irb(main):010:0> list.to_a
=> [1, 2, 3, 4]
irb(main):011:0> list.add_all([5, 6, 7])
=> true
irb(main):012:0> list.to_a
=> [1, 2, 3, 4, 5, 6, 7]
```

# Of Bytes and Strings

- Ruby has no concept of byte arrays
- Strings are used instead of byte arrays
- Java API's often take and return byte[]

# Byte[] -> RString

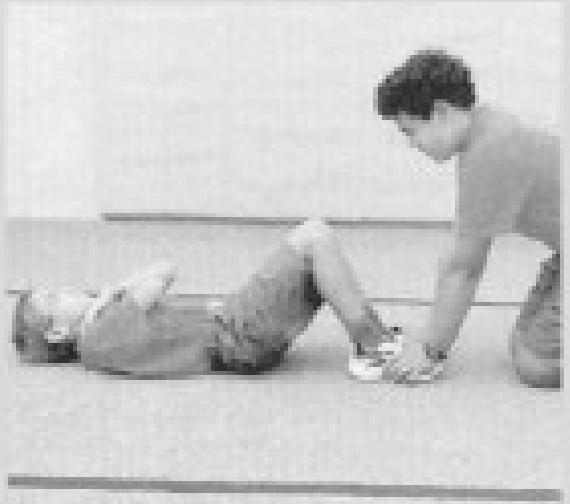
```
irb(main):026:0> String.from_java_bytes [83, 119, 101,
  101, 101, 101, 116, 46].to_java(:byte)
=> "Sweeeet."
```

# RString -> Byte[]

```
irb(main):027:0> "Frankenstein".to_java_bytes
=> #<#<Class:01xf8297b>:0x60efe7 @java_object=[B@df7713>
irb(main):028:0> "Frankenstein".to_java_bytes.to_a
=> [70, 114, 97, 110, 107, 101, 110, 115, 116, 101, 105,
  110]
```

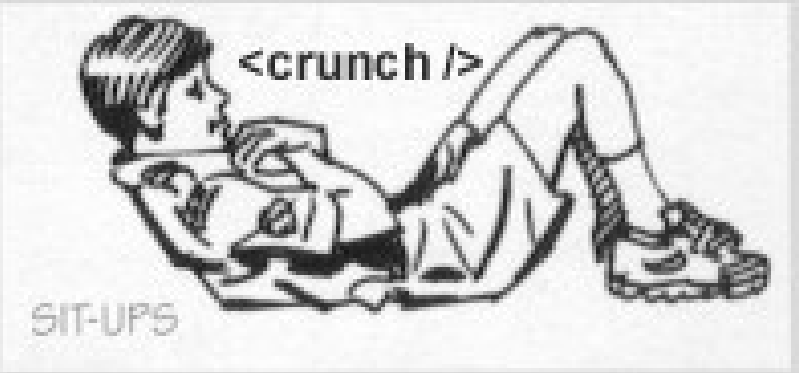
# So What Can JRuby Do For Me?

## Strength Tests



**XML Sit-Ups**

To assume the starting position, lie on your back with your knees in the angle bracket form. Your partner will act as the trailing slash. Cross your arms over your chest as if you are shielding yourself from arctic winds. Mentally assign a DTD. Once you feel adequately valid, curl your body into a complete tag. Look your partner in the eye and say, "One." For every ten repetitions, scoot your buttocks forward two inches to honor indentation and beauty. Once this activity is complete, return to your seat and await instruction.




**Do not rush through your situps. This boy did and he is stuck in a permanent self-closing position.**

# Domain Specific Languages in Java

- Most DSL's for Java are declarative
  - Property files
  - XML Based languages
- Property files
  - Key and value pairs are simplistic
- XML based languages
  - <your><angle><keys><wear><out>
- Why do we use them?
  - Writing a parser is a greater waste of time

# There is Another Way

- JRuby, of course
  - Ruby is a fantastic language for DSL's
  - No parser writing!
  - No XML situps!
  - DRY up your configuration

# Spring Configuration

```
<bean id="bean3" class="springy.beans.Bean3">
    <constructor-arg value="${vic}"/>
    <constructor-arg ref="bean2"/>
    <property name="aProperty">
        <list>
            <ref bean="bean1"/>
            <value>A String</value>
        </list>
    </property>
</bean>
```

# Springy Equivalent

```
bean :bean3, "springy.beans.Bean3" do |b|
  b.new(vic, :bean2)
  b.aProperty = [ :bean1, "A String" ]
end
```

# Spring Extensible XML

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.springframework.org/schema/myns"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:beans="http://www.springframework.org/schema/beans"
    targetNamespace="http://www.mycompany.com/schema/myns"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">

   <xsd:import namespace="http://www.springframework.org/schema/beans"/>

   <xsd:element name="dateformat">
      <xsd:complexType>
         <xsd:complexContent>
            <xsd:extension base="beans:identifiedType">
               <xsd:attribute name="lenient" type="xsd:boolean"/>
               <xsd:attribute name="pattern" type="xsd:string" use="required"/>
            </xsd:extension>
         </xsd:complexContent>
      </xsd:complexType>
   </xsd:element>

</xsd:schema>
```

# Springy Equivalent

```
def dateformat(id, format, lenient=false, args = {})
  register_bean(
   BeanDef.new(id, "java.text.SimpleDateFormat", args).
       new(format).lenient(lenient))
end
```

# Testing

- Use Ruby's testing frameworks to test your Java
  - RSpec
- Dynamically typed languages make it easier to write tests
- Develop testing DSL's
- Write executable documentation!

# Behavior Driven Development

- Is your documentation this useful (executable)?

```
describe User do
  it "should be logged in" do
    user = User.new
    user.login
    user.should be_logged_in
  end
end
```

# Rails

- The vast majority of rails runs perfectly
- ActiveRecord-JDBC project
- Integration through the Goldspike project
- Multiple deployment options
  - jruby script/server - Webrick
  - Command line Jetty
  - Mongrel with the mongrel-support project
  - WAR for deployment in app server

# Installing Rails Integration

```
jem install activerecord-jdbc

jem install rails

rails jruby_test

cd jruby_test

./script/plugin install svn://rubyforge.org/var/svn/jruby-
   extras/trunk/rails-integration/plugins/goldspike

jake war:standalone:create

jake war:standalone:run
```

# Gotchas

- Path issues: gem, rails, jake, mongrel, etc.
  - ln -s gem jem ...
- ./script/* will invoke MRI
  - jruby ./script/server
- JDBC driver loading
  - JAVA_OPTS="-Xbootclasspath/p:/path/to/driver.jar" jruby script/server

# Questions?

# Links

- JRuby
  - http://jruby.codehaus.org
- JRuby Extras
  - http://rubyforge.org/projects/jruby-extras
- Springy
  - http://code.trampolinesystems.com/springy
- RSpec
  - http://rspec.rubyforge.org/