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

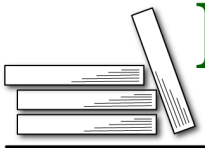
edited by Jacquelyn Carter

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*.

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

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

Printed in the United States of America.
ISBN-13: 978-1-68050-226-8
Encoded using the finest acid-free high-entropy binary digits.
Book version: B1.0—May 23, 2018

Preface

Welcome, beta readers! This second edition of Programming Phoenix has been a long time coming. Thank you for your patience. We needed to think through some structural application changes to get here, but those changes are now behind us. We'd like to point out a few things that are happening before you roll up your sleeves and get to work.

- If you're coming from the earlier versions of this book, you'll notice that the structure of all of the examples changed. We have an additional layer throughout, called the context. We believe these changes will improve your code and the way you think about building functional programs.
- You may also notice that the directory structure has changed. Instead of separating web code into `web` and all other code into `lib`, we're now including all code in `lib` to conform to the rest of the Elixir community and the mix tool.
- Ecto has a new major version. This version changes the API significantly, especially around managing relationships in your changesets.

With the Phoenix changes mostly behind us, we can now focus on this book. We have a big request for our readers. We want your feedback. Your feedback is critical for helping us shape this book. Please help us!

Thank you so much for your readership. Books need readers and beta books need them even more. We'll try our hardest to reward your good faith. Without further ado, let's get to it!

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 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.

Rails Developers Seeking Solutions

If you follow the Rails community closely, you know that it has experienced some attrition. Bear in mind that this team believes that Ruby on Rails was great for our industry. Rails still solves some problems well, and for those problems it can be a productive solution. The problem for Rails developers is that the scope of problems it's best able to solve is rapidly narrowing.

In fact, the early growth of Elixir is partially fueled by Rails developers like you. The similar syntax provided an attractive base for learning the language, but the radically improved programming paradigms, introspectable runtime, and concurrency models all provide the solid foundation that those who push Rails the hardest find lacking.

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 Rails to be more scalable or more interactive, you're not alone. You're going to find Phoenix powerful and interesting.

Dynamic Programmers Looking for a Mature Environment

Like the authors of this book, you may be a fan of dynamic languages like Ruby, Python, and JavaScript. 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 the 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. All of that early ubiquity has led to a fragmented and bloated ecosystem that moves too slowly and takes years to master, but delivers a fraction of the punch of emerging languages. The Java concurrency story places too much of a burden on the developer, leaving libraries that may or may not be safe for production systems.

New languages are emerging on the JVM, and some of those are rich in terms of features and programming models. This team respects those languages tremendously, but we didn't find the same connection there that we found

elsewhere. We also had a hard time separating the good from the bad in the Java ecosystem.

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. You'll see a syntax that provides Clojure-style metaprogramming on syntax that we think is richer and cleaner than Scala's. You'll find an existing ecosystem from the Erlang community that has a wide range of preexisting libraries, but ones built from the ground up to support not only concurrency, but also distributed software.

Erlang Developers Doing Integrated Web Development

Curiously, we're only beginning to see a heavy proliferation of Erlang developers in the Elixir community. We expect that to accelerate. 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 `GenServers` 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

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. We've run benchmarks among the best in the industry, and our numbers seem to be improving 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.

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.

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