The Pragmatic Programmers

Your Elixir Source

Real-World Event Sourcing

Distribute, Evolve, and Scale Your Elixir Applications

Kevin Hoffman

edited by Kelly Talbot

This extract shows the online version of this title, and may contain features (such as hyperlinks and colors) that are not available in the print version.

For more information, or to purchase a paperback or ebook copy, please visit https://www.pragprog.com.

Preface

I can still remember the first time I needed event sourcing without knowing it. I'd trapped a handful of players in an elevator in an online multiplayer game (it was text-based, and it was so long ago we had to share our keyboards with dinosaurs and remnants of the ice age). My code naively set values on data in direct response to events (like angry players pushing a button). As a result, all it took was a couple of players to hit the button around the same time to trap them in a void halfway between upstairs and downstairs.

At some point many moons after that disaster, I came to the realization that *everything* is event sourced. This new understanding brought me joy and returned some of the awe of learning to my job again. I wanted to share this joy with everyone, but never felt qualified to do so. Finally, after some (redacted) number of years of building software and learning from having failed to build software properly, I have both the joy and the qualifications I've always needed to write this book.

This book is opinionated, but its goal isn't to make sure you blindly adopt all of my opinions. Rather, its goal is to give you the knowledge, hands-on experience, and information you need to decide whether or not you agree with my opinions, and which ones you will embrace for your next event sourcing project.

This book will teach you how to event source everything through examples written in Elixir, and how to avoid many of the pitfalls common to first attempts at building event sourced apps. My singular hope is that when you reach the end of it, event sourcing will bring you as much joy as it does me and you will go on to build amazing things.

Who Should Read This Book?

This book is for the curious. It's for developers who build app after app, service after service, and wonder if if there's a better way. If you've ever been curious about Event Sourcing, or wondered if there was a way to create more reliable,

testable, and stable systems and maybe even simplify your code at the same time, then this book is for you. If it's always bugged you that an application can give you its current state, but can't give you any guarantees about the reliability of that data or even how it came to be, then you'll definitely want to read this book.

The language of choice for this book is Elixir. While some previous knowledge of Elixir will definitely help with running and understanding some of the samples, it isn't strictly required. Many of the samples should be easy enough to read by anyone with exposure to functional programming languages. Some more advanced content leverages GenServer[1] and GenStage,[2] so if you're unfamiliar with those concepts you may want to read up on them before you progress beyond the fundamentals section (the first four chapters).

Event sourcing, at least as this book covers it, is primarily a back-end technology. If you're curious about the use of event sourcing in front-end technology, then I suggest checking out languages like Elm[3] and the myriad libraries and tools available for JavaScript/TypeScript like Redux.[4]

About This Book

This book opens with an overview of the fundamental building blocks of event sourcing: aggregates, projectors, notifiers, injectors, and process managers. Once you have a firm grasp of those fundamentals, it's time to start applying those by using libraries, external data stores, persistent streams, and more. Finally, the book aims you firmly at production by covering modeling, model discoverability and documentation, testing, security, and preparing for scale. These topics are organized into the following chapters:

Chapter 1: Building Your First Event-Sourced Application

Whet your appetite with an introduction to event sourcing, what it is, and why we want to use it. Learn about the first building block of event sourcing, the aggregate.

Chapter 2: Separating Read and Write Models

Learn to separate the read and write models, what projections are, and when and how to use them.

1. https://hexdocs.pm/elixir/GenServer.html
2. https://hexdocs.pm/gen_stage/GenStage.html
3. https://elm-lang.org
4. https://redux.js.org

Chapter 3: Enforcing Perimeters with Injectors and Notifiers

Interact with external systems and maybe even the "real world" using notifiers and injectors.

Chapter 4: Exploring the Saga of the Process Manager

Model event flows, sequences, and long-running activities using process managers.

Chapter 5: Building Event-Sourced Elixir Apps with Commanded

Learn about Elixir's defacto standard event sourcing library, Commanded.

Chapter 6: Building Resilient Applications with Event Stores

This chapter will introduce you to the criteria for choosing an event store and when and why you need them.

Chapter 7: Evolving Event-Sourced Systems

In systems that are supposed to be immutable, learn techniques and rules for how those systems can evolve over time to meet new demands and business requirements.

Chapter 8: Securing Event-Sourced Applications

Security can never be an after-thought. This chapter will discuss some of the security concerns that are specific to event-sourced applications.

Chapter 9: Testing Event-Sourced Systems

One of the most powerful benefits of event sourcing is how easily event sourced applications can be tested. Learn some techniques and patterns for making assertions about event flows.

Chapter 10: Modeling Discoverable Application Domains

We rarely ever build applications in a vacuum. In this chapter, learn some tools, techniques, and patterns for modeling event sourced applications and sharing and communicating those models.

Chapter 11: Scaling Up and Out

We finish up our event sourcing journey with a discussion of how scaling event-sourced applications may be different from traditional applications, and why that's a good thing.

Appendix 1: The Laws of Event Sourcing

This reference provides a handy recap of all of the laws of event sourcing that are discussed throughout the book. Of course, it's important to read the book's actual chapters to put them into proper context.

How to Read This Book

This book is neither a reference manual nor is it a guided tour through a single all-encompassing sample. It is best read from start to finish in chapter order. Though some some samples may exist on their own without the code from previous chapters; the knowledge, lessons, and rules are all designed to be read in order and build on the work done earlier.

Conventions Used in This Book

The conventions in this book are generally the same as any other book from *The Pragmatic Bookshelf*, with a few things specific to this material.

Laws of Event Sourcing

Throughout the book I will introduce the so-called *laws* of event sourcing. Each one will appear in an exclamatory block like this one:

All Events are Immutable and Past Tense

Every event represents something that actually happened. These events cannot be modified and always refer to an event that took place. Modeling the absence of a thing or a thing that didn't actually occur may often seem like a good idea, but it can confuse both developers and event processors.

While there will be other blocks similar to this, I've made sure that the ones with the exclamation point are the "laws" of event sourcing, which are also contained in the appendix.

Code Blocks

Code blocks throughout the book generally take the same form, as shown below:

```
defmodule Calculator do
  def add(x, y) do
    x + y
  end

  def mul(x, y) do
    x * y
  end

  def div(x, y) do
    x / y
  end

  def sub(x, y) do
```

```
    x - y
  end
end
```

For the most part, the code is formatted the way you might expect Elixir modules to be formatted. However, in some cases the standard Elixir format produces listings that are too long due to excessive carriage returns, too wide for a printed page, etc. In those cases, we've made our own book-specific formatting choices and this isn't an oversight but intentional.

Sidebars and Commentary

You've already seen the blocks of text that provide additional context about the part of the chapter you're reading. You may also occasionally see a sidebar like the one below. Sidebars offer parenthetical or related information that might not be considered part of the book's main core.

Dungeons and Dragons

I mentioned a random number generator in the code, so as an aside, I think I should discuss dice-based games like Dungeons and Dragons.

Online Resources

As always, the single source of truth for this book and its associated code is the Pragmatic Programmers website.[5] Throughout the book there will be links to GitHub repositories, websites, and other reference materials that complement the book's core text.

Let's Get Started!

Now the fun begins and we can start our event sourcing journey!

5. https://pragprog.com/titles/khpes