Workin' on the Rails Road

Obie Fernandez

Original presentation delivered to the

Object Technology User Group

St. Paul / Minneapolis

September 20th 2005

Agenda

- Introduction to Ruby Programming
- Ruby on Rails Fundamentals and Interactive Demonstration

  Break
- Working in Ruby and Rails
- Is Rails Ready for Prime Time?
- Questions and Answers

Questions and Slides

- Ask questions at any time
- Slides and samples will be available online at obiefernandez.com
- I'll show that address and others again at the end

Introduction to Ruby Programming

“Smalltalk was one of the main influences for Ruby. Whenever you find a main difference between Perl and Ruby there's a good chance to find this feature in Smalltalk.”

RubyGarden

What is Ruby?

- Object-Oriented scripting language
- Conceptual similarities to Smalltalk
- Many system and text manipulation features like Perl
- Emphasis on simplicity and design elegance

Ruby is a Dynamic Language

- Completely Object-Oriented
  - All data is an object, no exceptions
  - Operators are methods
- Dynamic Behavior
  - Possible to add new classes, add or redefine methods at runtime
  - An instance of one class can behave differently than another instance of the same class at runtime

Powerful Features

- Single Inheritance, but…
  - Modules provide namespaces and allow ‘mixin’ capability
  - A module is a collection of methods and constants
- Ruby has blocks
  - Code surrounded by do…end or { … }
  - They are true closures; get variable bindings
  - Are passed to methods quite often

Ruby Variables

- Don’t need declaration
- Variable scope determined by naming convention
  - foo … local variable
  - @foo … instance variable
  - $foo … global variable
  - Foo or FOO … constant

Variable Assignment

- An assignment sets the *rvalue* to the *lvalue*
- Then it returns the that value as the result of the assignment expression
- Assigning to a variable or constant is hardwired into Ruby
- Assigning to an object attribute is a method call
- Parallel assignment

Strings

- String
  - Instances usually created with literal expressions
  - Sequences of 8-bit bytes
  - Printable characters or not, doesn't matter
  - Over 75 standard methods
- Symbol
  - Denoted by colon, like :foobar
  - Basically an interned string, always the same instance no matter where used in codebase

Regular Expressions

Ruby supports Perl-style regular expressions…

```
# Extract the parts of a phone number
phone = "123-456-7890"

if phone =~ /(\d{3})-(\d{3})-(\d{4})/
  ext  = $1
  city = $2
  num  = $3
End
```

Arrays and Hashes

- Array
  - Ordered collection of references
  - Literal is list of objects between square brackets
- Hash
  - Used extensively in practice
  - Any type of object can be used as index
  - Elements are not ordered
  - Literal form is key/value pairs enclosed in {…} using => to map keys to values

Numerics

- Integer
  - Small integers are Fixnum, up to size of a native machine word minus 1 bit
  - Big integers are Bignum, up to size of available memory
  - Conversion happens automatically
- Float
  - Real numbers
  - Uses native double-precision floating-point representation

Ruby Methods

- Only way to change an object’s state
- Use def keyword and start with lowercase letter
- Use trailing ? for queries and ! for *dangerous* methods
- Use trailing = for methods that take assignment

Writing Methods

- Default values for parameters supported
- Variable-length argument lists by placing an asterisk on the last parameter
- Methods accept blocks implicitly or via the last parameter being prefixed with an ampersand
- Every called method returns the value of the last expression. Explicit use of return statement is optional

Calling Methods

- Specify receiver, name of method
- Parameters and block optional
- Parentheses optional
- Leaving off receiver defaults to current object
- No ‘keyword arguments’ but commonly accomplished with hashes

Access Control

- Determined dynamically
- Public is the default except for initialize method
- Protected
  - Access by any instance of class
  - Includes subclasses
- Private methods can be called only in the context of the current object

Conditionals

- Boolean expressions
  - Any value that is not *nil* or the constant *false* is true
  - && and || operators are shorcircuiting
  - defined? operator checks its parameter
  - ||= is a common idiom to assign a default value to a nil element

If and Unless

- Similar to other languages - *if*, *then*, *elsif*, *else*, *end*
- *unless* is negated form
- then is mandatory only on a single-line if statement
- tack on to end of normal statement to use as conditional modifiers

Iterators

- *while* and *until* are built-in
- for … in is syntactic sugar, translated automatically to a call to the *each* iterator method
- Other iterators are *times*, *upto*, *downto*, and *step* and work with numerics

Exceptions

- Ruby has hierarchy of exceptions
  - Catch them using *begin*, *rescue*, *ensure*, and *end*
  - Define multiple rescue clauses or specify more than one exception to catch as parameters
  - Exception object is available as $! or use hash notation to assign variable names on the rescue clause
- Use *retry* after attempting to fix an exception to rerun the begin clause
- Throw exceptions in your own code by using a *raise* statement
- Subclass *StandardError* or one of its children

Memory Model

- Garbage Collection
  - True mark-and-sweep
  - Works with all Ruby objects
- Multithreading
  - In-process inside the interpreter
  - Completely portable
  - Some negatives like not taking advantage of multi-processor hosts

More Language Features

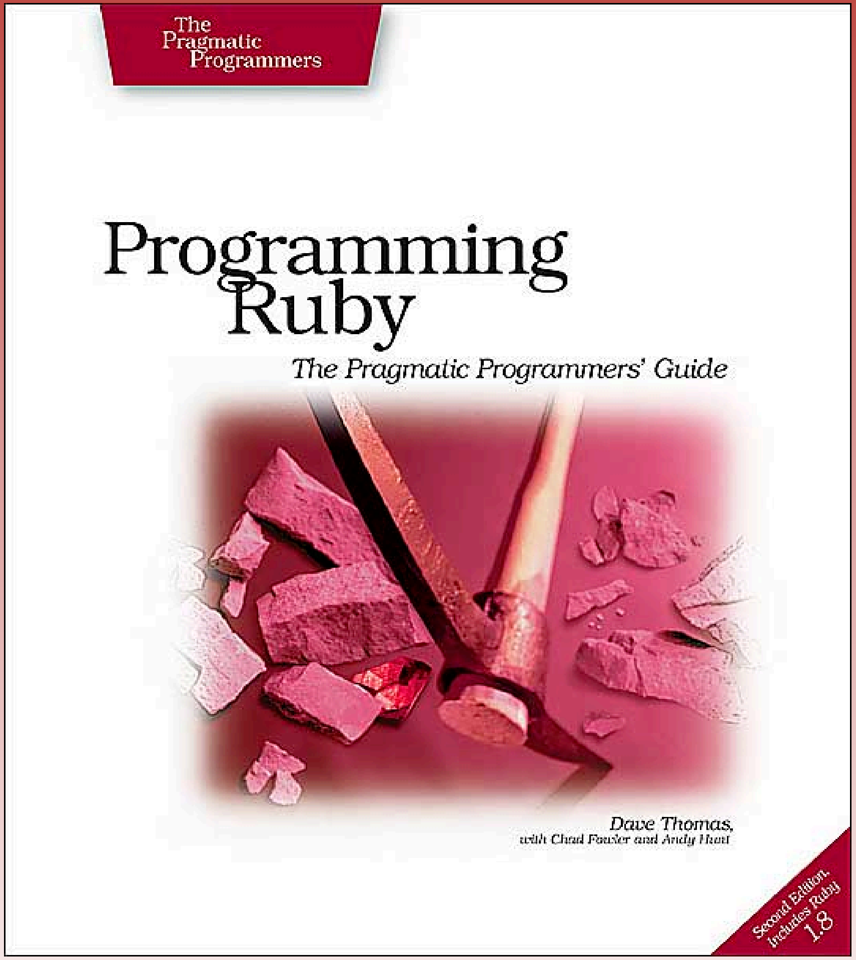
- Portable
  - OS independent threading
  - Can load extension libraries dynamically
- Library support
  - RubyGems package manager
  - Tons of high-quality open source libraries available at RubyForge (similar to CPAN for Perl)

Ruby Tools

The following tools are included with the Ruby distribution

- debugger
- irb – interactive ruby shell
- benchmark
- profiler
- rdoc

The Pickaxe Book


The Pragmatic Programmers
Programming Ruby
The Pragmatic Programmers' Guide
Dave Thomas,
with Chad Fowler and Andy Hunt
Second Edition, Includes Ruby 1.8

Ruby on Rails Fundamentals

“Rails is a full-stack, open-source web framework in Ruby for writing real-world applications with joy and less code than most frameworks spend doing XML sit-ups”

Rails Creator - David H. Hansson

What Is Rails?

‘Kitchen Sink’ MVC Web Application Framework written in Ruby

- ActiveRecord API
- ActionController API
- Templating Engine
- Scripts and lots of other stuff…

Big Productivity Gains

- Convention over configuration
- Everything is in Ruby
- Imposes strong design constraints
- Generators for creating code skeletons and scaffolding

Rails Models

ActiveRecord Model Classes

- Encapsulate persistence logic
- Contain business rules
- Tightly coupled to database tables
- Declare relationships to each other

ActiveRecord Basics

- Extend ActiveRecord::Base
- Don’t declare properties
- Declare relationships to other models with the following ‘macros’
  - belongs_to
  - has_many
  - has_and_belongs_to_many (joins)

Rails Views

User interface done with templates

- HTML with Ruby snippets in .rhtml files
- Easy XML generation in .rxml files
- Lots of HTML and AJAX helper methods
- Sophisticated layout and 'partials' functionality

ERB Template Example

```
<% for post in @posts %>
  Title: <%= post.title %>
<% end %>

All post titles: <%= @post.collect{ |p| p.title }.join ", " %>

<% unless @person.is_client? %>
  Not for clients to see...
<% end %>
```

XML Builder Example

```
xml.rss("version" => "2.0") do
  xml.channel do
    xml.title(@feed_title)
    xml.link(@url)
    xml.description "Basecamp: Recent items"
    xml.language "en-us"
    xml.ttl "40"

    for item in @recent_items
      xml.item do
        xml.title(item_title(item))
        xml.description(item_description(item))
        xml.pubDate(item_pubDate(item))
        xml.guid(@recent_items.url(item))
        xml.link(@recent_items.url(item))
      end
    end
  end
end
```

Rails Controllers

Process requests via action methods that map to URL

- Interact with model classes
- Set any data needed by view as field variables
- Select view to render or redirect

Controller Basics

- Extend ActionController:Base
- Request parameters in ‘params’
  - Naming conventions mean Rails can translate paramaters into a hashtable
  - Handles multi-dimensional data in forms pretty easily
- Web session in ‘session’ hash
- Redirect scope available in ‘flash’ hash for next request only

Actually Working with Ruby on Rails

Based on real-world project experience

Real Productivity Gains Possible

Your results may vary!

- How many developers on team?
- Proficient with Ruby?
- Proficient with Rails?
- Greenfield project or trying to adapt to legacy data sources?

Rails Doesn't Make Data Modeling Any Easier

- Stabilize your schema before ramping up UI development
- Bootstrap your db schema with a migration script
- Write an exampledb rake task

Speaking of Rake

- Dependency-based programming
- So much better than Ant you'll want to use it on all your projects
- Martin Fowler wrote a great article about it at http://martinfowler.com/articles/rake.html

Not much IDE support...

No Intellij for Ruby and none on the horizon for awhile

- Current Ruby editors are kind of crappy
- Eclipse RDT is starting to improve but not there yet (IMHO)
- The nature of dynamically-typed languages means you might want to just use a good text-editor instead

… how to deal with it

- Testing, testing, testing
- Use IRB, the interactive console
- Pair-programming very useful
- Keep Pickaxe and Rails books handy
- ri – command-line documentation

Unit Tests Crucial

- Once you start getting beyond CRUD features you better unit test
- The lack of static typing will get you if you're not used to it
- It's a big namespace so watch out for your code colliding with Rails methods or magic

Learning Ruby

- Think about syntax, cause your editor won’t
- Learn to use the debugger instead of ‘puts’ing around
- Blocks are very powerful. Learn how to use them

Learning Rails

- Don’t overcommit based on initial enthusiasm
- Easy to get started, but there is a learning curve
- Read the Rails book

You'll Write Less Code

- Let Rails do the heavy lifting for you under the scenes
- Ruby is significantly less verbose than Java and C#
- Take advantage of…
  - Convention over configuration
  - Ruby lends itself to writing DSL

Don’t Repeat Yourself

- Refactor, refactor, refactor!
  - Move repetitive view code into helpers
  - Consolidate common page chunks into partials
- Rails has A LOT of built-in functionality (which you won’t know about as a beginner)

Don't Reinvent the Wheel

- Form helpers are your friend
- Rails 'acts_as' methods are very useful
- Look for RAA and RubyForge libraries, particularly to integrate to web services

ActiveRecord Reminder

"When you have this static view of the database, expressed in terms of *n* classes to match your *n* tables, then you tend to solve your problems in those precise terms, because the code generated by the O/R code generation tools will encourage (and perhaps even enforce) such behavior." - Brad Wilson

Take Advantage of ActiveRecord Flexibility

- Remember ActiveRecord works at runtime and doesn't enforce those static views of your data
- The ActiveRecord pattern itself encourages addition of meaningful finder methods like find_specials
- Custom SQL queries can cause additional columns to get "tacked on" to returned objects without extra effort.
  (like for aggregate and other types of calculated columns defined in your SQL select statement)

Belongs_to Table Must Have the Foreign Key

- This can really trip you up, even though it’s repeated multiple times in the docs
- The term “belongs to” is admittedly confusing
- Consider it as “references” or “has reference to”

Don’t Abuse HABTM

Speaking from experience…

- has_and_belongs_to_many can be pretty difficult to understand and use effectively
- Prefer simple has_many/belongs_to relationships where possible
- Once you start adding attributes to the join table, ask yourself if that join actually wants to be a first-class object

ActionMailer Notes

- ActionMailer works great for sending email, particularly the integration with templating
- Receiving email is still nightmarish depending on your platform configuration

AJAX is Easier With Rails but...

- Some of the documentation is poor
- You still have to understand JavaScript to use it effectively
- There are definite “dos and don’ts” around where and why to use AJAX

Other Gotchas

- Learning where it’s okay to use symbols instead of strings tricky
- Lots of methods take hash of parameters and it’s easy to mistype one
- ‘Whiny Nil’ is annoying

Are Ruby and Rails Ready for Prime Time?

No simple answers, so let’s discuss it…

The Future of Ruby

“It sure looks like more than a fad to me.” - Tim Bray

- 10 years of continual development and refinement will continue
- Ruby 2.0 is on the horizon
- No Microsoft or Sun Microsystems stupidity to ruin things ☺
- Compare with Python adoption rates
- Dynamic languages in general gaining wider acceptance in the enterprise

Road to Rails 1.0

80 issues pending (as of 9/20/2005)

http://dev.rubyonrails.com/report/9

- Internationalization tops the list
- Several issues address developer-friendliness with error messages and unknown parameters to API calls
- Minor bugs with acts_as_ features
- RC might be available in October '05

Softer Issues

- Am I ready for a new language and development style?
- Is my team ready?
- Is my company ready?
- Do I have the right projects?

Common Apprehensions

Mostly concerns about '-ilities'

- Maintainability
  - Future availability of Ruby programmers?
  - Quality/readability of code?
  - Platform support?
- Scalability
  - Horizontal vs. Vertical scaling
  - Performance concerns

Is Ruby better than..?

- Better *at what*?
- Some situations need one tool, other situations another tool.
- Performance-wise Ruby is probably slower than Java or C# in real-world situations
- Is it worth worrying about relative performance?

Pragmatic Dave on J2EE

“**Using the full might of a J2EE stack to write a small stand-alone application is using a sledgehammer to crack a nut.** But I keep hearing the sound of nuts being pulverized as developers seem to think that using anything other than J2EE is somehow unprofessional.”

More Pragmatic Dave

- "I'd rather write in a language that let's me focus on the application, and which lets me express myself clearly and effectively."
- **"A better algorithm will easily gain back any marginal performance hit I take for using a slower language."**

Posted on http://blogs.pragprog.com/

J2EE Backlash Fueling Interest in Rails

- “Enterprise Java, has grown into a complex behemoth that consists of layer upon layer of complexity”

  David Geary, author of the best-selling book on Java Server Faces (JSF)

Web 2.0 and Rails

- Fast time-to-market
- Tight integration with AJAX libraries
- New emphasis on focused, interoperable applications
- Less time coding infrastructure means more emphasis on clean design and elegance in all aspects of the application

When should I use Rails?

- Small developer-focused applications
- Opportunities to do parallel development as proof of productivity impact
- Once you are comfortable with Ruby and Rails programming

When not to use Rails!

- Really large applications
- Dealing with legacy databases.
  Hibernate is much better for "schemas-from-hell"
- Unenthusiastic or mediocre developers won't "get it"

The Right Developer Attitude is Crucial

"Agile teams *get* Ruby on Rails sooner than traditional ones"

- Ruby on Rails increases productivity and sheer joy of development
- Reality is some programmers simply don't care about that

Ruby on Rails and Consulting Businesses

- Expect web design shops to continue moving into Rails and away from PHP (it's just too much better not to do so)
- *Faster* projects with *less people* means larger consulting firms might have trouble adapting!

Future of Ruby on Rails

- Ruby is what makes Rails special, the interest in general Ruby programming will continue to expand rapidly
- Tons of similar projects popping up, but none with mindshare and critical mass of Rails project
- Integration of dynamic languages such as Ruby with Semantic Web technologies such as RDF holds significant promise

Questions and Comments

In Conclusion…

Thank you for coming to the presentation!

- Rails has a large enthusiastic community at rubyonrails.org
- ThoughtWorks is seeking Ruby on Rails enterprise projects
- I blog regularly about Rails and agile enterprise topics at obiefernandez.com
- I love getting email… obiefernandez@gmail.com