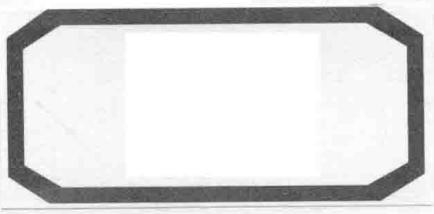
O'REILLY®

Designing for Scalability with Erlang/OTP

Erlang/OTP可扩展性设计指南（影印版）

Francesco Cesarini,
Steve Vinoski 著

東南大學出版社



Erlang/OTP可扩展性设计指南（影印版）

Designing for Scalability with Erlang/OTP

Francesco Cesarini,
Steve Vinoski 著

l · Tokyo O'REILLY®
a, Inc.授权东南大学出版社出版

南京 东南大学出版社

图书在版编目(CIP)数据

Erlang/OTP 可扩展性设计指南：英文/(英)弗朗西斯科·切萨里尼(Francesco Cesarini)，(美)史蒂夫·温斯基(Steve Vinoski)著. —影印本. —南京：东南大学出版社，2017.1

书名原文：Designing for Scalability with Erlang/OTP

ISBN 978-7-5641-6902-2

Ⅰ.①E… Ⅱ.①弗…②史… Ⅲ.①程序语言—程序设计—指南—英文 Ⅳ.①TP312-62

中国版本图书馆 CIP 数据核字(2016)第 318007 号

图字：10-2014-151 号

© 2016 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press, 2017. Authorized reprint of the original English edition, 2014 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc.出版 2016。

英文影印版由东南大学出版社出版 2017。此影印版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc.的许可。

版权所有，未得书面许可，本书的任何部分和全部不得以任何形式重制。

Erlang/OTP 可扩展性设计指南(影印版)

出版发行：东南大学出版社

地　　址：南京四牌楼 2 号　　邮编：210096

出 版 人：江建中

网　　址：http://www.seupress.com

电子邮件：press@seupress.com

印　　刷：常州市武进第三印刷有限公司

开　　本：787 毫米×980 毫米　16 开本

印　　张：30.25

字　　数：592 千字

版　　次：2017 年 1 月第 1 版

印　　次：2017 年 1 月第 1 次印刷

书　　号：ISBN 978-7-5641-6902-2

定　　价：94.00 元

本社图书若有印装质量问题，请直接与营销部联系。电话(传真)：025-83791830

To Alison, Peter and baby Bump for being patient and supportive.

—Francesco

To Dooley and Ed, for teaching me how, and to Cindy, Ryan, Erin, Andrew, and Jake, for being why.

—Steve

To Joe, Mike, Robert, for making that phone call.

—Francesco & Steve

Preface

This book is what you get if you put together an Erlang enthusiast who worked on the R1 release of OTP in 1996 and a Distributed Systems specialist who discovered a decade later how Erlang/OTP allows you to focus on the real challenges of systems development while avoiding accidental difficulties.

By describing how OTP behaviors are built and why they are needed, we show you how to use them to architect standalone nodes. In our original proposal to O'Reilly, we stopped here. But when writing the book, we decided to push the bar further, documenting our practices, design decisions, and common pitfalls when architecting a distributed system. These patterns, through a set of design choices and tradeoffs we make, give us the scalability, reliability, and availability for which Erlang/OTP is well known. Contrary to popular belief, this does not happen magically or out of the box, but it sure is much easier to achieve than with any other programming language out there that does not emulate Erlang's semantics nor run on the BEAM virtual machine.

Francesco: Why This Book?

Someone once told me that writing books is a bit like having children. Once you've written one and are holding your paper copy, excitement takes over, you quickly forget the hard work and sacrifices, and you want to start writing another one. I've been intending to write the sequel to *Erlang Programming* (O'Reilly) since first holding the paper copy in June 2009. I had no children of my own when I started this project, but it ended up taking so long that my second one is now on its way. Whoever said that good things are not worth waiting for?

As with the first book, we based *Designing for Scalability with Erlang/OTP* on the examples in the Erlang Solution's OTP training material I developed. I used the examples and started explaining them, converting my lectures and approach to teaching into words. When done with a chapter, I went back and ensured the parts students struggled to understand were clear. Questions that were commonly asked by

此为试读，需要完整PDF请访问：www.ertongbook.com

the best students ended up in sidebars, and long chapters were divided into smaller ones. It all went well until we reached Chapter 11 and 12, because there was no unified way of doing release handling or software upgrade. Rather, there were tools, many of them. Some were integrated in our client's build and release cycle, others worked out of the box. Some were unusable. The chapters are what we hope will become the ultimate guide to anyone wanting to understand how release handling and software upgrade of systems works behind the scenes. They also explain what you need to know should you have to troubleshoot existing tools or write your own.

But the real trouble started with Chapter 13. Not having examples or training material, I found myself formalizing what was in our heads and documenting the approaches we take when architecting Erlang/OTP systems, trying to align it with the theory of distributed computing. Chapter 13 turned into four chapters that took as long to write as the first ten. For those of you who bought the early access, I hope the wait was worth it. For those who wisely waited for us to finish before buying your copy, enjoy!

Steve: Why This Book?

I first discovered Erlang/OTP in 2006 while researching ways to develop enterprise integration software faster, cheaper, and better. No matter how I looked at it, Erlang/OTP was clearly superior to the C++ and Java languages my colleagues and I had long been using at that time. In 2007 I joined a new company and began using Erlang/OTP for a commercial product, and it turned out to be everything my earlier investigation promised it would be. I taught the language to some colleagues and before long, fewer than a handful of us were developing software that was more capable, more reliable, easier to evolve, and ready for production far faster than similar code being written by a significantly larger team of C++ programmers. To this day I remain wholly convinced of the impressive practical effectiveness of Erlang/OTP.

Over the years I've published quite a bit of technical material, and my intended audience for all of it has always been other practitioners like me. This book is no exception. In the first 12 chapters we provide the deep level of detail that practicing developers need in order to fully understand the fundamental design principles of OTP. With those details we mix a number of useful nuggets of practical knowledge—modules, functions, and approaches that will save you significant time and effort in your day-to-day design, development, and debugging efforts. In the final four chapters we shift gears, focusing more on the big picture of the tradeoffs involved in developing, deploying, and operating resilient, scalable distributed applications. Due to the staggering amount of knowledge, approaches, and tradeoffs involved in distributed systems, fault tolerance, and DevOps, writing these chapters concisely proved difficult, but I believe we hit just the right balance of providing plenty of great advice without getting lost in the weeds.

I hope this book helps you improve the quality and utility of the software and systems you develop.

Who Should Read This Book

This book's intended audience includes Erlang and Elixir developers and architects who have made their way through at least one of the introductory books and are ready to take their knowledge to the next level. It is not a book to start off with, but rather the book that picks up where all others leave you. Chapters 3–12 build on each other and should be read sequentially, as do Chapters 13–16. If you do not need an Erlang primer, feel free to skip Chapter 2.

How To Read This Book

We wrote this book to be compatible with Erlang Release 18.2. Most of the features we describe work with earlier releases; major features that don't are indicated in the book. Currently unknown incompatibilities with future releases will be detailed on our errata page and fixed in the book's github repository. You are encouraged to download the examples in the book from our github repository and run them yourself to better understand them.

Acknowledgments

Writing this book has been a long journey. While undertaking it we've had a lot of great help from a lot of wonderful people. Our editor Andy Oram has been an endless source of ideas and suggestions, patiently guiding us, giving us feedback while providing ongoing encouragement. Thank you Andy, we couldn't have done it without you! Simon Thompson, coauthor of *Erlang Programming* helped with the book proposal and laid the foundation for the second chapter. Many thanks to Robert Virding for contributing some of the examples. We've had many readers, reviewers and contributors give us feedback as we drip-fed them the chapters. At the risk of forgetting someone, they are: are Richard Ben Aleya, Roberto Aloi, Jesper Louis Andersen, Bob Balance, Eva Bihari, Martin Bodocky, Natalia Chechina, Jean-François Cloutier, Richard Croucher, Viktória Fördős, Heinz Gies, Joacim Halén, Fred Hebert, Csaba Hoch, Torben Hoffmann, Bob Ippolito, Aman Kohli, Jan Willem Luiten, Jay Nelson, Robby Raschke, Andrzej Śliwa, David Smith, Sam Tavakoli, Premanand Thangamani, Jan Uhlig, John Warwick, David Welton, Ulf Wiger, and Alexander Yong. If we missed you, our sincere apologies! Drop us an email and you will be promptly added. A shout-out goes to the staff at Erlang Solutions for reading the chapters as they were being written and everyone else who submitted to the errata as part of the early release. A special thank you goes to all of you who cheered us on through social media channels, especially other authors. You know who you are! Last,

but not least, thanks to the production, marketing, and conference teams at O'Reilly who kept on reminding us that it's not over until you are holding the paper copy. We really appreciate your support!

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, applications, URLs, email addresses, filenames, directory names, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords. Also used for behaviors, commands, and command-line options.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This icon signifies a tip or suggestion.

This icon signifies a general note.

This icon indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at: *https://github.com/francescoc/scalabilitywitherlangotp*

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Designing for Scalability with Erlang/OTP* by Francesco Cesarini and Steve Vinoski (O'Reilly). Copyright 2016 Francesco Cesarini and Stephen Vinoski, 978-1-449-32073-7."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://bit.ly/designing-for-scalability-with-erlangotp

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Table of Contents

Preface xiii

1. Introduction 1
Defining the Problem 2
OTP 4
Erlang 6
Tools and Libraries 7
System Design Principles 10
Erlang Nodes 11
Distribution, Infrastructure, and Multicore 12
Summing Up 13
What You'll Learn in This Book 14

2. Introducing Erlang 21
Recursion and Pattern Matching 21
Functional Influence 25
Fun with Anonymous Functions 25
List Comprehensions: Generate and Test 27
Processes and Message Passing 29
Fail Safe! 33
Links and Monitors for Supervision 34
Links 35
Monitors 37
Records 38
Maps 41
Macros 42
Upgrading Modules 43
ETS: Erlang Term Storage 45

此为试读，需要完整PDF请访问：www.ertongbook.com

Distributed Erlang 48
Naming and Communication 48
Node Connections and Visibility 49
Summing Up 51
What's Next? 51

3. Behaviors..... 53
Process Skeletons 53
Design Patterns 56
Callback Modules 57
Extracting Generic Behaviors 60
Starting the Server 62
The Client Functions 64
The Server Loop 66
Functions Internal to the Server 68
The Generic Server 69
Message Passing: Under the Hood 72
Summing Up 76
What's Next? 76

4. Generic Servers..... 77
Generic Servers 77
Behavior Directives 78
Starting a Server 80
Message Passing 82
Synchronous Message Passing 82
Asynchronous Message Passing 84
Other Messages 85
Unhandled Messages 86
Synchronizing Clients 88
Termination 89
Call Timeouts 91
Deadlocks 94
Generic Server Timeouts 95
Hibernating Behaviors 97
Going Global 97
Linking Behaviors 99
Summing Up 99
What's Next? 100

5. Controlling OTP Behaviors..... 101
The sys Module 101

Tracing and Logging 101
System Messages 103
Your Own Trace Functions 103
Statistics, Status, and State 105
The sys Module Recap 107
Spawn Options 108
Memory Management and Garbage Collection 109
Spawn Options to Avoid 113
Timeouts 114
Summing Up 114
What's Next? 114

6. Finite State Machines. 117
Finite State Machines the Erlang Way 118
Coffee FSM 119
The Hardware Stub 121
The Erlang Coffee Machine 122
Generic FSMs 125
A Behavior Example 127
Starting the FSM 127
Sending Events 131
Termination 139
Summing Up 141
Get Your Hands Dirty 141
The Phone Controllers 141
Let's Test It 143
What's Next? 145

7. Event Handlers. 147
Events 147
Generic Event Managers and Handlers 149
Starting and Stopping Event Managers 149
Adding Event Handlers 150
Deleting an Event Handler 152
Sending Synchronous and Asynchronous Events 153
Retrieving Data 156
Handling Errors and Invalid Return Values 158
Swapping Event Handlers 160
Wrapping It All Up 162
The SASL Alarm Handler 165
Summing Up 166
What's Next? 167

8. Supervisors..... 169
Supervision Trees 170
OTP Supervisors 174
The Supervisor Behavior 175
Starting the Supervisor 176
The Supervisor Specification 179
Dynamic Children 186
Non-OTP-Compliant Processes 194
Scalability and Short-Lived Processes 195
Synchronous Starts for Determinism 197
Testing Your Supervision Strategy 199
How Does This Compare? 200
Summing Up 201
What's Next? 201

9. Applications..... 203
How Applications Run 204
The Application Structure 206
The Callback Module 209
Starting and Stopping Applications 210
Application Resource Files 213
The Base Station Controller Application File 215
Starting an Application 216
Environment Variables 219
Application Types and Termination Strategies 221
Distributed Applications 222
Start Phases 226
Included Applications 228
Start Phases in Included Applications 228
Combining Supervisors and Applications 230
The SASL Application 231
Progress Reports 236
Error Reports 236
Crash Reports 237
Supervisor Reports 238
Summing Up 239
What's Next? 240

10. Special Processes and Your Own Behaviors..... 241
Special Processes 242
The Mutex 242
Starting Special Processes 244

The Mutex States 247
Handling Exits 247
System Messages 249
Trace and Log Events 250
Putting It Together 251
Dynamic Modules and Hibernating 255
Your Own Behaviors 255
Rules for Creating Behaviors 256
An Example Handling TCP Streams 256
Summing Up 260
What's Next? 261

11. System Principles and Release Handling..... 263
System Principles 264
Release Directory Structure 265
Release Resource Files 269
Creating a Release 273
Creating the Boot File 274
Creating a Release Package 283
Start Scripts and Configuring on the Target 287
Arguments and Flags 290
The init Module 302
Rebar3 303
Generating a Rebar3 Release Project 304
Creating a Release with Rebar3 308
Rebar3 Releases with Project Dependencies 310
Wrapping Up 312
What's Next? 316

12. Release Upgrades..... 317
Software Upgrades 318
The First Version of the Coffee FSM 320
Adding a State 323
Creating a Release Upgrade 326
The Code to Upgrade 330
Application Upgrade Files 333
High-Level Instructions 337
Release Upgrade Files 339
Low-Level Instructions 342
Installing an Upgrade 343
The Release Handler 346
Upgrading Environment Variables 350

Upgrading Special Processes 350
Upgrading in Distributed Environments 351
Upgrading the Emulator and Core Applications 352
Upgrades with Rebar3 353
Summing Up 356
What's Next? 358

13. Distributed Architectures. 359
Node Types and Families 360
Networking 363
Distributed Erlang 366
Sockets and SSL 373
Service Orientation and Microservices 375
Peer to Peer 376
Interfaces 377
Summing Up 380
What's Next? 381

14. Systems That Never Stop. 383
Availability 383
Fault Tolerance 384
Resilience 385
Reliability 387
Sharing Data 392
Tradeoffs Between Consistency and Availability 400
Summing Up 401
What's Next? 403

15. Scaling Out. 405
Horizontal and Vertical Scaling 405
Capacity Planning 409
Capacity Testing 412
Balancing Your System 414
Finding Bottlenecks 416
System Blueprints 419
Load Regulation and Backpressure 419
Summing Up 422
What's Next? 424

16. Monitoring and Preemptive Support. 425
Monitoring 426
Logs 428

Metrics 433
Alarms 436
Preemptive Support 439
Summing Up 441
What's Next? 443

Index **445**

此为试读,需要完整PDF请访问: www.ertongbook.com