# JRuby (for Rubyists)

Tyler Jennings
Principal Consultant

tyler@obtiva.com
@tyler_jennings

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


• Have JRuby installed
• Have Java 1.6
• Run some examples


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# Bootsrapping

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


//obtiva

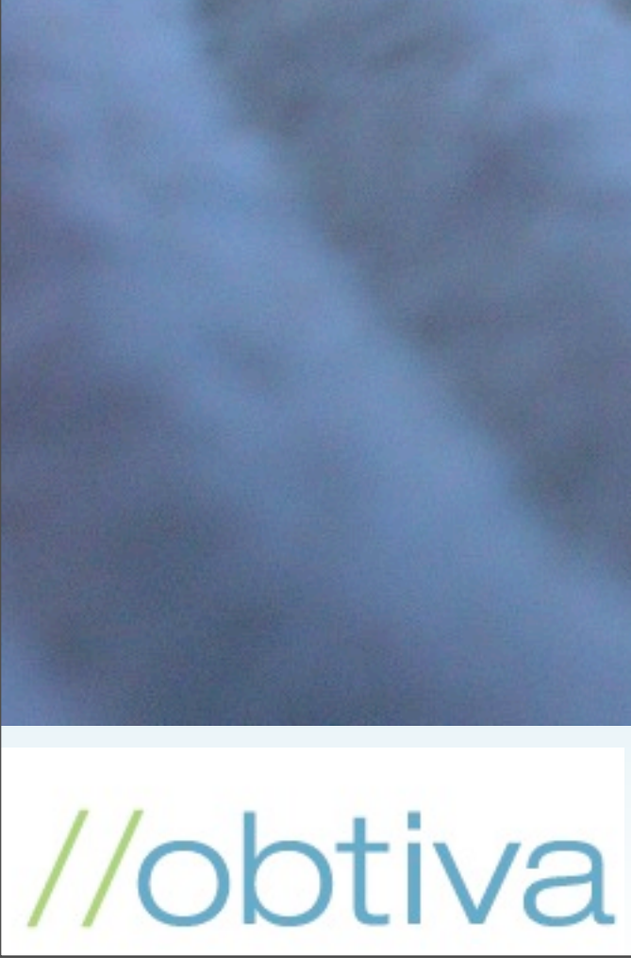
AGILITY APPLIED.
SOFTWARE DELIVERED.

```
> 1 + 1
=> 2
> ary = []
=> []
> ary << [1,2,3]
=> [[1, 2, 3]]
> ary.flatten
=> [1, 2, 3]
>
```

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# Using Java (Classes)




AGILITY APPLIED.
SOFTWARE DELIVERED.

# Importing

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

```
require 'java'

import javax.swing.JFrame

frame = JFrame.new("Test")
frame.default_close_operation =
  JFrame::EXIT_ON_CLOSE
frame.setSize(450,450)
frame.visible = true

```

# importing_classes.rb

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# By Package

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

```
 3 class TestUI
 4   include_package "javax.swing"
 5   include_package "java.awt"
 6
 7   def initialize
 8     frame = JFrame.new("Test")
 9     frame.default_close_operation =
10       JFrame::EXIT_ON_CLOSE
11     frame.setSize(450,450)
12
13     box = JTextField.new
14     box.maximum_size = Dimension.new(30,30)
15     frame.add box
16
17     frame.visible = true
18   end
19 end
20 TestUI.new
```

# importing_by_package.rb

# Subclassing

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

```ruby
require 'java'
class RubyMap < java.util.HashMap
end

p RubyMap.new.methods.grep(/key/).sort
```

# subclassing.rb

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# Interfaces

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

```ruby
#Implement an interface (optional)
include ActionListener

def initialize
  frame = JFrame.new("Calculator")
  frame.default_close_operation = JFrame::EXIT_ON_CLOSE
  frame.setSize(450,450)
  button = JButton.new("Press ME")

  button.add_action_listener self
  button.add_action_listener do |event|
    puts "BLOCK: " + event.source.label
  end

  frame.add button
  frame.visible = true
end

def actionPerformed(event)
  puts  "METHOD: " + event.source.label
end
```

# interfaces.rb

# Extensions

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

```
 3 JHashMap = java.util.HashMap
 4
 5 class JHashMap
 6   def + (other)
 7     map = JHashMap.new
 8     map.put_all self
 9     map.put_all other
10     map
11   end
12   def print
13     to_a.join(">")
14   end
15 end
16
17 one_two = JHashMap.new(:one => 1) +
18           JHashMap.new(:two => 2)
19
20 puts one_two.print
```

# extensions.rb

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# Methods

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

```
require 'java'
import javax.swing.JFrame

frame = JFrame.new("Test")
#Java version
frame.setSize(450,450)
#Jruby's
frame.set_size(450, 450)

#Java setter
frame.setVisible true
#JRuby's version
frame.visible = true
```

# methods.rb

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# Grid layout

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

```
 3 class ButtonLayoutUI
 4   include_package 'javax.swing'
 5   include_package 'java.awt'
 6
 7   def initialize
 8     frame = JFrame.new("Calculator")
 9     frame.default_close_operation = JFrame::EXIT_ON_CLOSE
10     frame.setSize(100,100)
11     digits = JPanel.new
12     digits.layout = GridLayout.new 3, 3
13
14     (1..9).each do |n|
15       digits.add JButton.new n.to_s
16     end
17     frame.add digits
18     frame.visible = true
19   end
20 end
21 ButtonLayoutUI.new
```

# layout.rb

# Exercise

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

- Build a Calculator UI
- Using calculator.rb

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

```
 9 require 'java'
10 require 'calculator'
11
12 class CalculatorUI
13   def initialize
14     @calculator = Calculator.new
15   end
16 end
17
18 CalculatorUI.new
```

# calculator_ui_exercise.rb

```ruby
require 'java'

# Usage:
#
# calculator = Calculator.new
#
# calculator.evaluate("1")
# calculator.evaluate("+")
# calculator.evaluate("1")
# calculator.evaluate("=")
# 2 == calculator.current_value
#
class Calculator
  OPERATORS = ["+", "-", "/", "*"]
  FUNCTIONS = ["=", "CE"]
  OPERATIONS = OPERATORS + FUNCTIONS
```

# calculator.rb

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# Deploying (Rails)

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# Warbler

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# warble -T

```
warble compiled   # Feature: precompile all Ruby files
warble config     # Generate a configuration file to customize your war
warble executable # Feature: make an executable archive
warble gemjar     # Feature: package gem repository inside a jar
warble pluginize  # Install Warbler tasks in your Rails application
warble version    # Display version of Warbler
warble war        # Create the project .war file
warble war:clean  # Remove the .war file
warble war:debug  # Dump diagnostic information
```

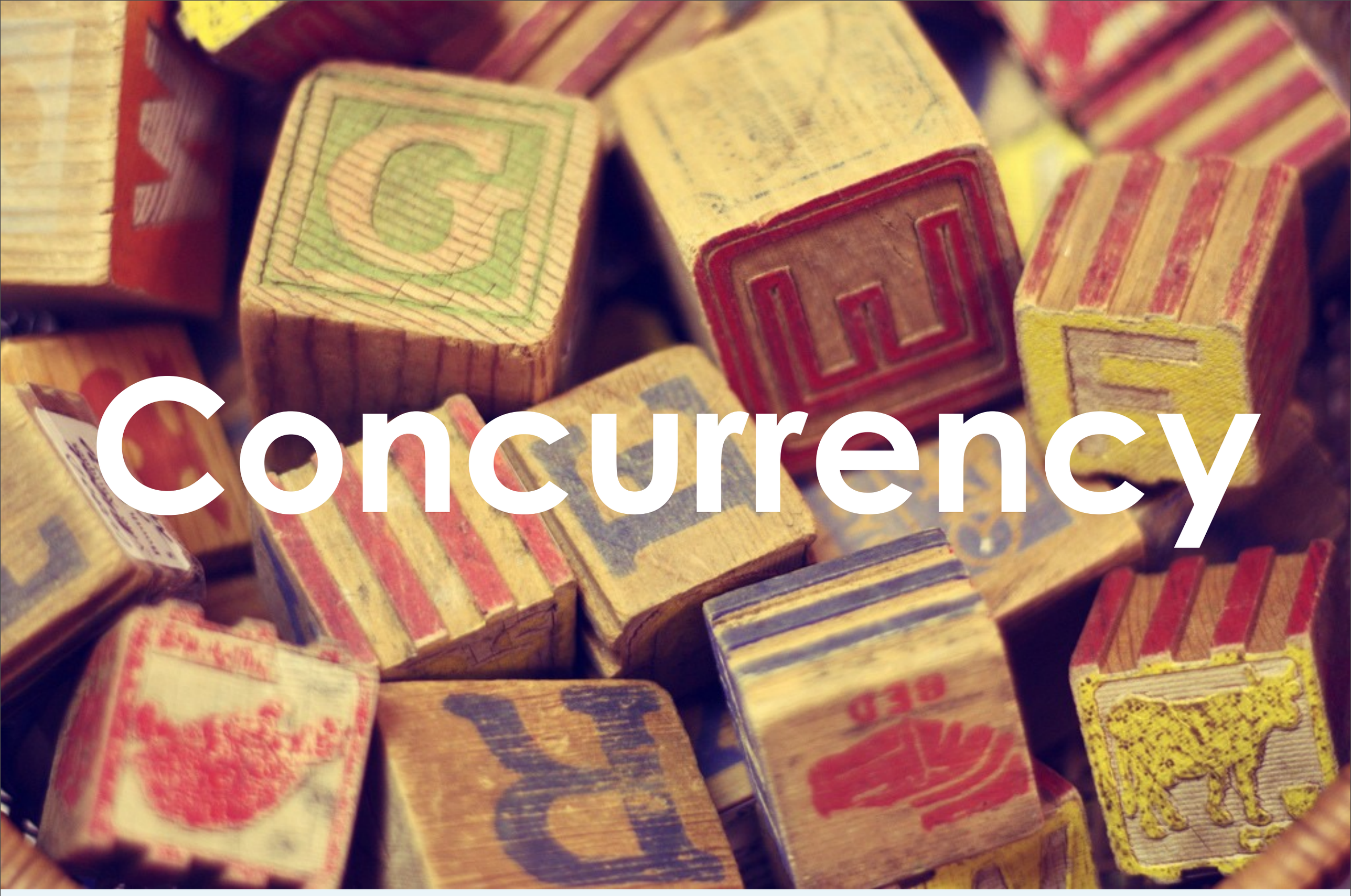
# JRuby Rack

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.



# Concurrency

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# In Rails

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# 3 Ways

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# 1: Immutable

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# 2: Mutex

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


# threads/mutex.rb

# 3: Threadlocal

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

```
1 a = Thread.new { Thread.current["name"] = "A"; Thread.stop }
2 b = Thread.new { Thread.current[:name]  = "B"; Thread.stop }
3 c = Thread.new { Thread.current["name"] = "C"; Thread.stop }
4
5 Thread.list.each {|x| puts "#{x.inspect}: #{x[:name]}" }
```

# threads/threadlocal.rb

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# Exercise

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

```ruby
require 'thread'

output, threads = [], []
string = "Don't muddle me up!!\n"

100.times do |t|
  threads << Thread.start do
    100.times do
      string.each_char do |c|
        output << c
      end
    end
  end
end

threads.each {|t| t.join}
puts output.join.split("\n").reject{|s| s == string.strip}

```

# threads/concurency_exercise.rb

# Tuning JRuby

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


--help

- Pass JVM options
- -J[java options]
- Ruby compatability
- --1.9, --1.8
- JVM mode
- --server, --client

- Eval
- -e “puts ‘hello’”


--properties

- Specify them:
- -J-D<prop>=<value>

- JIT compilation
- jruby.compile.mode
- JIT threshold
- jruby.jit.threshold
- JIT max methods
- jruby.jit.max

- Stand up classes
- jruby.reify.classes


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


• Specify them:
• -J<prop>=<value>
• -J-X<prop>=<value>


• Min heap
• -Xms<size> (min)
• Max heap
• -Xmx<size> (max)
• Max Permgen
• -XX:MaxPermSize

# Exercise

optimization/start_rails.sh

optimization/benchmark_rails.rb

//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

- Profile
- --profile
- --profile.flat
- --profile.graph

# Exercise

profiling/start_rails.sh

profiling/benchmark_rails.rb

//obtiva

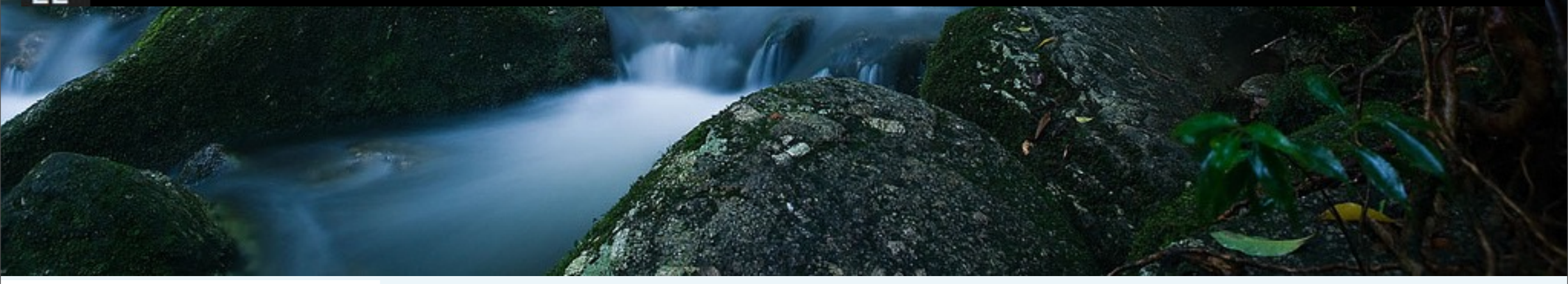
AGILITY APPLIED.
SOFTWARE DELIVERED.


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

```
1  task :sleep => :snooze do
2    sleep 1
3  end
4  task :snooze do
5    sleep 2
6  end
7  task :default => :sleep
8
9  require 'stickshift'
10 ::Rake::Application.instrument :top_level
11 ::Rake::Application.instrument :[], :with_args => 0
12 ::Rake::Task.instrument :invoke, :execute, :inspect_self => true
```

```
14     0ms >  Rake::Application#top_level < 3001ms
15     0ms >    Rake::Application#[]("default") < 0ms
16     0ms >    Rake::Task#invoke{<Rake::Task default => [sleep]>} < 3001ms
17     0ms >      Rake::Application#[]("sleep") < 0ms
18     0ms >      Rake::Application#[]("snooze") < 0ms
19  2000ms >      Rake::Task#execute{<Rake::Task snooze => []>} < 2000ms
20  1000ms >      Rake::Task#execute{<Rake::Task sleep => [snooze]>} < 1000ms
21     0ms >      Rake::Task#execute{<Rake::Task default => [sleep]>} < 0ms
22
```



//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.

# JRuby startup is a little slow ...

```
time ruby -e "puts 'hello'"
```

```
real    0m1.424s
user    0m0.788s
sys     0m0.160s
```

# Start an ng server

```
$ ruby --ng-server &

$ NGServer started on all interfaces, port 2113.
```

# Much better with the --ng switch!

```
time ruby --ng -e "puts 'hello'"
```

```
real    0m0.292s
user    0m0.011s
sys     0m0.027s
```


//obtiva

AGILITY APPLIED.
SOFTWARE DELIVERED.