H3RALD

Book Review: Distributed Programming with Ruby

Just what you need to get started with the right tools to build large and scalable applications in Ruby

by *Fabio Cevasco*

February 2013

Originally published on H3RALD.com

Back when I read *Programming Ruby* for the first time, I distinctly remember a short reference to dRb, the **D**istributed **R**u**b**y library included in the Standard Library.

"Cool!" — I thought

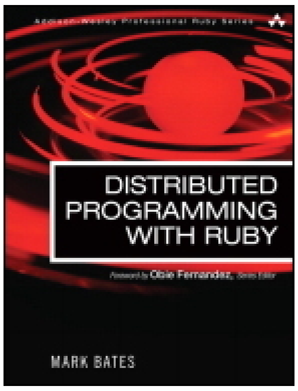
...and that was pretty much it. The documentation for DRb was pretty much nonexistent (at the time), I didn't need it, so I pretty much forgot about it altogether until this book came out.

Distributed Programming with Ruby fills a very particular niche of the Ruby programming world: *distributed* programming. Moreover, this book is somehow *justified* by the scarce documentation on the subject:

> *Although these libraries [DRb and rinda] have been included with Ruby for many years now, they have received little or no attention (or documentation). This has led to a lot of FUD (fear, uncertainty, and doubt) about what these libraries can and cannot do, and when they are appropriate to use (if at all).*
>
> — Mark Bates, *Distributed Programming with Ruby*

But there's more. This book gives the reader a complete overview of what's out there, in the Ruby world, to support distributed programming. This includes quite a few gems and libraries besides the ones provided in the standard library.

Overview



The book is organized into four parts, each dealing with a particular set of Ruby libraries related to distributed programming.

The author, Mark Bates, does a good job maintaining a sort of continuity in the examples throughout the book: you'll get accustomed to a *Logger* class of some kind being punctually re-implemented more or less once per chapter, using a different library.

Additionally, the libraries described in the book are ordered by "reverse preference" in each part of the book, so normally the libraries described later on in a part fix some of the shortcomings of the preceding ones.

Part I: Standard Library

This part is the most important of all: it gives you the very basics about Distributed Programming and it describes the "building blocks" (DRb and Rinda) used in nearly all the other libraries described in the book. If you want you can skip some chapters in the other parts of the book, but make sure this part is crystal clear in your head before proceeding any further.

Part II: Third-Party Frameworks and Libraries

If you read part I, you're probably a bit disappointed by DRb and Rinda and the amount of code you have to write to make simple things work in a distributed environment. The good news is that there are some Ruby gems out there that can make life simpler:

- RingyDingy
- Starfish
- Distribunaut
- Politics

Part III: Distributed Message Queues

In this part, the author introduces more in detail the concept of distribute message queues, and also the technologies and protocols available not only in the Ruby world but elsewhere. It focuses on two libraries:

- Starling, originally used by Twitter.
- AMQP, an implementation of the AMQP protocol in Ruby, that can be used in conjunction with RabbitMQ, an Erlang-based messaging system.

Part IV: Distributed Programming with Ruby on Rails

The book ends somewhat abruptly with this last part that deals with distributed programming in the Rails world. It feels a bit like a last-minute addendum that I would have left for an appendix, nevertheless it briefly introduces BackgrounDRb and Delayed Job.

Technical Analysis

Unlike other technical books, this one can (must?) be read sequentially. Generally each chapter focuses on a library, describes how to install it and use it, and highlights its pros and cons. Typically, the "cons" are solved in the following chapter by another library, and so on…
The book is not meant to contain a full technical reference of each library, and it's quite short (256 pages), so you really get the most out of it if you read it all, from start to finish. I didn't realize there were so many different libraries in this particular niche of Ruby programming, and Mark does a good job demistifying some of them.
One thing that really struck me out of this book is the focus on gems. We're not talking about *mainstream* frameworks like Rails or Merb here, but rather of some rather specialized, smaller libraries that fullfill very specific tasks. Personally, I don't remember any other Ruby book doing this in the same way, and I was quite happy about it.
On the other hand, gems are a double-edged sword: while some of them are really cool and well-maintained, others may disappear tomorrow with no prior notice. I was actually very surprised to see even some of the *quirks* of these gems documented in the book:
p91: *"Notice that we added client{ } to the bottom of the server file. The reason for this appears to be a bug or flaw in the Starfish architecture."*
Really? Hasn't it be fixed now? Apparently not, that's the way it works, so no, you can't blame the author of the book for this.

Formatting and Readability

As I pointed out earlier, this book is somehow meant to be read sequentially, and Mark does a good job making sure you don't get bored. Chapters and sections are quite short and there's a good text/code ratio: the examples are short and clear, and you don't have to try them out yourself, because most of the time the author does it for you. It's not infrequent for the author to tell you to run "wrong" code, but that's a great way to show you how to do the right thing right afterwards.
Sidebars and boxes are used properly and they do provide actual value-added content: some information on a non-Ruby technology, some tips and tricks on how to run things smoothly, etc. On the other hand, one thing I couldn't stand were the *endnotes*. I must say I don't like endnotes at the best of times, but when they

are pointless I just can't suffer them. Each chapter has its own fair share of endnotes, but unfortunately most of them are just URLs to Wikipedia pages or RubyForce/GitHub projects: I would have preferred the URLs inline with the rest of the text, but that's just me.

Style and Contents

Mark has a nice, informal writing style. Exactly what you expect from a programming book nowadays, even if sometimes it feels a bit too informal:
p86: *"I think I understand what Eric means by all that. However, that is as deep as the documentation goes on the subject. I have not been able to test what I think he means, so I won't make any grand promises about what the library can and cannot do in regards to expiring/ renewing registrations."*
Although this is not something you'd see in a professional book everyday, it definitely helps to connect with the reader: Mark is one of us after all, even if he happens to have created quite a few interesting projects, like the Mack framework, the Distribunaut library (which is also mentioned in his book, but in a very impartial way) and Configatron. From his book you understand that he's neither one of those rockstar developers nor one of those famous authors who just writes books for a living: he's a competent programmer who knows quite a bit about a particular, but relevant, niche of Ruby programming.

Final Thoughts

This is one of those books I'd like to see a second edition of. Partly because there are some relatively new gems which have been left out (BrB, for example), partly because this is a rather hot topic at the moment, and different solutions are popping out at a rather extreme rate.
The decision to write about mainly about gems was bold but necessary, and I'd really like to see more authors doing that, but with extra care. From reading this book, you understand that there's no *silver bullet* when it comes to Distributed Programming, but rather different tools to do different jobs.
The thing I missed the most? A proper conclusion to the book. You're left with two chapter about Rails-specific libraries which could have easily become appendixes, and nothing else. I would have liked a sort of "summing up" end chapter (re-)highlighting the pros and cons of each library and a sort of feature matrix.
Nevertheless, it was well worth my time and it proved to be a very good resource to get started in writing distributed Ruby programs.