The journal of Apple technology.

Volume Number: 25
Issue Number: 11
Column Tag: Programming

Apple, Meet Ruby

A gentle introduction

by Rich Morin

Welcome

Mac OS X is very popular with Ruby developers, but most Mac OS X developers are unfamiliar with Ruby. This is unfortunate, because the Ruby language, ecosystem, and community have a lot to offer. If you've been curious about Ruby, read on...

The Ruby Language

The Ruby language was developed by Yukihiro Matsumoto (Matz), as a way to "make programmers happy". It does this by providing powerful features, an elegant syntax, and a very accommodating attitude. The following description, while terse, covers most of the specifics:

Ruby is a dynamic programming language with a complex but expressive grammar and a core class library with a rich and powerful API. Ruby draws inspiration from Lisp, Smalltalk, and Perl, but uses a grammar that is easy for C and Java programmers to learn. Ruby is a pure object-oriented language, but it is also suitable for procedural and functional programming styles. It includes powerful metaprogramming capabilities and can be used to create domain-specific languages or DSLs.

-- "The Ruby Programming Language"

Possibly because the term "scripting language" didn't get enough respect, purveyors of these tools now call them "dynamic programming languages". So, Ruby has all of the convenience (ie, expressive syntax, low overhead, good OS integration) you'd expect to find in a scripting language. However, because it is a "pure object-oriented language", it is capable of handling much larger programming tasks than you might otherwise expect.

Ruby's grammar and dynamic nature (eg, metaprogramming, DSLs) combine to make it very expressive. So, Ruby code tends to be surprisingly small. Even a simple transliteration of Objective-C into Ruby can yield a substantial reduction in code size (a 5x reduction in character counts is quite plausible). This means a lot less code to write, read, and debug.

Ruby supports convenient interactive modes for debugging and general experimentation. As a stand-alone program, Interactive Ruby (irb) allows programmers to try out arbitrary code. Used in the context of a running program (eg, as a Rails "console"), it can allow developers to examine data, try out method calls, etc. Here's a sample irb session, to give you a small taste:

```
% irb -simple-prompt
```

```
>> 2+2
=> 4
>> s = "a string"
=> "a string"
>> s.reverse
=> "gnirts a"
>> ^D
```

Ruby is easy and pleasant to use, while providing a rich (and quite extensible) set of features. Although Ruby's origins are eclectic, Matz has excellent taste in language design. Consequently, Ruby offers (IMHO) a smooth integration of concepts and syntax. For details, see my weblog entry, "How I arrived at Ruby".

However, if Ruby is such a great language, who's using it? A few years ago, there wasn't a very good answer to this. Although Ruby was reputed to be "big in Japan", it was mostly used by system administrators, language aficionados, etc. The scripting API for Google SketchUp, although useful and fun to play with, did not exactly put Ruby in the big leagues.

Ruby rose to prominence, however, as the foundation for David Heinemeier Hansson's web development framework, Ruby on Rails. As Rails became popular, many web developers also learned Ruby. Other Ruby-based web frameworks (eg, Merb, Rack, Sinatra, Waves) have since been developed, making Ruby an important player in web development.

Any substantial web site uses a mixture of technologies: databases, routers, servers, etc. So, web developers have to be facile in a variety of languages and tools. Ruby is important as the "glue language", but the heavy lifting may well be done by tools written in C, Erlang, Java, etc.

Although Ruby can be run on any modern operating system, most Ruby code is developed on Mac OS X. In particular, Apple laptops dominate the picture at Ruby conferences, hackfests, and meetings. OSX-only tools such as Keynote and TextMate dominate the presentations.

Ezra Zygmuntowicz took advantage of this fact in a talk on cloud computing technologies (eg, chef, nanites, rabbitmq) at RailsConf 2009. He had all of the Mac users in the room download some server code, then ran some parallel-processing demonstrations from the podium, causing a chorus of Macs to "say" snippets of text.

Testing

The Ruby community emphasizes testing very strongly, putting it in the center of the development process. In fact, the definition for the Ruby language itself (including assorted variations) is captured by RubySpec, a test suite containing tens of thousands of executable "specs".

Tools and practices for "behavior-driven development" (BDD) and "test-driven development" (TDD) are also very popular. In these approaches, developers are encouraged to write tests first, verifying that they do not pass. Then, developers write enough code to make the tests pass. Once everything is "green", they are free (and encouraged) to refactor (ie, tidy up) the code.

This produces various levels of tests (eg, unit, functional, integration), giving the project a useful (and very comforting) "safety net" for changes. In many shops, "continuous integration" (CI) tools run the test suite after each code check-in. If a developer makes a change that breaks a test, the CI suite will let the developer (and maybe the entire shop :-) know about it.

Because most Ruby development is done on Rails (ie, web-based) applications, most Ruby deployment uses Linux-based PCs. The reasons are fairly straightforward. Commodity PCs make very economical servers. Also, Linux supports a variety of technologies (eg, clustering, virtual machines) that are useful for servers.

So, Rails developers have created a large number of tools for deployment, remote testing, etc. Many of these are also useful for network administration tasks, such as installing and configuring user applications.

In fact, one of the biggest differences between Mac and Ruby development is the wealth of community-based libraries, frameworks, and other tools. Instead of relying on Apple (and a handful of third-party vendors), the Ruby community develops its own infrastructure and shares it freely.

Implementations

Leopard ships with the original Ruby (1.8.6) implementation, known colloquially as MRI (Matz's Ruby Interpreter). Snow Leopard increments the version to 1.8.7. Many other variants are available or under development, including:

- Duby - Ruby-like syntax with static types
- ERuby - Embedded Ruby (for templating)
- HotRuby - Ruby on JavaScript and Flash
- IronRuby - Ruby on .NET (DLR)
- JRuby - Ruby on Java
- MacRuby - Ruby on Objective-C
- MagLev - Ruby with object persistence
- Rubinius - Ruby on Ruby and C
- RubyCocoa - MRI with a Cocoa bridge
- YARV - MRI, the next generation

Some of these variants change the language itself; others add features or provide access to particular run-time environments. RubySpec (the executable specification suite) provides guidance to developers of alternative implementations. Not surprisingly, RubySpec is also a critical resource as Ruby moves to new versions (eg, from 1.8.6 to 1.9 and 2.0).

The most interesting of these variants, from the perspective of a Mac developer, is MacRuby. MacRuby allows Ruby programs access to the entire range of Objective-C capabilities, including some that the base language makes difficult or impossible (eg, redefining ObjC methods at runtime). All without using a bridge (like RubyCocoa or PyObj-C).

The experimental branch, as of early July, uses LLVM to perform Just In Time (JIT) compilation of Ruby into machine code. Ahead Of Time (AOT) compilation is also in prospect, so the ability to run MacRuby code in sanctioned iPhone apps appears quite likely to present itself within a matter of months. Currently, as of beta 5, MacRuby can create a Mach-O object from Ruby source code. Stay tuned...

The Ruby Community

Rubyists are a motley crew. Some of us came because of the language itself. If so, our background may be in another dynamic language such as JavaScript, Lisp, Perl, PHP, Python, or Smalltalk. Others, who came to Ruby because of Rails, may have Java, JavaScript, and/or PHP experience. This diversity makes conversations interesting, to say the least!

Rubyists also tend to be "early adopters" of new technology, both in and out of the Ruby language. For example,

CouchDB, Erlang, and jQuery get a lot of attention, despite the fact that they have nothing directly to do with Ruby. Basically, Rubyists are pretty agnostic about technology origins and implementation details, as long as it meets their needs.

This also extends to various aspects of "social computing". About a year ago, the entire Rails community switched to Git and GitHub for revision control and code exchange. Rubyists also tend to be active participants on email lists, IM, IRC, Twitter, wikis, etc. Finally, Internet-based videos (eg, screencasts, conference talks) are very popular as a way to present ideas and demonstrate technology.

Conference Videos

Speaking of videos, I really love the ability to attend Ruby conferences from the comfort of my office chair. I don't have to worry about missing a parallel track, never get stuck in a boring or irrelevant session, and have the ability to pause or back up when I start getting lost.

Confreaks is by far the biggest source of Ruby-related conference videos, but events such as MerbCamp and the South Carolina Ruby Conference are also recorded. My weblog entry, "Video Resources for Rubyists", has a relatively complete list.

Even when the entire conference isn't recorded for posterity (tsk!), occasional presentations may be recorded. O'Reilly, for example, commonly records keynotes. I'm hoping they will start recording all of their conference sessions, but keynotes are certainly better than nothing.

Live Conferences

Of course, videos do not provide the full conference experience. Less travel and dislocation, to be sure, but no opportunity for hallway discussions, BOFs, etc. So, taking in an occasional conference is quite worthwhile. This year, I have attended GoGaRuco (San Francisco) and RailsConf (Las Vegas); a couple of others are on my "wish list".

More than a dozen Ruby or Rails conferences are held each year. Some, like RailsConf and RubyConf, are big-tent, multi-track events; others tend to be smaller, single-track events. Both styles have their advantages and disadvantages.

About half of the Ruby and Rails conferences are held in the USA (eg, CA, DC, FL, HI, NC, NV, OH, TX, UT). Others are held around the world: Argentina (Locos X Rails), Australia (Rails Camp), Canada (FutureRuby, Ruby in the Rain), Europe (EuRuKo, Rails Konferenz, Ruby Fools, Scotland on Rails), and Japan (RubyKaigi). There are even some specialized events, such as the one-day "Ruby on OS X" conference and "erubycon" (Enterprise Ruby Conference). A web search (eg, "Conference Rails Ruby") will bring up lots of listings.

Local Groups

There are dozens of local groups for Ruby and Rails developers. Meetings often have invited speakers, but "hack sessions" and "lightning talks" are also popular. Announcements (eg, firms looking for Rails developers) are also a common feature.

Although groups are scattered around the world, the distribution is far from even. In the San Francisco Bay Area, for example, we have half a dozen groups that meet regularly and a few other events (eg, hackfests, seminars) that surface on occasion. The Ruby Brigade and Meetup pages are useful for finding (and if need be, starting) new groups.

Books

I'm a big fan of technical books, in both paper and online formats. I mark up the paper ones with errata and notes, add Post-it tabs liberally, and generally "make them my own". I find the online versions harder to read (and impossible to mark up), but much better for some kinds of searching and rapid access. So, I often get both versions.

As a MacTech reader, you're likely to be familiar with at least a few programming languages. Objective-C, in particular is de rigeur for Mac OS X application developers. So, I'm going to suggest some books that should help you get going in Ruby, without wasting your time.

Programming Ruby is the canonical "handbook" for Ruby. It contains a fairly complete introduction to the language, covers the standard classes and modules quite well, and covers a smattering of ancillary topics (eg, common add-on libraries). Versions are available for both the traditional (1.8) and upcoming (1.9) versions of the language.

The Ruby Programming Language is a definitive reference for Ruby (after all, Matz is a co-author!). It covers both Ruby 1.8 and 1.9, making comparisons as needed. I've been working my way (carefully) through the book, learning about language details I've missed in the past. I'm also working on *Ruby Best Practices*, which covers a lot of (deservedly) popular Ruby programming idioms.

Design Patterns in Ruby, despite its title, might well be the best introduction to Ruby for an Objective-C programmer. It begins with a slightly simplistic description of the language, then dives into assorted "design patterns". Typically, it begins with an approach that mimics that of the original (ie, in Design Patterns), but it often uses that as a starting point to discuss more and more Rubyish approaches.

The Well-Grounded Rubyist is a bit of a sleeper. The early chapters are paced pretty slowly, with rather elementary material. However, the author is able to maintain the same sedate (and unthreatening) pace as he delves further and further into Ruby arcana. So, if you really want to learn how to "do Ruby", this book is a must-read.

There are a variety of "cookbooks" for Ruby. I find them to be quite handy when I'm looking for an idiom or simply an example of how to use a particular part of the language. I use *Ruby Cookbook* a lot, but also dive into *The Ruby Way* and *Enterprise Integration with Ruby* on occasion. The latter book is particularly useful for topics such as LDAP and XML.

Bibliography and References

Here are citations for some of the books and web sites mentioned above:

Black, David A. *The Well-Grounded Rubyist*. Manning, 2009.

Brown, Gregory T. *Ruby Best Practices*. O'Reilly, 2009.

Carlson, Lucas and Richardson, Leonard. *Ruby Cookbook*. O'Reilly, 2006.

Flanagan, David and Matsumoto, Yukihiro. *The Ruby Programming Language*. O'Reilly, 2008.

Fulton, Hal. *The Ruby Way*. 2nd. Edn. Addison-Wesley, 2006.

Gamma, Erich, et al. *Design Patterns*. Addison-Wesley, 1995.

Schmidt, Maik. *Enterprise Integration with Ruby*. Pragmatic Bookshelf, 2006.

Thomas, Dave, et al. *Programming Ruby*. 2nd. Edn. Pragmatic Bookshelf, 2005.

Thomas, Dave, et al. *Programming Ruby 1.9*. Pragmatic Bookshelf, 2009.

Git - http://git-scm.com

GitHub - http://github.com

Google SketchUp Ruby API - http://code.google.com/apis/sketchup

IronRuby - http://www.ironruby.net

JRuby - http://jruby.codehaus.org

LLVM - http://llvm.org

MacRuby - http://macruby.org

MagLev - http://maglev.gemstone.com

Merb - http://merbivore.com

Rack - http://rack.rubyforge.org

Rich Morin's weblog - http://www.cfcl.com/rdm/weblog

Rubinius - http://rubini.us

Ruby APIs - http://www.ruby-doc.org

Ruby on OS X - http://rubyonosx.com

Ruby on Rails - http://rubyonrails.org

RubyCocoa - http://rubycocoa.sourceforge.net

Sinatra - http://www.sinatrarb.com

TextMate - http://macromates.com

Waves - http://rubywaves.com

YARV - http://www.atdot.net/yarv

Aloha on Rails - http://www.alohaonrails.com

Confreaks - http://www.confreaks.com

erubycon - http://erubycon.com

Lone Star Ruby Conference - http://lonestarrubyconf.com

Rails Konferenz - http://rails-konferenz.de

RailsConf - http://railsconf.com

Ruby Brigade - http://rubybrigade.org

Ruby Meetups - http://ruby.meetup.com

RubyConf - http://rubyconf.org

RubyKaigi - http://rubykaigi.org

Rich Morin has been programming computers since 1970, using a variety of languages and operating systems. He provides technical editing and writing, programming, and web development services, using (primarily) Ruby on Mac OS X.