JRuby
Apples AND Oranges

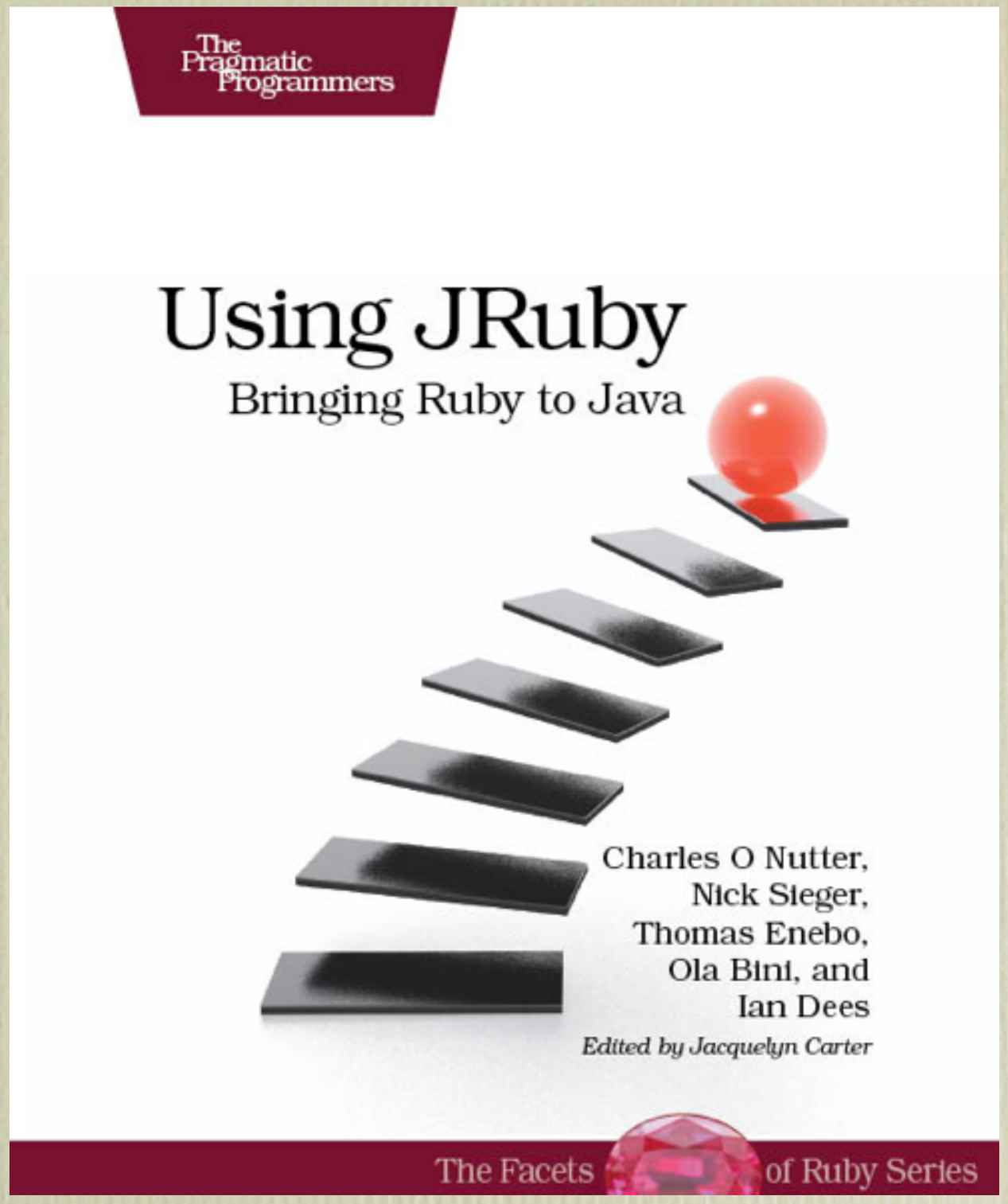
Thomas E. Enebo
Engine Yard Inc., www.jruby.org

GOALS

- Get an idea of how JRuby can help you in day-to-day development

Who Am I?

- JRuby Guy
  - 4+ years as “day job” + few years as hobby
- Almost 10 years of Ruby
- 15+ years of Java
- Friendly
- Likes Beer



http://www.pragprog.com/titles/jruby/using-jruby

JRuby @ 10_000

- Implementation of Ruby runtime
- Runs on JVM (Java 5+)
- Open Source (CPL, GPL, LGPL)
- Integrates well with Java

Ruby Quick Tour

- The relative feel between Java and Ruby

Classes

```java
public class Circle extends Shape {
  private final int radius;

  public Circle(int radius) {
    this.radius = radius;
  }
  public int getRadius() {
    return radius;
  }
  public double getArea() {
    return Math.PI * Math.pow(radius,2);
  }
  public static void main(String[] a) {
    double area=new Circle(2).getArea();
    System.out.println(area);
  }
}
```

```ruby
class Circle < Shape
  def initialize(radius)
    @radius = radius
  end

  attr_reader :radius

  def area
    Math::PI*(@radius ** 2)
  end
end

puts Circle.new(2).area
```

Single Inheritance

Same Thing

```java
public class Circle extends Shape {
  private final int radius;

  public Circle(int radius) {
    this.radius = radius;
  }
  public int getRadius() {
    return radius;
  }
  public double getArea() {
    return Math.PI * Math.pow(radius,2);
  }
  public static void main(String[] a) {
    double area=new Circle(2).getArea();
    System.out.println(area);
  }
}
```

```ruby
class Circle < Shape
  def initialize(radius)
    @radius = radius
  end

  attr_reader :radius

  def area
    Math::PI*(@radius ** 2)
  end
end

puts Circle.new(2).area
```

Constructor

```java
public class Circle extends Shape {
  private final int radius;

  public Circle(int radius) {
    this.radius = radius;
  }
  public int getRadius() {
    return radius;
  }
  public double getArea() {
    return Math.PI * Math.pow(radius,2);
  }
  public static void main(String[] a) {
    double area=new Circle(2).getArea();
    System.out.println(area);
  }
}
```

```ruby
class Circle < Shape
  def initialize(radius)
    @radius = radius
  end

  attr_reader :radius

  def area
    Math::PI*(@radius ** 2)
  end
end

puts Circle.new(2).area
```

No Type Declarations

```java
public class Circle extends Shape {
  private final int radius;

  public Circle(int radius) {
    this.radius = radius;
  }
  public int getRadius() {
    return radius;
  }
  public double getArea() {
    return Math.PI * Math.pow(radius,2);
  }
  public static void main(String[] a) {
    double area=new Circle(2).getArea();
    System.out.println(area);
  }
}
```

```ruby
class Circle < Shape
  def initialize(radius)
    @radius = radius
  end

  attr_reader :radius

  def area
    Math::PI*(@radius ** 2)
  end
end

puts Circle.new(2).area
```

Dynamic Typing

- Ruby is Dynamically Typed
  - Why we see no Type decls on variables
  - Runtime type-checking (like Java casts)
  - “If it waddles like a duck and quacks like a duck…”

Instance Variables

'@' == 'this.' and is mandatory

```java
public class Circle extends Shape {
  private final int radius;

  public Circle(int radius) {
    this.radius = radius;
  }
  public int getRadius() {
    return radius;
  }
  public double getArea() {
    return Math.PI * Math.pow(radius,2);
  }
  public static void main(String[] a) {
    double area=new Circle(2).getArea();
    System.out.println(area);
  }
}
```

```ruby
class Circle < Shape
  def initialize(radius)
    @radius = radius
  end

  attr_reader :radius

  def area
    Math::PI*(@radius ** 2)
  end
end

puts Circle.new(2).area
```

Point of No Return

Last Expression is always return value

```java
public class Circle extends Shape {
  private final int radius;

  public Circle(int radius) {
    this.radius = radius;
  }
  public int getRadius() {
    return radius;
  }
  public double getArea() {
    return Math.PI * Math.pow(radius,2);
  }
  public static void main(String[] a) {
    double area=new Circle(2).getArea();
    System.out.println(area);
  }
}
```

```ruby
class Circle < Shape
  def initialize(radius)
    @radius = radius
  end

  attr_reader :radius

  def area
    Math::PI*(@radius ** 2)
  end
end

puts Circle.new(2).area
```

.new is just a class method!

```java
public class Circle extends Shape {
  private final int radius;

  public Circle(int radius) {
    this.radius = radius;
  }
  public int getRadius() {
    return radius;
  }
  public double getArea() {
    return Math.PI * Math.pow(radius,2);
  }
  public static void main(String[] a) {
    double area=new Circle(2).getArea();
    System.out.println(area);
  }
}
```

```ruby
class Circle < Shape
  def initialize(radius)
    @radius = radius
  end

  attr_reader :radius

  def area
    Math::PI*(@radius ** 2)
  end
end

puts Circle.new(2).area
```

Blocks/Closures

- Pass an anonymous function to a method call

```
# Look mom...no boilerplate!
my_open(file) do |io|
  first_line = io.gets
  # ...
end

def my_open(file, mode='r')
  io = File.open(file)
  yield io
ensure
  io.close
end
```

```
letters.group_by {|b| b.zip_code }
```

```
button.action_performed do
  exit
end
```

Modules

```ruby
module Enumerable
  def find(if_none = nil)
    each { |o| return o if yield(o) }
    if_none
  end

  # Many other methods not shown

  def collect
    ary = []
    each { |o| ary << yield(o) }
    ary
  end
end
```

```ruby
class MyTree
  include Enumerable

  # tree impl not shown :)

  def each
    # yield each element
    # in tree
  end
end
```

```ruby
dude = people.find { |person| person.id == 123 }

floats = [1,2].collect { |int| int.to_f } # => [1.0, 2.0]
```

Everything is an Expression

```
class Color
  COLORS = {:red => 0xff0000, :green => 0x00ff00, :blue => 0x0000ff}

  COLORS.each do |name, value|
     define_method(name) do
      value
     end
   end
end
```

Classes/Modules are open

```ruby
class MyTree
  def each
    #...
  end
end

#... later in source
#... maybe even different file

class MyTree
  def debug
    #...
  end
end
```

JRuby Stuff

Installation

- Windows

- Others
  - Unzip or Untar installation
  - Add ‘bin’ directory to your PATH

Why not just a Jar?

- ‘jruby’ command-line launcher
  - Works just like C Ruby launcher
- Many useful tools out of the box
  - rspec, rake, rubygems, irb

Integration with Java

- Call into Ruby from Java
  - Java 6 Scripting
  - Internal “Red Bridge” API
- Access Java classes from Ruby
  - Call, Decorate, and Define in Ruby

Access Ruby from Java

- Red Bridge Shown...

```
ScriptingContainer container = new ScriptingContainer();

// ...

container.put("$revisions", new Revisions(args[0], args[1]));

List<Diff> f = (List<Diff>) container.runScriptlet("history");
for (GitDiff file: f) {
    System.out.println("FILE: " + file.getPath());
    System.out.println(file.getPatch());
}
```

Use Case: Scripting Java

- One-off scripts are easy as opening an editor
- Not everything must be an IDE project
- Easy to play with technology

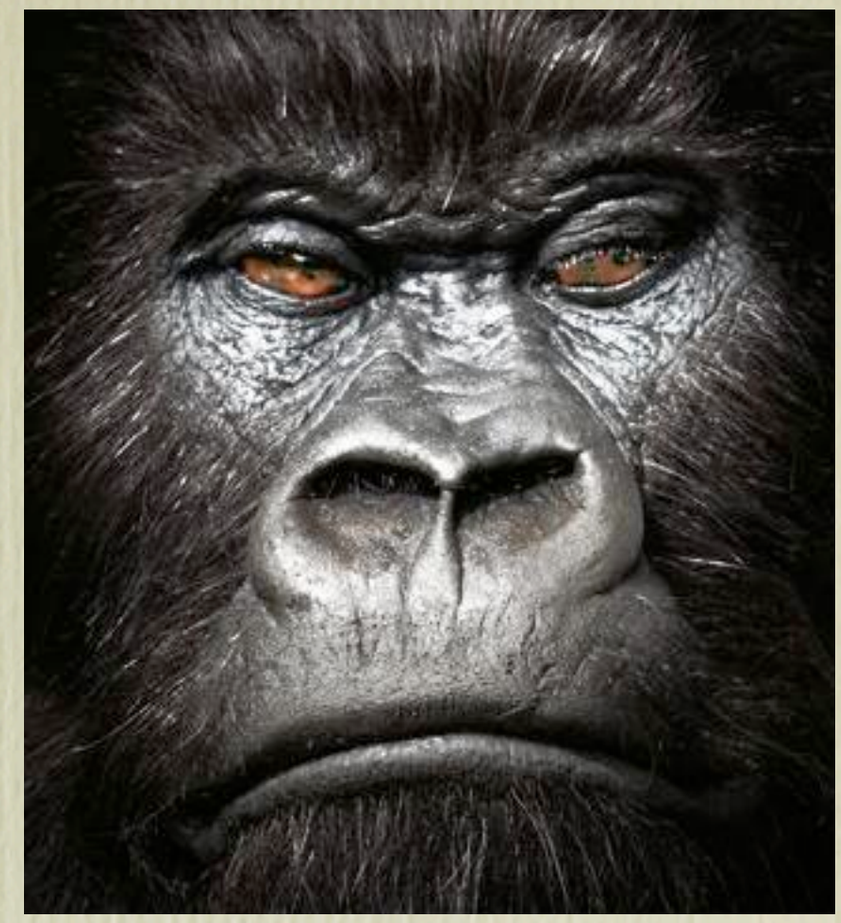
Demo: Scripting Java

Demo: Eye Candy

Use-case: Build tools

- Leverage Ruby Tooling to make building project more manageable

Ant + Rake



- Ant is 800 pound Gorilla of Java
- Declarative and a little Dumb but Dependable
- Used Everywhere

Ant + Rake

- Rake is Orangutang of Ruby
- Like Make or Ant
- ....but with Ruby for Imperative Goodness

Ant + Rake

- Call Rake From Ant
  - Call tasks
  - Rake tasks as Ant target dependencies
- Call Ant From Rake
  - Ditto!
- Mix and Match

Rake Calling Ant Tasks

```
require 'rake'
require 'ant'

task :init do
  ant.mkdir :dir => 'build'
end

task :compile => :init do
  ant.javac :destdir => 'build' do
    src { pathelement :location => 'src' }
  end
end

task :test => :some_ant_task
```

Rake Calling Ant

```
task :call_ant do
  ant '-f my_build.xml its_in_ant'
end

task :ant_import do
  ant_import # you could also do this outside
end

task :compile => [:ant_import, :its_in_ant] do
  # Do some compilation
end
```

Rake from Ant

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="foobar" default="default" basedir=".">
  <description>Builds, tests, and runs the project foobar.
      </description>

  <target name="load-rake-task">
    <taskdef name="rake" classname="org.jruby.ant.Rake"/>
  </target>

  <target name="default" depends="load-rake-task">
    <rake task="jar"/>
  </target>

  ...
</project>
```

Ant Calling Rake Tasks

```
<?xml version="1.0" encoding="UTF-8"?>

<project name="foobar" default="top-level" basedir=".">
    <description>Builds the project foobar.</description>

    <taskdef name="rakeimport"
             classname="org.jruby.ant.RakeImport"/>
    <rakeimport/>

    <target name="top-level" depends="its_in_rake" />

    <target name="its_in_ant">
      <echo message="ant: its_in_ant"/>
    </target>
</project>
```

Maven?

- Maven support is being worked on
- Are you a maven user?
- Any features you want?

Use Case: Testing

“#JRuby's Java integration is so awesome I will never write another line of java unit test code... it's all jruby (if I can help it)”

-Jay ‘obviously a JRuby user’ McGaffigan

JTestr

- Bridge to make it easy to use Ruby testing tech.
  - test/unit, rspec, mocha, ...
- Integrates with both Ant and Maven
- Allows mocking of non-final Java class/method
- Written by Ola Bini (previous speaker)
  - Ask him questions later :)

JTestr Example

```
my_project/
    build.xml
    test/
        test_hashmap.rb
        hashmap_spec.rb
    lib/
        jtestr.jar
    src/
        <your source ;) >
```

JTestr in Ant

```
<?xml version="1.0" encoding="UTF-8"?>

<project basedir="." default="jar" name="JRuby">
  <target name="test" description="Runs all tests">
    <taskdef name="jtestr"
             classname="org.jtestr.ant.JtestRAntRunner"
             classpath="lib/jtestr.jar"/>
    <jtestr/>
  </target>
</project>
```

Ruby's test/unit

test/test_hashmap.rb

```
class HashMapTests < Test::Unit::TestCase
  def setup
    @map = java.util.HashMap.new
  end

  def test_that_an_entry_can_be_added
    @map.put "foo", "bar"
    assert_equal "bar", @map.get("foo")
  end

  def test_empty_key_set_iterator_throws_exception
    assert_raises(java.util.NoSuchElementException) do
      @map.key_set.iterator.next
    end
  end
end
```

Ruby's rspec

test/hashmap_spec.rb

```
describe "An empty", HashMap do
  before :each do
    @hash_map = java.util.HashMap.new
  end

  it "should be able to add an entry to it" do
    @hash_map.put "foo", "bar"
    @hash_map.get("foo").should == "bar"
  end

  it "should not be empty after an entry has been added to it" do
    @hash_map.put "foo", "bar"
    @hash_map.should_not be_empty
  end

  it "should be empty" do
    @hash_map.should be_empty
  end
end
```

OUTPUT!

% ant test

```
Buildfile: /Users/enebo/work/jtestr/build.xml

test:
    [jtestr] Other TestUnit: 2 tests, 0 failures, 0 errors
    [jtestr]
    [jtestr] Other Spec: 3 examples, 0 failures, 0 errors
    [jtestr]
    [jtestr] Total: 5 tests, 0 failures, 0 errors, 0 pending
    [jtestr]

BUILD SUCCESSFUL
Total time: 9 seconds
```

Use Case: Web Development

- Ruby on Rails
  - Full-stack web framework
  - Model View Controller (MVC)
  - Convention over Configuration
  - Don’t Repeat Yourself (DRY)
  - Agile Programming

Rails Quick Tour

Getting Started is easy

% rails new myapp -m http://jruby.org/rails3.rb

```
myapp/
    Gemfile
    app/
        controllers/
        helpers/
        mailers/
        models/
        views/
    db/
    log/
    test/
    config/
    public/
    Rakefile
    lib/
```

- All apps the have same layout
- Totally self-contained
- Type ‘rails server’ and you are in development mode
    - Code and see changes live*
- Simple generation scripts

Migrations

- DB-agnostic DSL for managing Schemas
  - Each migration a committable file in ‘db’ dir

```ruby
class CreatePeople < ActiveRecord::Migration
  def self.up
    create_table :people do |t|
      t.string :last_name
      t.string :first_name
      t.integer :age
      t.timestamps
    end
  end

  def self.down
    drop_table :people
  end
end
```

ActiveModel

- Model single-sources from DB metadata
  - Pluralizes model name and looks for table
  - Grabs meta data and at runtime adds
    - fields + methods

```ruby
class Person < ActiveRecord::Base
  # Models really start out this small
end
```

ActiveModel

- Validations

```ruby
class Person < ActiveRecord::Base

  validates_presence_of :last_name
  validates_length_of :age, :in => 0..120
end
```

- Serialization

```ruby
person.to_json
person.to_xml
```

- Much More...

ActiveRelation

- Composable Relational Algebra DSL
- Lazy

```
Person.where(:last_name => "enebo").where(:first_name => "thomas")


# All Enebo's
all_enebos = Person.where(:last_name => "enebo")

# Yarrr...there can be only one
me = all_enebos.where(:first_name => "thomas")
```

Scopes in ActiveRecord

- Save useful relations as a name in your model!

```ruby
class Person < ActiveRecord::Base
  scope :enebos, where(:last_name => "enebo")
end

# In your controller
enebos = Person.enebos.order(:first_name)
```

Routes

```
MyApp::Application.routes.draw do
  resources :people, :except => [:index]
end
```

Verb	Path	Action	Used for
~~GET~~	~~/comments~~	~~index~~	~~display a list of all comments~~
GET	/comments/new	new	return a form for creating a comment
POST	/comments	create	create a new comment
GET	/comments/:id	show	display a comment
GET	/comments/:id/edit	edit	return a form for editing a comment
PUT	/comments/:id	update	update a comment
DELETE	/comments/:id	destroy	delete a comment

Controllers

```ruby
class PeopleController < ApplicationController
  respond_to :html
  respond_to :json, :only => :show

  def show
    @person = Person.find_by_id params[:id]
    respond_with @person
  end

  def create
    @person = Person.new(params[:person])
    @person.save
    respond_with @person
  end
end
```

Views

- ERB is the JSP of Rails
  - Rails generates lots of helpers like *_path to not have brittle URLs
  - Partials for fragment rendering

```
<b>Name:</b><%= @person.name %>

<% if @person.age > 21 %>
  <b>Age:</b><%= @person.age %>
<% end %>

<%= link_to 'Edit', edit_person_path(@person) %>
<%= link_to 'Back', people_path %>
```

JRuby on Rails Deployment

- Deploy to any Java App Server
  - warbler gem to generate .war files
- Use Trinidad or Glassfish Gem for Ruby style deployment
  - Start up server in directory...done.

Rails and Java

- Seamlessly use Java business objects in Rails project
- People hybridize Java web apps by adding Rails to the project for new pieces
  - Have JSPs <-> ERBs mix together
    - taglibs to render ERBs in JSPs

Conclusions

- Ruby is easy to learn for Java programmers
- Easy to mix Ruby+Java
- Useful Productivity Frameworks and Libraries
  - Rails, Rake, JTestr, ...
- One more tool for Java programmer's toolbelt

Thank You

- twitter: tom_enebo
- website: http://www.jruby.org/
- book: www.pragprog.com/titles/jruby/using-jruby
- JtestR: http://jtestr.codehaus.org/