Extracted from:

Crafting Rails 4 Applications

Expert Practices for Everyday Rails Development

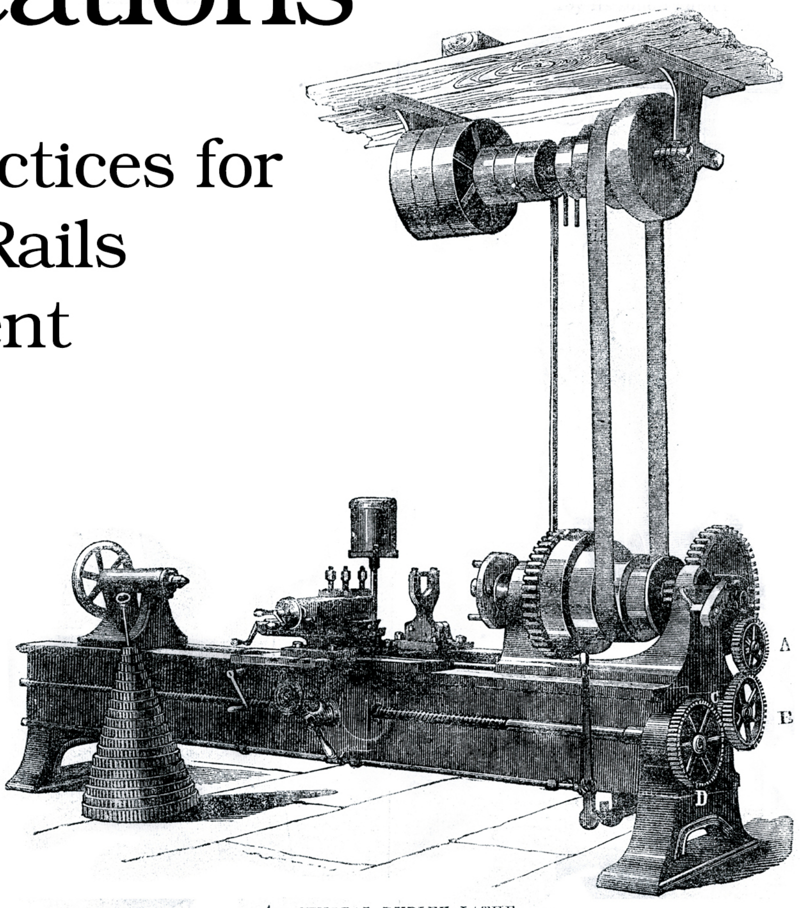
The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Crafting Rails 4 Applications

Expert Practices for Everyday Rails Development



4.—CENTRAL DUPLEX LATHE.

José Valim

edited by Brian P. Hogan

The Facets of Ruby Series

Crafting Rails 4 Applications

Expert Practices for Everyday Rails Development

Jose Valim

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

Printed in the United States of America.
ISBN-13: 978-1-937785-55-0
Encoded using the finest acid-free high-entropy binary digits.
Book version: B2.0—June 21, 2013

Preface

When Rails was first released in 2004, it revolutionized how web development was done by embracing concepts like Don't Repeat Yourself (DRY) and convention over configuration. As Rails gained momentum, the conventions that were making things work so well on the golden path started to get in the way of developers who had the urge to extend how Rails behaved or even replace whole components.

Some developers felt that using DataMapper instead of Active Record was a better fit. Other developers turned to MongoDB and other nonrelational databases but still wanted to use their favorite web framework. Then there were those developers who preferred RSpec to Test::Unit. These developers hacked, cobbled, or monkey-patched solutions together to accomplish their goals because previous versions of Rails did not provide a solid API or the modularity required to make these changes in a clean, maintainable fashion.

With time, Rails started to listen to those developers and after years, the end result is a robust and wide set of plugin APIs, targetted to developers that want to customize their workflows, replace whole components, bending Rails to their needs without messy hacks.

This book guides you through these plugin APIs through practical examples. In each chapter, we will use test-driven development to build a Rails plugin or application that covers those APIs and how they fit in the Rails architecture. By the time you finish this book, you will understand Rails better and be more productive while writing more modular and faster Rails applications.

Who Should Read This Book?

If you're an intermediate or advanced Rails developer looking to dig deeper and make the Rails framework work for you, this is for you. We'll go beyond the basics of Rails; instead of showing how Rails lets you use its built-in features to render HTML or XML from a controller, we'll show you how the render() method works so you can customize it to accept custom options, such as :pdf.

Rails Versions

All projects in *Crafting Rails Applications* were developed and tested against Rails 4.0.0. Future stable releases, like Rails 4.0.1, 4.0.2, and so forth, should be suitable as well. You can check your Rails version with the following command:

```
rails -v
```

And you can use gem install to get the most appropriate version:

```
gem install rails -v 4.0.0
```

This book also has excerpts from Rails' source code. All these excerpts were extracted from Rails 4.0.0.

Most of the APIs described in this book should remain compatible throughout Rails releases. Very few of them changed since the release of the first edition of this book.[1].

Note for Windows Developers

Some chapters have dependencies that rely on C extensions. These dependencies install fine in UNIX systems, but Windows developers need the DevKit,[2] a toolkit that enables you to build many of the native C/C++ extensions available for Ruby.

Download and installation instructions are available online at http://rubyinstaller.org/downloads/.

Alternatively, you can get everything you need by installing RailsInstaller,[3] which packages Ruby, Rails, and the DevKit, as well as several other common libraries.

What Is in the Book?

We'll explore the inner workings of Rails across eight chapters.

In Chapter 1, *Creating Our Own Renderer*, on page ?, we will introduce rails plugin, a tool used throughout this book to create Rails plugins, and customize

1. http://www.pragprog.com/titles/jvrails/
2. http://rubyinstaller.org/downloads/
3. http://railsinstaller.org

render() to accept :pdf as an option with a behavior we will define. This chapter starts a series of discussions about Rails' rendering stack.

In Chapter 2, *Building Models with Active Model*, on page ?, we will take a look at Active Model and its modules as we create an extension called Mail Form that receives data through a form and sends it to a preconfigured email.

Then in Chapter 3, *Retrieving View Templates from Custom Stores*, on page ?, we will revisit the Rails rendering stack and customize it to read templates from a database instead of the filesystem. At the end of the chapter, we will learn how to build faster controllers using Rails' modularity.

In Chapter 4, *Sending Multipart Emails Using Template Handlers*, on page ?, we will create a new template handler (like ERB and Haml) on top of Markdown.[4] We'll then create new generators and seamlessly integrate them into Rails.

And in Chapter 5, *Streaming Server Events to Clients Asynchronously*, on page ?, we will build a Rails engine that streams data to clients. We will also see how we can use Ruby's Queue class in the Ruby Standard Library to synchronize the exchange of information in between threads, finishing with a discussion about thread safety and eager loading.

In Chapter 6, *Writing DRY Controllers with Responders*, on page ?, we will study Rails' responders and how we can use them to encapsulate controllers' behavior, making our controllers simpler and our applications more modular. We will then extend Rails responders to add HTTP Cache and internationalized Flash messages by default. At the end of the chapter, we'll learn how to customize Rails' scaffold generators for enhanced productivity.

In Chapter 7, *Managing Application Events with Mountable Engines*, on page ?, we will build a mountable engine that stores information about each action processed by our application in a MongoDB database and exposes them for further analysis through a web interface. We will finish the chapter talking about Rack and Rack' middleware stacks while writing our own middleware.

Finally, in Chapter 8, *Translating Applications Using Key-Value Backends*, on page ?, we will learn about I18n and customize it to read and store translations in a Redis data store. We will create an application that uses Sinatra as a Rails extension so we can modify these translations from Redis through

4. http://daringfireball.net/projects/markdown

a web interface. We will protect this translation interface using Devise[5] and show Capybara's[6] flexibility to write integration tests for different browsers.

Changes in the Second Edition

All of the projects and code examples have been updated and tested to work with Rails 4. The projects also use more up-to-date workflows for creating Rails plugins and interfacing with the framework.

In addition, Chapter 5, *Streaming Server Events to Clients Asynchronously*, on page ? is brand new and covers Rails' support for Server Sent Events, and digs into eager loading and thread safety. We also explore isolated and mountable engines and single file Rails applications in this edition.

How to Read This Book

We'll build a project from scratch in each chapter. Although these projects do not depend on each other, most of the discussions in each chapter depend on what you learned previously. For example, in Chapter 1, *Creating Our Own Renderer*, on page ?, we discuss Rails' rendering stack, and then we take this discussion further in Chapter 3, *Retrieving View Templates from Custom Stores*, on page ? and finish it in Chapter 4, *Sending Multipart Emails Using Template Handlers*, on page ?. In other words, you can skip around, but to get the big picture, you should read the chapters in the order they are presented.

Online Resources

The book's website[7] has links to an interactive discussion forum as well as errata for the book. You'll also find the source code for all the projects we build. Readers of the ebook can click the gray box above the code excerpts to download that snippet directly.

If you find a mistake, please create an entry on the errata page so we can address it. If you have an electronic copy of this book, there are links in the footer of each page that you can use to easily submit errata to us.

Let's get started by creating a Rails plugin that customizes the render() method so we can learn how Rails' rendering stack works.

José Valim

jose.valim@plataformatec.com.br

April 2013

5. https://github.com/plataformatec/devise
6. https://github.com/jnicklas/capybara
7. http://www.pragprog.com/titles/jvrails2/