

The
Pragmatic
Programmers

Scala
2.11

Pragmatic Scala

Create Expressive,
Concise, and
Scalable
Applications

Venkat Subramaniam

edited by Jacquelyn Carter



easy ●●●
computing

easy ●●●
computing

Early praise for *Pragmatic Scala*

A well-paced, easy-to-read, and practical guide to Scala for Java programmers. Covers important aspects of this powerful multi-paradigm language, ensuring that you can become productive with Scala right away.

► **Ramnivas Laddad**

Author of *AspectJ in Action*, speaker, and consultant

In *Pragmatic Scala*, Venkat provides a solid foundation to help you get started with Scala in a thorough yet succinct book that you can (and should) read cover to cover. He explores the most important topics you'll need to get comfortable with Scala, starting with the REPL and progressing through functional programming in Scala, handling concurrency with actors, and Java interoperability. You will definitely want to fire up your editor or IDE and code along with the book's numerous, interesting, and fun examples!

► **Scott Leberknight**

Software architect, Fortitude Technologies

In a world of tweets, blogs, and bite-sized videos, there is still a place for the long-form narrative. It gives a teacher the time necessary to introduce challenging and complex topics. And, if we're being honest, Scala is a challenging and complex topic. Take this opportunity for Venkat to guide you through the Scala language, functional programming, concurrency, testing strategies, and more.

► **Brian Sletten**

President, Bosatsu Consulting, Inc.

There are two reasons why I recommend this book for all my Scala classes. It covers the basics without insulting your intelligence and escalates you to advanced topics without going over your head. A must-have for anyone learning Scala.

► **Daniel Hinojosa**

Programmer, instructor, presenter, and author of *Testing in Scala*

The thing I like most about Venkat is the way he can introduce complex concepts or unknown topics through a conversational style that starts with something that the reader is familiar with and then builds upon it. This latest version of *Pragmatic Scala* is no exception. I highly recommend it to anyone looking to learn Scala, especially those coming from a Java background.

► **Ian Roughley**

Engineering director, nToggle

Pragmatic Scala

Create Expressive, Concise, and Scalable Applications

Venkat Subramaniam

easy 
computing The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at <https://pragprog.com>.

The team that produced this book includes:

Jacquelyn Carter (editor)
Potomac Indexing, LLC (index)
Liz Welch (copyedit)
Dave Thomas (layout)
Janet Furlow (producer)
Ellie Callahan (support)

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

Copyright © 2015 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

Printed in the United States of America.

ISBN-13: 978-1-68050-054-7

Encoded using the finest acid-free high-entropy binary digits.

Book version: P1.0—September 2015

easy 
computing

Contents

Acknowledgments	ix
Introduction	xi

Part I — Getting Your Feet Wet

1.	Exploring Scala	3
	Scala Features	3
	More with Less	4
	Functional Programming	8
2.	Taking Scala for a Ride	13
	Using the REPL	13
	Scala on the Command Line	15
	Running Scala Code as a Stand-alone Script	16
	Compiling Scala	18
3.	From Java to Scala	21
	Scala as Concise Java	21
	Scala Classes for Java Primitives	24
	Tuples and Multiple Assignments	25
	Flexible Parameters and Arguments	27
	Implicit Parameters	31
	Strings and Multiline Raw Strings	33
	String Interpolation	34
	Sensible Defaults	36
	Operator Overloading	38
	Scala Surprises for the Java Eyes	40
	Default Access Modifier	44

4.	Working with Objects	49
	Creating and Using Classes	49
	Following the JavaBean Convention	54
	Type Aliasing	56
	Extending a Class	56
	Parameterized Types	58
	Singletons and Companions	60
	Creating Enumerations	65
	Package Objects	67
5.	Making Use of Types	71
	Type Inference	71
	Fundamental Types	76
	Return Type Inference	80
	Variance of Parameterized Type	82
	Implicit Type Conversions	86
	Value Classes	90
	Using Implicit Conversions	94

Part II — Diving Into Scala

6.	Function Values and Closures	101
	Limitations of Normal Functions	101
	Extensibility with Higher-Order Functions	102
	Function Values with Multiple Parameters	104
	Currying	107
	Positional Notation for Parameters	108
	Parameter Routing	109
	Reusing Function Values	110
	Partially Applied Functions	112
	Closures	114
	Execute Around Method Pattern	115
7.	Traits	119
	Understanding Traits	119
	Selective Mixins	122
	Decorating with Traits	123
	Method Late Binding in Traits	125

8.	Collections	129
	Common Scala Collections	129
	Using a Set	131
	Associative Maps	132
	Immutable Lists	134
	Method Name Convention	137
	The for Expression	139
9.	Pattern Matching and Regular Expressions	143
	Comprehensive Matching	143
	Pattern Variables and Constants in case Expressions	147
	Matching Using case Classes	149
	Extractors and Regular Expressions	152
	The Omnipresent Underscore Character	158
10.	Handling Exceptions	161
	Exceptions in Scala	161
	Mind the Catch Order	163
11.	Programming Recursions	167
	A Simple Recursion	167
	Tail Call Optimization (TCO)	168
	Trampoline Calls	173

Part III — Concurrency in Scala

12.	Lazy Evaluations and Parallel Collections	179
	Let's Get Lazy	179
	Making Strict Collections Lazy	182
	The Ultimately Lazy Streams	185
	Parallel Collections	189
13.	Programming with Actors	195
	A Sequential Time-Consuming Problem	195
	The Treacherous Paths to Concurrency	197
	Creating Actors	198
	Actors and Threads	201
	Isolated Mutability	202
	Concurrency with Actors	206
	Usage Recommendations	209

Part IV — Applying Scala

14.	Intermixing with Java	213
	Using Scala Classes from Scala	213
	Using Java Classes from Scala	215
	Using Scala Methods from Java	217
	Using Traits from Java	218
	Using Singletons and Companion Objects from Java	221
	Extending Classes	223
15.	Creating an Application with Scala	227
	Getting Users' Input	227
	Reading and Writing Files	228
	XML as a First-Class Citizen	229
	Reading and Writing XML	232
	Getting Stock Prices from the Web	235
	Making the Net Asset Application Concurrent	238
16.	Unit Testing	241
	Using JUnit	241
	Using ScalaTest	243
	Using Mockito	244
A1.	Installing Scala	251
	Downloading	251
	Installing	251
A2.	Web Resources	253
	Bibliography	255
	Index	257

Acknowledgments

When I signed up to write this book, little did I know about the challenges ahead. Daily routines became unbearable due to a neck injury. After spending months fighting that issue, I decided to back out of this book. Instead of responding as publishers, the Pragmatic Programmers came together as friends and family would. I went looking for a publisher—now I know I found real friends. Thanks to Susannah Pfalzer, Dave Thomas, Andy Hunt, and the rest of the crew who helped with this book.

I sincerely thank the technical reviewers of this book. Thanks to Scott Leberknight for his constructive reviews—whenever he reviews one of my books, my deepest learning happens. I thank Daniel Hinojosa, Rahul Kavale, Anand Krishnan, Ted Neward, Rebecca Parsons, Vahid Pazirandeh, and Ian Roughley for their valuable time and input in the making of this book—I really appreciate what you’ve done. Any errors in the book are solely mine.

Thanks to everyone who purchased a copy of this book in the beta form. Thanks to David Dieulivol and Chris Searle for submitting errata.

I greatly benefited from Jackie Carter’s encouragement, support, advice, and suggestions. She is an absolute pleasure to interact with and such a guiding force. Thank you, Jackie, for all you’ve done. You made writing this book a pleasurable task.

I could not have done any of this without the support of my wife Kavitha and sons Karthik and Krupa—thank you, folks.



Introduction

Glad to see your interest in Scala. Thank you for choosing this book to learn and exercise the combined power of two programming paradigms—object-oriented and functional—fused together in one language.

The Java ecosystem is one of the most powerful platforms to develop and deploy enterprise applications. It's ubiquitous and versatile; it has a rich set of libraries and runs on multiple types of hardware; and there are well over 200 languages to program on it.

I've had the privilege to learn and work with a dozen languages and have written books on a few. Languages are like vehicles—they come with different capabilities and help us navigate the platform. It's quite heartwarming to see that programmers today have the liberty to choose from, and also intermix, multiple languages to program their applications.

Typical enterprise applications suffer from multiple issues—verbose code is hard to maintain, mutability increases bugs, and shared mutability turns the pleasurable task of programming concurrency into hell. We've repeatedly fallen prey to accidental complexities that arise from poor abstractions offered by mainstream languages.

Scala is one of the most powerful languages that compiles down to bytecode. It's statically typed, concise, and expressive, and it's being used to develop performant, scalable, responsive, and resilient applications by many organizations.

The right set of features have come together in this language to remove a number of traps. The Scala language and its libraries let us focus on the problem domain rather than being bogged down by low-level infrastructure details like threads and synchronization.

Scala has been designed to create applications that require high performance, faster response, and greater simplicity. It's a language created to meet the

easy ●●●
computing

high frequency and volume of data processing that large corporations and social media demand.

Scala has been used to build applications in various domains, including telecommunications, social networking, semantic web, and digital asset management. Apache Camel uses Scala for its DSL to create routing rules. Play and Lift are powerful web development frameworks built using Scala. Akka, built using Scala, is a prominent library for creating highly responsive concurrent and reactive applications. These libraries and frameworks take full advantage of Scala features such as conciseness, expressiveness, pattern matching, and concurrency.

Scala is a powerful language, but to get productive with it we need to focus on the essential parts of the language that provide the most value. This book will help you learn the essentials of Scala, so you can quickly get productive, get your work done, and create practical applications.

And, to help you create practical applications, Scala offers two different styles of programming.

Programming Styles in Scala

Scala does not limit us to one programming style. We can program with objects, or in functional style, and also mix the two to get the best of both worlds.

Java programmers are familiar and comfortable with OOP. Scala is object-oriented and statically typed—a notch more than Java on both fronts. That's good news since our investment over the years in OOP is not wasted but earns dividends as we begin to program in Scala. When creating traditional applications we can lean toward the OO style provided by Scala. We can write code much like the way we're used to in Java, leveraging the power of abstraction, encapsulation, inheritance, and above all, polymorphism. At the same time, we're not restricted to this model when our needs stretch beyond its strengths.

The functional style of programming is gaining traction, and Scala readily supports that as well. We can lean more easily toward immutability, create pure functions, reduce accidental complexities, and apply function composition and lazy evaluations. With the full benefit of the functional style we can create high-performant—single-threaded and multithreaded—applications in Scala.



Scala and Other Languages

Scala has drawn a good number of features from other languages, most notably Erlang. The actor-based concurrency in Scala was inspired by the success of that model in Erlang. Likewise, static typing and type inference in Scala was influenced by languages like Haskell. Functional style capabilities came from several languages that came before.

With the introduction of lambda expressions and the powerful streams API in Java 8 (see [Functional Programming in Java: Harnessing the Power of Java 8 Lambda Expressions \[Sub14\]](#)) we can write functional style code in Java. This is not a threat to Scala or any of the other languages on the JVM; instead it closes the gap between these languages, making it less difficult for programmers to adopt or switch between languages.

Scala nicely fits into the Java ecosystem, and we can readily use Java libraries from Scala. We can build full applications entirely in Scala or intermix it with Java and other languages on the JVM. So, Scala code could be as small as a script or as large as a full-fledged enterprise application.

Who Is This Book For?

This book is for experienced Java programmers. I assume you know the Java language syntax and the Java API. I also assume you have strong object-oriented programming skills. These assumptions will allow you to quickly get into the essence of Scala and make use of it on real applications.

Developers who are familiar with other languages can use this book as well but will have to supplement it with good Java books.

Programmers who are somewhat familiar with Scala can use this book to learn some language features that they may not otherwise have had the opportunity to explore. Those already familiar with Scala can use this book for training fellow programmers in their organizations.

What's in This Book?

My objective in writing this book is to get you up to speed on Scala so you can use it to write scalable, responsive, and resilient applications. There is a lot you need to learn to do that, but there is a lot more you don't need to know as well. If your objective is to learn everything that there is to learn about Scala, you will not find that in this book. There are other books on Scala that do a great job of introducing the language in great depth. What you will see in this book are essential concepts that you need to know to start using Scala.

easy
computing

Pragmatic Programming

We'll show you how to be more pragmatic and effective, for new code and old.

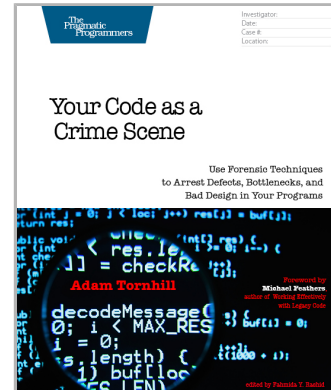
Your Code as a Crime Scene

Jack the Ripper and legacy codebases have more in common than you'd think. Inspired by forensic psychology methods, this book teaches you strategies to predict the future of your codebase, assess refactoring direction, and understand how your team influences the design. With its unique blend of forensic psychology and code analysis, this book arms you with the strategies you need, no matter what programming language you use.

Adam Tornhill

(218 pages) ISBN: 9781680500387. \$36

<https://pragprog.com/book/atcrime>



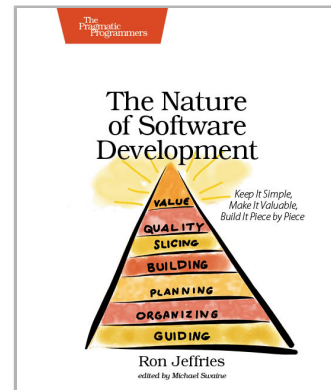
The Nature of Software Development

You need to get value from your software project. You need it “free, now, and perfect.” We can’t get you there, but we can help you get to “cheaper, sooner, and better.” This book leads you from the desire for value down to the specific activities that help good Agile projects deliver better software sooner, and at a lower cost. Using simple sketches and a few words, the author invites you to follow his path of learning and understanding from a half century of software development and from his engagement with Agile methods from their very beginning.

Ron Jeffries

(178 pages) ISBN: 9781941222379. \$24

<https://pragprog.com/book/rjnsd>



The Pragmatic Bookshelf

The Pragmatic Bookshelf features books written by developers for developers. The titles continue the well-known Pragmatic Programmer style and continue to garner awards and rave reviews. As development gets more and more difficult, the Pragmatic Programmers will be there with more titles and products to help you stay on top of your game.

Visit Us Online

This Book's Home Page

<https://pragprog.com/book/vsscala2>

Source code from this book, errata, and other resources. Come give us feedback, too!

Register for Updates

<https://pragprog.com/updates>

Be notified when updates and new books become available.

Join the Community

<https://pragprog.com/community>

Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

New and Noteworthy

<https://pragprog.com/news>

Check out the latest pragmatic developments, new titles and other offerings.

Buy the Book

If you liked this eBook, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: <https://pragprog.com/book/vsscala2>

Contact Us

Online Orders: <https://pragprog.com/catalog>

Customer Service: support@pragprog.com

International Rights: translations@pragprog.com

Academic Use: academic@pragprog.com

Write for Us: <http://write-for-us.pragprog.com>

Or Call: +1 800-699-7764

