Extracted from:

Programming Phoenix ≥ 1.4

Productive |> Reliable |> Fast

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Your Elixir Source

Programming Phoenix ≥ 1.4

Productive |> Reliable |> Fast

Chris McCord,
Bruce Tate,
and José Valim

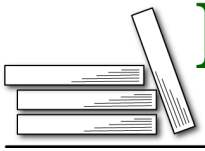
edited by Jacquelyn Carter

Programming Phoenix ≥ 1.4

Productive |> Reliable |> Fast

Chris McCord
Bruce Tate
José Valim

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
Managing Editor: Susan Conant
Development Editor: Jacquelyn Carter
Copy Editor: Jasmine Kwityn
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: 978-1-68050-226-8
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—October 2019

Preface

It doesn't seem possible, but it's been three years since we released the first edition of this book. Indeed, the Phoenix team has been busy. The additions of Channel Presence and LiveView are changing the way all programmers think about web development. The underlying directories have changed, having a rippling impact on all of the code in this book. Ecto has also produced a major release.

Through it all, one thing remains constant. Phoenix is still positioned as one of the most productive and scalable web development platforms available anywhere. From cryptocurrencies to media companies to commerce, Elixir developers are using Phoenix to push the boundaries of what's possible. In this book, the same folks who built Elixir and Phoenix will show you how you can do the same.

Is This Book for You?

If you've followed Phoenix for any period of time, you already know that this book is the definitive resource for Phoenix programming. If you're using Phoenix or are seriously considering doing professional Elixir development, you're going to want this book. It's packed with insights from the team that created it. Find just one tip in these pages, and the book will pay for itself many times over. This section seeks to answer a different question, though. Beyond folks who've already decided to make an investment in Phoenix, who should buy this book?

Programmers Embracing the Functional Paradigm

Every twenty years or so, new programming paradigms emerge. The industry is currently in the midst of a shift from object-oriented programming to functional programming. If you've noticed this trend, you know that a half dozen or so functional languages are competing for mindshare. The best way to understand a programming language is to go beyond basic online tutorials to see how to approach nontrivial programs.

With *Programming Phoenix*, we don't shy away from difficult problems such as customizing authentication, designing for scale, or creating interactive web pages. As you explore the language, you'll learn how the pieces fit together to solve difficult problems and how functional programming helps us do it elegantly. When you're done, you might not choose Phoenix, but you'll at least understand the critical pieces that make it popular and if those pieces are likely to work for you.

Developers Seeking to Modernize

Developers from many web frameworks written in many languages can find something here. Phoenix measures response times in microseconds, and it has been shown to handle millions of concurrent WebSocket connections on a single machine without sacrificing the productivity we've come to appreciate. If you're pushing your favorite framework to be more scalable or more interactive, you're not alone. You're going to find Phoenix powerful and interesting. And if you are trying to build single page apps or provide a more consistent or interactive experience for your user, you'll find Elixir one of the best available languages for solving that problem, period.

Dynamic Programmers Looking for a Mature Environment

Like the authors of this book, you may be a fan of dynamic languages like JavaScript, Python, and Ruby. You may have used them in production or even contributed to those ecosystems. Many developers like us are looking for similar flexibility but with a more robust runtime experience. We may love the programming experience in those languages, but we often find ourselves worn out by the many compromises we have to make for performance, concurrency, and maintainability. Phoenix resonates with us because many of the creators of this ecosystem built it to solve these problems.

Elixir is a modern dynamic language built on the three-decades-old, battle-tested Erlang runtime. Elixir macros bring a lot of the flexibility that Ruby, Python, and JavaScript developers came to love, but those dynamic features are quarantined to compile time. With Elixir, during runtime, you have a consistent system with great type support that's generally unseen in other dynamic languages.

Mix these features with the concurrency power, and you'll see why Phoenix provides such excellent performance for everything on the web, and beyond.

Java Developers Seeking More

When Java emerged twenty years ago, it had everything a frustrated C++ community was missing. It was object-oriented, secure, ready for the Internet, and simple, especially when compared to other C++ alternatives at the time. As the Java community flourished and consolidated, the tools and support came. Just about everyone supported Java, and that ubiquity led to a language dominance that we'd never seen before.

As Java has aged, it's lost some of that luster. As the committees that shaped Java compromised, Java lost some of the edge and leadership that the small leadership team provided in early versions. Backward compatibility means that the language evolves slowly as new solutions emerge. You might find that all of that early ubiquity has led to an experience that's more fragmented or bloated than you like it. You may enjoy the extra punch of emerging languages like Elixir. The Java concurrency story *does* place plenty of burden on the developer, leaving libraries that may or may not be safe for production systems to cope with increasingly parallel designs.

If you're a Java developer looking for where to go next, or a JVM-language developer looking for a better concurrency story, Phoenix would mean leaving the JVM behind. Maybe that's a good thing. You'll find a unified, integrated story in Phoenix with sound abstractions on top. The choice is up to you.

Erlang Developers Doing Integrated Web Development

As time goes on, the number of Erlang developers who also gain proficiency in Elixir is growing. The toolchain for Phoenix is spectacular, and many of the tools that exist for Erlang can work in this ecosystem as well. If you're an Erlang developer, you may want to take advantage of Mix's excellent scripting for the development, build, and testing workflow. You may like the package management in Hex, or the neat composition of concerns in the Plug library. You may want to use macros to extend the language for your business, or test with greater leverage. You'll have new programming features like protocols or structs.

If you do decide to embrace Elixir, that doesn't mean you need to leave Erlang behind. You'll still be able to use the Erlang libraries you enjoy today, including the Erlang process model and full OTP integration. You'll be able to access your OTP 'GenServer's directly from the Elixir environment, and directly call libraries without the need for extra complex syntax. If these terms aren't familiar to you, don't worry. We'll explore each of them over the course of the book.

Heat Seekers

As web demands grow, an increasing number of developers require infrastructure that will serve more users reliably. If you need raw power supported by a rich language, we have a solution and the numbers to back it up. You'll have to work for it, but you'll get much better speed and reliability when you're done. We've run a single chat room on one box supporting two million users. That means that each new message had to go out two million times. Phoenix performs well out of the box and our numbers improve as more cores are added. If you need speed, we have the tonic for what ails you.

Others

Certainly, this book isn't for everyone. We do think that if you're in one of these groups, you'll find something you like here. We're equally confident that folks that we haven't described will pick up this book and find something valuable. If you're one of those types, let us know your story.

About this Book

This book is about building web applications with the primary web framework for the Elixir language, Phoenix. In its pages we will walk you through building a web application, piece by piece.

In Part I, we will show you how to build a traditional model-view-controller (MVC) application. We'll guide you through the Phoenix landscape, showing you in intimate detail how things are stitched together. We will show you how to build a controller and how to organize your business logic into modules called contexts. Along the way, we'll build our own authentication and build database-backed code with a database library called Ecto.

In Part II, we will explore channels and presence, Phoenix features that allow a highly interactive experience. Then we'll learn to tie those interactive features into Elixir's extensive OTP, a framework for building concurrent, self-healing projects. We will focus on techniques for productively writing code that will be easier to maintain in the future.

To illustrate both parts of this book fully, we will build a web application together. The application will let users take videos and annotate them with real-time events.

Online Resources

The apps and examples shown in this book can be found at the Pragmatic Programmers website for this book.[1] You'll also find the errata-submission form, where you can report problems with the text or make suggestions for future versions.

When you're ready, turn the page and we'll get started. Let's build something together!

1. http://pragprog.com/book/phoenix14/