Extracted from:

Designing Elixir Systems with OTP

Write Highly Scalable, Self-Healing Software with Layers

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Designing Elixir Systems with OTP

Write Highly Scalable, Self-Healing Software with Layers

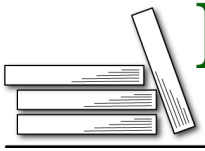
James Edward Gray, II
Bruce A. Tate
edited by Jacquelyn Carter

Designing Elixir Systems with OTP

Write Highly Scalable, Self-Healing Software with Layers

James Edward Gray, II
Bruce A. Tate

The Pragmatic Bookshelf
Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Executive Editor: Dave Rankin
Development Editor: Jacquelyn Carter
Copy Editor: Jasmine Kwytin
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-661-7
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—December 2019

Introduction

In October 2018, we were gathered with some family and friends in a mock Chattanooga train station working to solve a fictional puzzle so we could escape. We had burned through most of our clock and were calling for our next clue. We scrambled to take this last bit of information and translate it to the various combination locks and levers that would let us out of the room. Eventually the host called through the intercom that we'd failed. We'd run out of time.

Roughly two years before, we started working on an advanced book about OTP. We knew that Elixir developers were starting to push the set of tools beyond the basic libraries and books that were on the market at that time. They wanted a way to express increasingly complex code in ways that would scale and hold up to years of revision.

We set ourselves to this effort with a will and fell short. It seemed that we would run out of time, or patience, or will. Some days we came up with outlines that looked like a watered down table of contents for better books. Others we wrote chapters that had nuggets of wisdom presented awkwardly. Sometimes life just got in the way. The train was all but dead and we hauled it back to the station.

Luckily, not every project has a time limit. The last few months seem like we've just been given a clue, the cheat codes that helped us start to pressurize the boiler in this train to get the wheels turning again. These insights helped us break through:

- We didn't want to write strictly about OTP. Sometimes OTP is the *wrong* thing to do. The first half of this book does not cover OTP at all!
- We didn't want to write about simplicity. We wanted to write about revealing complexity piece by piece in layers.
- We wanted to present material that developers could remember and take with them.

With these ideas in mind, we rebooted the project. The boiler pressure built enough that our wheels started to turn and we pulled out of the station once again.

Worker-Bee-Driven Design

James came up with a great way to generalize the layers for a typical OTP project, which led to the sentence "Do fun things with big, loud worker-bees" to remember the layers: data, functions, tests, boundaries, lifecycles, workers. We shared these ideas with some trusted advisors and they resonated strongly. We began to experience an unfamiliar feeling of blessed momentum!

All at once, with that system of layering we had the overarching structure for our table of contents. We could finally imagine the book that Elixir developers have long desired. The system of layers gave us a framework for expressing the deep wisdom we've collected and the simple layers let us express those ideas in a way our readers could understand and digest piecemeal.

James picked the perfect project for the book and we could immediately imagine what the layers in our software would look like and how to present each piece to the user. As we used all of these layers together in the context of a complex project, it *felt right*. We had discovered WDD, or worker-bee-driven design. As we continue to write software, we can testify that the approaches work.

We hope these layers have the same impact on your software that it has had on our book. We hope they feel like cheat codes that completely unlock your thought processes so you can escape some of the concurrency ceremony and move on to the hard pieces of your problem.

Who Should Read This Book

Hopefully, you have a rough idea of the work we'll be doing together. We'll examine design through layers.

In this book, we're addressing intermediate and advanced programmers who want a better understanding of how to design Elixir projects. We'll offer advice in this book that may conflict with concepts you've seen elsewhere, but that's OK. You can take what you like and leave the rest behind.

If you are an Elixir beginner, this book will be for you eventually, but not yet. You should take advantage of one of the many excellent Elixir books and courses available, including *Programming Elixir 1.6 [Tho18]* by Dave Thomas.

If you would like to focus on programming user interfaces and want to skip the heavy back-end designs, you'd be better off reading *Programming Phoenix 1.4 [TV19]*. Similarly, if you're concerned with pure database programming, *Programming Ecto [WM19]* is the book you'll want to check out instead.

Online Resources

You can get the code from the book page on the Pragmatic Bookshelf website.[1] We hope that when you find errors or suggestions that you will report them via the book's errata page.[2]

If you like the book we hope you'll take the time to let others know about it. Reviews matter, and one tweet or post from you is worth ten of ours! We're both on Twitter, and tweet regularly. Find James at @jeg2 and Bruce at @redrapids. You can also drop notes to @pragprog!

We're excited to head down the tracks with you. We hope you enjoy it as much as we have.

James E. Gray, II and Bruce A. Tate
December 2019

1. https://pragprog.com/book/jgotp
2. https://pragprog.com/titles/jgotp/errata