Combining Dynamically and Statically Typed Languages for Fun and Profit

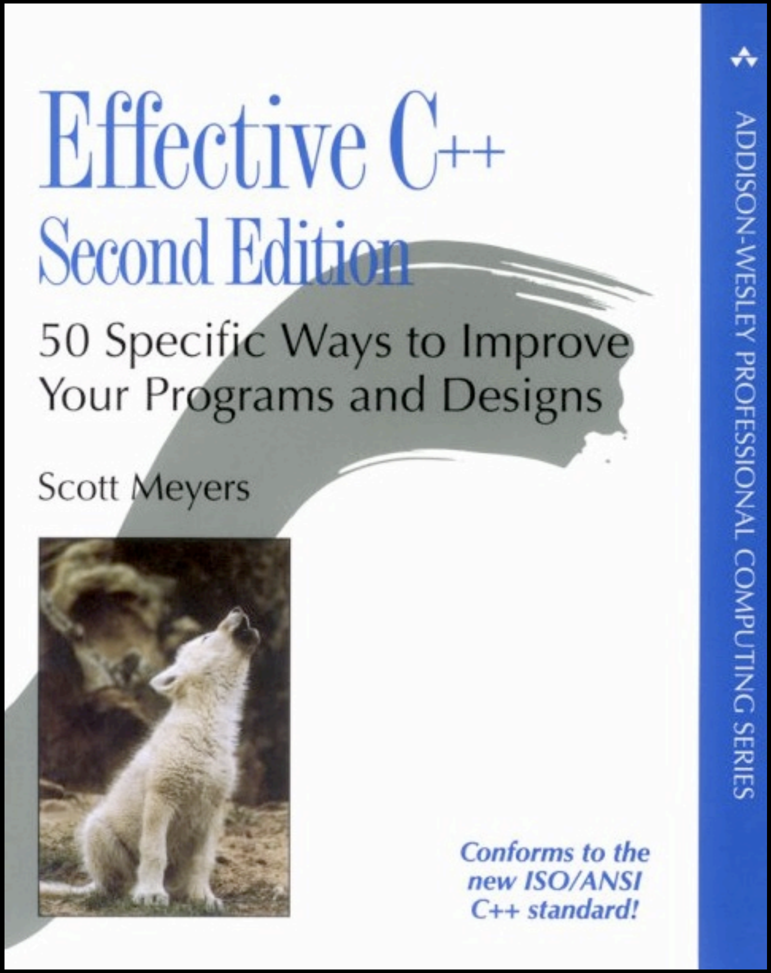
Gavin Stark
Lunch 2.0
June 3, 2009

@jruby

Me

- Twitter: @GavinStark
- Blog: http://hasmanyquestions.wordpress.com
- Co-organizer of Tampa Ruby Brigade
- VP Product Development, Real Digital Media

Me^2


Effective C++
Second Edition
50 Specific Ways to Improve Your Programs and Designs
Scott Meyers
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES
Conforms to the new ISO/ANSI C++ standard!

Daily Code

C++ Java

Ruby

bash shell JavaScript

Why Java?

- Several Recovering C++ Developers
- 3rd Party Support
- Cross Platform

Why Ruby?

Started for Rails

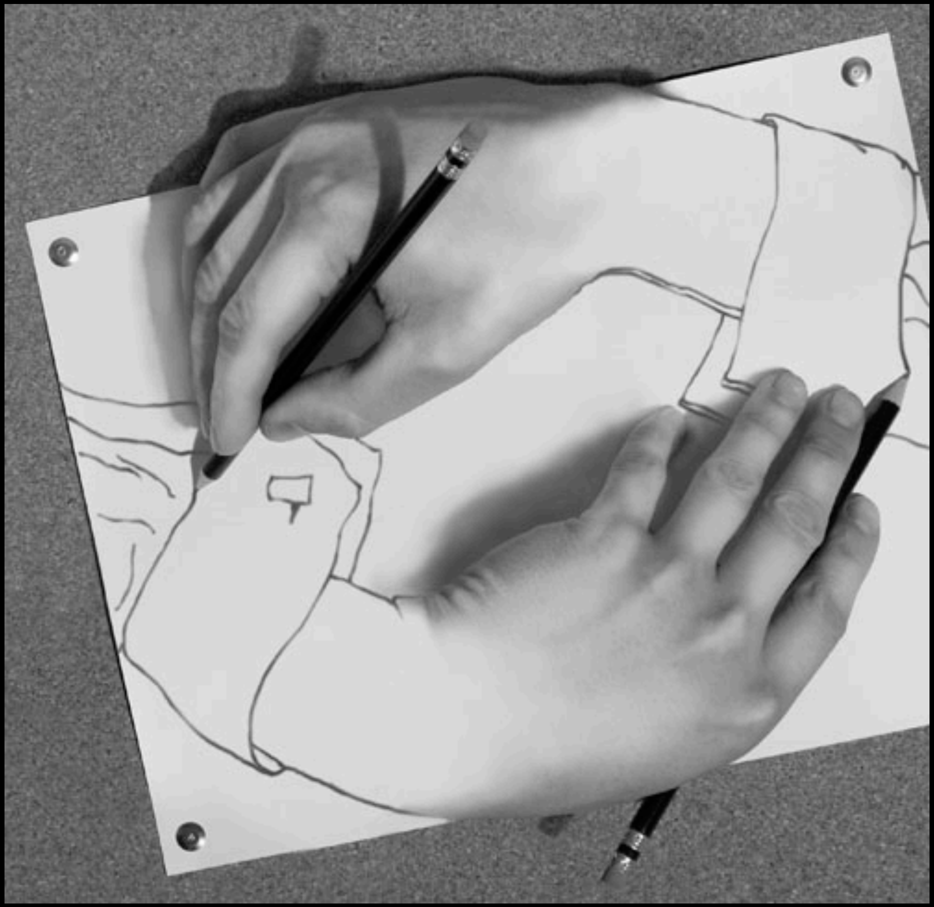
(Rails logo from http://railslogo.com)

Why Ruby?

- Metaprogramming
- DSLs
- Testing Frameworks!

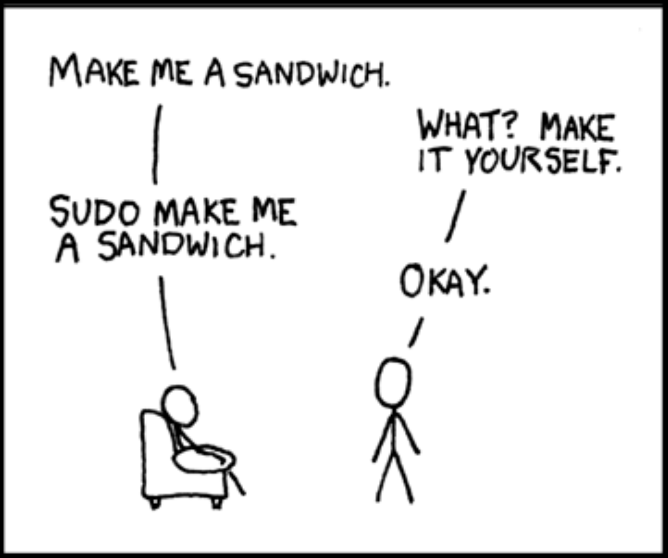
(what? did a programmer just get ***excited*** about testing!?!?)

Metaprogramming



http://www.flickr.com/photos/41992302@N00/136743967/

Domain Specific Language



```ruby
class Animal

end

class Cat < Animal
  has 4.legs
  is furry
  hates water
end

class Fish < Animal
  has fins
  swims
  loves water
end

```

DSLs

Rails leverages this to great effect

```
class Employee < ActiveRecord::Base
  validates_uniqueness_of :name
end

```

How?

- Everything is an object
- Classes and objects are open
- Everything is reflective
- Declarations (most) are executable statements (this matters -- alot)

```ruby
class Employee
  attr_accessor :name
  attr_accessor :title
  attr_reader   :id
end

gavin       = Employee.new
gavin.name  = "Gavin Stark"
gavin.title = "VP Product Development"

puts gavin.id
```

attr_accessor / attr_reader are methods

- Defined at the class level
- Evaluated as the class definition is evaluated
- How might we implement these methods?

Metaprogramming accessors

```ruby
class Module
  def attr_reader(*syms)
    syms.each do |sym|
      class_eval %{def #{sym}; return @#{sym}; end}
    end
  end
end
```

From Glenn Vanderberg: http://www.vanderburg.org/Speaking/Stuff/oscon05.pdf

Metaprogramming accessors

```ruby
class Module
  def attr_writer(*syms)
    syms.each do |sym|
      class_eval %{def #{sym}=(val); @#{sym} = val; end}
    end
  end
end

```

From Glenn Vanderberg: http://www.vanderburg.org/Speaking/Stuff/oscon05.pdf

Blocks and Procs

- Dynamic language feature heavily employed in DSL/Metaprogramming
- Objects which represent code
- Can be called inside other code
- Heavily used in iteration/enumeration

Example, iteration

```ruby
class Employee

  def initialize
    @vacation_days = [ "July 4th, 2009",
                       "December 25th, 2009" ]
  end

  def each_vacation_day
    @vacation_days.each do |vacation_day|
      yield vacation_day
    end
  end
end

gavin = Employee.new
gavin.each_vacation_day do |day|
  puts "Taking #{day} off"
end
```

Call our block

Block

Metaprogramming secret weapon

method_missing

http://www.flickr.com/photos/fornal/368421738/

Example:

```ruby
class Employee
end

gavin = Employee.new
gavin.give_raise( 10_000 )
```

Implementing method_missing

```ruby
class Employee
  def method_missing( method, *args, &block)
   puts "Called method #{method} with arguments: #{args.inspect}"
  end
end

gavin = Employee.new
gavin.give_raise( 10_000 )
```

XML Builder library

```ruby
require 'rubygems'
require 'builder'

xml = Builder::XmlMarkup.new

xml.employees do |employees|
  employees.employee do |employee|
    employee.name  = "Gavin Stark"
    employee.title = "VP Product Development"
  end
end
```

Favorite Use of Metaprogramming

- Builders
- Rails
- Testing Frameworks (RSpec, Cucumber)

JRuby

- Access Java code from Ruby
- Access Ruby code from Java
- Run Ruby code on the very fast JVM
- Access java libraries (though Ruby library growth has been increasing since Rails)

Getting stared

```ruby
require 'java'

s = java.lang.String.new( "Testing 123")

if s.endsWith( "123" )
  puts "Yup, ends with 123"
end

if s.ends_with( "123" )
  puts "Also ends with 123"
end
```

What about our jars and classes?

- RDM's PropertyList
- Nested, order preserving
  key (multiple) => value data structure.
- Iteration, marshaling to/from string
- Wanted to use this from Ruby

Accessing the classes

```ruby
require 'java'

NEOCAST_LIB_HOME = "~/dev/git/RDM/XFPlayerServer/project/dist"
require "#{NEOCAST_LIB_HOME}/NEOCASTPlayer.jar"

property_list = com.rdm.util.propertylist.PropertyList.createFrom( "Sample:\n---" )
```

that. is. just. too. verbose.

```ruby
require 'java'

module RDM
  NEOCAST_LIB_HOME = "~/dev/git/RDM/XFPlayerServer/project/dist"
  require "#{NEOCAST_LIB_HOME}/NEOCASTPlayer.jar"
  include_class "com.rdm.util.propertylist.PropertyList"
end

property_list = RDM::PropertyList.createFrom( "Sample:\n---" )
```

One small step

```ruby
require 'java'

module RDM
  NEOCAST_LIB_HOME = "~/dev/git/RDM/XFPlayerServer/project/dist"
  require "#{NEOCAST_LIB_HOME}/NEOCASTPlayer.jar"
  include_class "com.rdm.util.propertylist.PropertyList"
end

property_list_string = %{
Sample:
\tfirst=Lorem
\tsecond=Ipsum
---
}

property_list = RDM::PropertyList.createFrom( property_list_string )

it = property_list.getPropertyIterator()
while it.hasNext
  value = it.next
  puts "Value is #{value}"
end
```

Not very Ruby like

- Open classes and blocks FTW

```ruby
require 'java'

module RDM
  NEOCAST_LIB_HOME = "~/dev/git/RDM/XFPlayerServer/project/dist"
  require "#{NEOCAST_LIB_HOME}/NEOCASTPlayer.jar"
  include_class "com.rdm.util.propertylist.PropertyList"
end

property_list_string = %{Sample:\n\tfirst=Lorem\n\tsecond=Ipsum\n---\n}

module PropertyListEnumerable
  def properties
    it = getPropertyIterator
    while( it.hasNext )
      yield it.next
    end
  end
end

class RDM::PropertyList
  include PropertyListEnumerable
end

property_list = RDM::PropertyList.createFrom( property_list_string )

property_list.properties do |property|
  puts "Value is #{property.value}"
end
```

Implications

- Every PropertyList returned from Java space has the new "properties" method
- I can replace Java methods with my own (if I wanted to)

How we' use JRuby at RDM

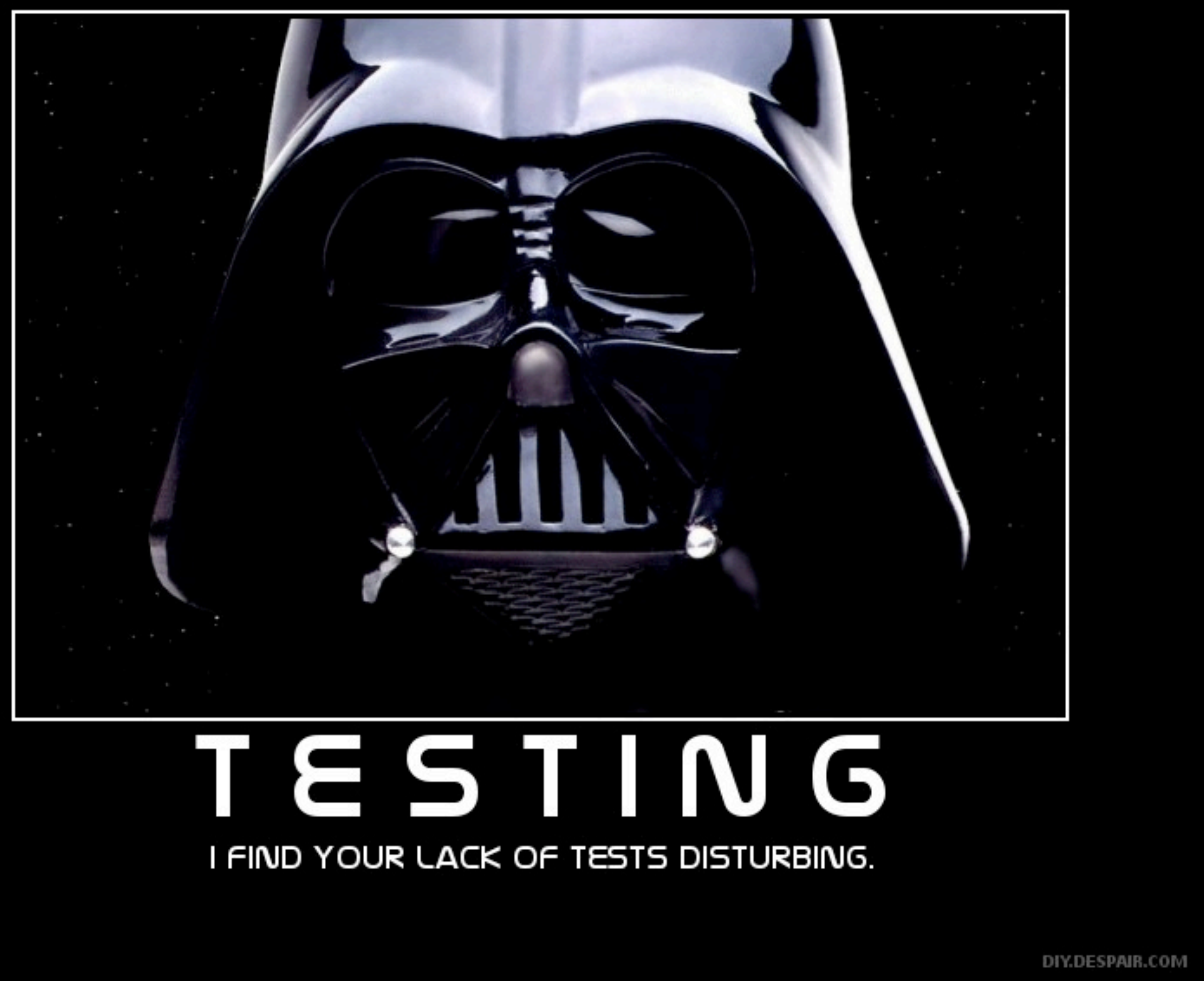
- Simulation
- Reporting
- Testing
- Reusing Rails business logic in Java

Simulation

- Use JRuby to write a simulator
- Reuse core Java code from Ruby
- Scales well
    - JRuby replaces Ruby's 'green threads' with native Java threads

Reporting

jasperreports + = $



http://www.flickr.com/photos/sebastian_bergmann/2282734669/

Cucumber

- Behavior Driven Development tool
- In Rails - typically used for web app integration via webrat.
- Slogans:
  - BDD that talks to domain experts first and code second
  - BDD with elegance and joy
  - Use your mother tongue

Cucumber

```
Feature: Division
  In order to avoid silly mistakes
  Cashiers must be able to calculate a fraction

  Scenario: Regular numbers
    Given I have entered 3 into the calculator
    And I have entered 2 into the calculator
    When I press divide
    Then the result should be 1.5 on the screen
```

http://cukes.info/

Cucumber - Demo

I'm not an expert, but I play one in demos.

Cucumber - remote services

```
Feature: Remote Time Service
  In order to keep accurate time
  Computers must have a good time source

  Scenario: Running ahead of time
    Given I am running 2 hours ahead of time
    When I request to have my time adjusted
    Then my time is accurate within 1 second
```

Good place to use Mocking an Stubbing

Cucumber - RDM

```
Feature: Synchronize Content
  In order to keep the player’s content up-to-date
  The player must run SynchronizeContent

  Scenario: Starting from empty
    Given I am a player with a campaign with three ads
    Given I am a player without any content
    When I start the task SynchronizeContent
    Then I am asked for my current content
    Then I am given 3 download steps
```

Good place to use a simulator (or mocking/stubbing)

Java + Rails

- Rails for UI
- Java for backend
- Significant part of Java code is database access
- Business logic in Rails only going to grow

JRuby to the rescue

- Access Ruby code from Java
- Currently:
  - Define Java interface
  - include interface in Ruby side
  - Do some tricks to gain access to Ruby class in Java

Questions?