Extracted from:

Property-Based Testing with PropEr, Erlang, and Elixir

Find Bugs Before Your Users Do

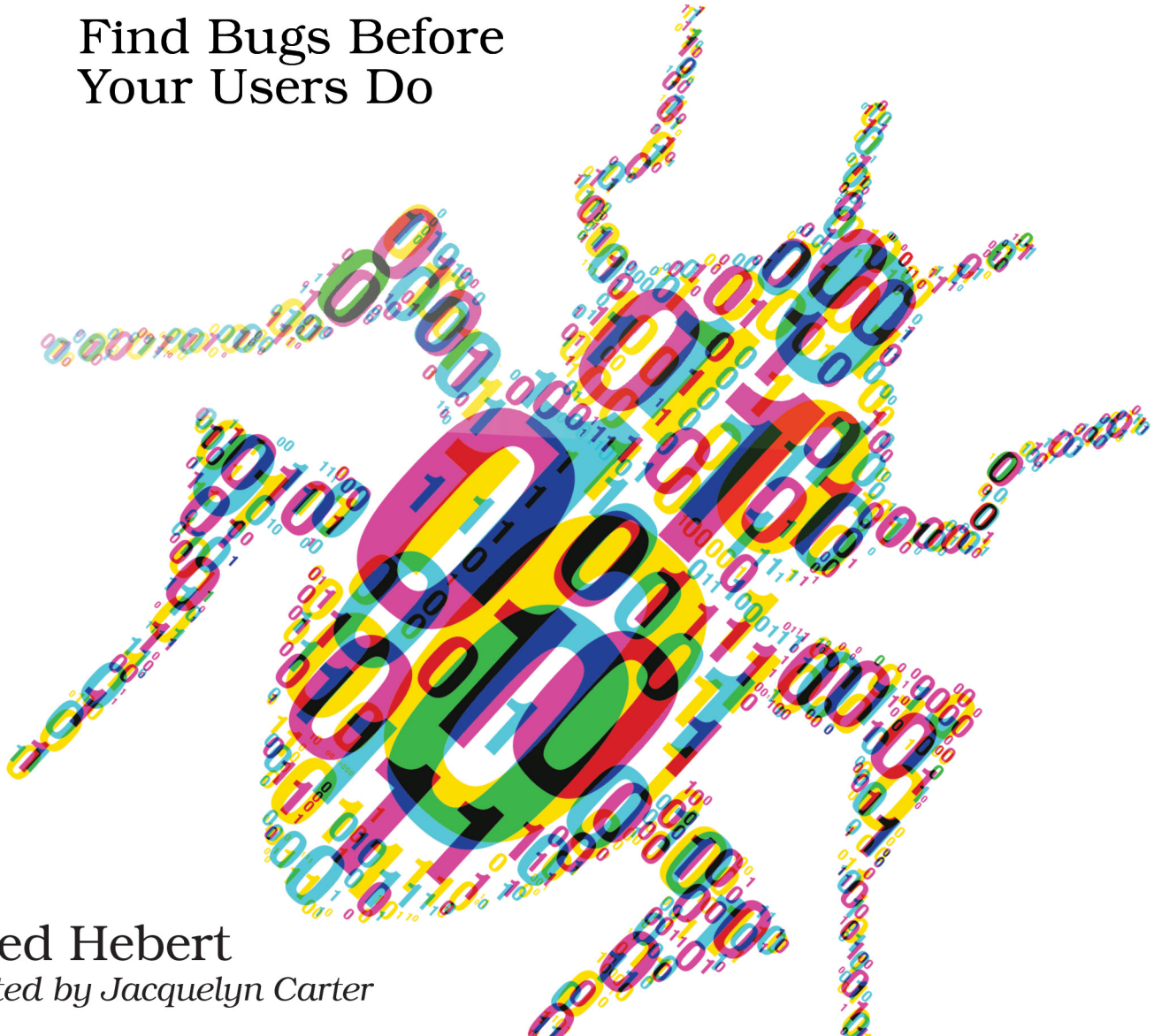
The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Property-Based Testing with PropEr, Erlang, and Elixir

Find Bugs Before Your Users Do



Fred Hebert
edited by Jacquelyn Carter

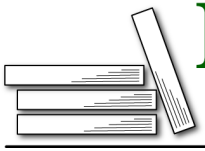
Property-Based Testing with PropEr, Erlang, and Elixir

Find Bugs Before Your Users Do

Fred Hebert

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: L. Sakhi MacMillan
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: 978-1-68050-621-1
Book version: P1.0—January 2019

Acknowledgments

I want to thank everyone who offered their time to review this book: Sean Cribbs, Chris Keathley (who also has been a huge cheerleader of this work on the Elixir Outlaws podcast), Gabrielle Denhez, Jamu Kakar, Maurice Kelly, Xavier Shay, Ben Wilson, Evan Vigil-McClanahan, Ben Marx, Bruce Williams, Kim Shrier, and folks who got the beta versions of this book and then sent feedback or filed errata, or both.

There's also an obvious list of people to thank at PragProg, including Andy, Bruce Tate, and more particularly Jackie Carter, who edited this book and without whom it wouldn't be nearly as readable as it is.

I need to thank Jenn, who tolerated me writing this on weeknights on and off for over a year. I should finally thank Drew Fradette, who suggested I pitch my early draft to PragProg, which started the whole process of turning it into this book.

Introduction

When I finished my first book, *Learn You Some Erlang*, I told myself "never again." There's something distressing about spending months and years of work writing a book, spending all bits of free time you can find on it, putting aside other projects and hobbies, and rewriting your own texts close to a dozen times. You reach the point where before you're even done, you're tired of writing about the topic you chose to write about.

I knew all of that was waiting for me if I ever wanted to write another book. I decided to do it anyway because I truly believe property-based testing is something amazing, worth learning and using. In fact, part of the reason why I wanted to write a book was that I wanted to use property-based testing in projects at work and online, and it's generally a bad idea to introduce a technology when only one person in the team knows how it works.

It's a better compromise to spend all that time and effort writing a book than never using property-based testing in a team when I know what it can do and bring to a project. Hopefully, you'll feel that learning about it here is worth your time as well.

Who Is This Book For

If you know enough of Erlang or Elixir to feel comfortable writing a small project, you're fit for this book. There're a few things that might be a bit confusing, but you should be able to pull through.

If you have experience with unit testing and TDD, you'll feel comfortable with most of this book's testing concepts. While the text does not advocate using TDD or not (we avoid this whole debate), techniques that use properties to help design your programs are still shown and constitute a valuable option to explore a new problem space.

If you are, like me, one of these grumpy people who are annoyed with the quality of software and feel that you can't trust yourself to deliver high-quality

code every time—you know your code will come back to haunt you sooner or later—then you will probably consider property-based testing a godsend.

Why Both Elixir and Erlang

The Erlang and Elixir communities possibly suffer from a kind of *narcissism of minor differences*; a kind of hostility exists based on small differences between the two languages and how developers do things, despite Elixir and Erlang being so much closer to each other than any other language or platform out there.

This book represents a conscious effort to bridge the gap between the two communities and see both groups join strengths rather than compete with each other; it is one small part, attempting to use one property-based testing tool, with one resource, to improve the code and tests of one community.

What's in This Book

This book covers pretty much everything you need to get started, up to the point where you feel confident enough to use the most advanced features of PropEr. We'll start smoothly, with the basic and foundational principles of property-based testing, see what the framework offers us to get started, make our way through thinking in properties, write our own custom data generators, and then really start going wild. You'll see how property-based testing can be used in a realistic project (and where it should not be used) and learn various techniques to make the best use of it possible to get the most value out of it. We'll also cover properties to test more complex stateful systems, a practice that is useful for integration and system tests.

Those are the topics covered, but more than anything, you may get a set of strategies to think about new approaches to test your software. Rather than just writing repetitive examples for tests, or just generating random data to throw at the code, you'll learn new techniques to find new bugs you never thought could be hiding in your code. You'll also gain tools to reason about how to build software, how to explore the problem space, and how to evaluate the fitness of the solutions you choose.

How to Read This Book

You should feel comfortable just reading all chapters in order. The first part of the book contains truly essential material to get your fundamentals right and get started properly. The second part applies properties in more realistic scenarios to gain comfort, and the last part covers stateful tests.

But really, it's easiest to read things in order. Some chapters have questions and exercises at the end. You can skip these if you want, but going through them will be a good way to reinforce your understanding of the material covered.

The exercises are added particularly when a lot of theory is introduced in the chapter and when the material will come back again and again in following chapters. Going through them may sometimes be tricky, but they will make the following chapters easier to go through. And because exercises left for the reader with no guidance are annoying as hell, all solutions are provided.

About the Code

Code is provided in both languages in most places where it makes sense to do so. Code samples may look like the following:

Erlang code/path/to/file.erl

```
%% This is some random code for demonstration purposes
path(_Current, Acc, _Seen, [_,_,_,_]) ->
    Acc;
path(Current, Acc, Seen, Ignore) ->
    frequency([
        {1, Acc}, % probabilistic stop
        {15, increase_path(Current, Acc, Seen, Ignore)}
    ]).
```

Elixir code/path/to/file.ex

```
# This is some random code for demonstration purposes
def path(_current, acc, _seen, [_,_,_,_]) do
  acc
end
def path(current, acc, seen, ignore) do
  frequency([
    {1, acc}, # probabilistic stop
    {15, increase_path(current, acc, seen, ignore)}
  ])
end
```

Code references such as ❶ will be used to point to locations in both languages at once.

Exceptions to this norm will include code that should be treated as pseudo-code, shell session output (which will be in Erlang only), and longer code samples that would take a lot of space, which will instead be located within an appendix to ease the reading flow. Otherwise, things should be quite readable for both languages at once.

When mentioned, file names for a code snippet point to where you should put the code if you're following along. Frequent reminders about this will be added, just in case. Downloadable code for this book contains the final code for each module and may not contain intermediary steps shown in the text.

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 a link there where you can provide feedback by submitting errata entries.

The book relies on the PropEr[2] library. Its online documentation[3] will invariably prove useful.

Elixir users will use the PropCheck[4] wrapper library, which also has its own online documentation.[5]

Fred Hebert

January 2019

1. https://pragprog.com/book/fhproper/property-based-testing-with-proper-erlang-and-elixir
2. https://github.com/proper-testing/proper
3. https://proper-testing.github.io/
4. https://github.com/alfert/propcheck
5. https://hexdocs.pm/propcheck