Extracted from:

Pragmatic Scala

Create Expressive, Concise, and Scalable Applications

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Scala 2.11

Pragmatic Scala

Create Expressive, Concise, and Scalable Applications

Venkat Subramaniam

edited by Jacquelyn Carter

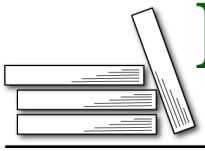
Pragmatic Scala

Create Expressive, Concise, and Scalable Applications

Venkat Subramaniam

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina



Pragmatic Bookshelf

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*.

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

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 resilience. It's a language created to meet the

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.

I assume you are quite familiar with Java. So, you will not learn basic concepts of programming from this book. However, I do not assume you have knowledge of functional programming or the Scala language itself—you will learn that in this book.

I have written this book for a busy Java developer, so my objective is to make you comfortable with Scala quickly so you can start building parts of your application with it really soon. You will see that the concepts are introduced fairly quickly but with lots of examples.

There is no better way to learn a language than trying out examples—a lot of them. As you follow along with the book, key in the examples, run them, see the output, modify them with your own ideas, experiment, break the code, and put it back together. That's the most fun way to learn.

Scala Version Used in This Book

Using automated scripts, the examples in this book have been tried out with the following version:

```
Scala code runner version 2.11.7 -- Copyright 2002-2013, LAMP/EPFL
```

Take a few minutes to download the appropriate version of Scala for your system. This will help you follow along with the examples in this book.

Online Resources

You can download all the example source code for the book from the Pragmatic Bookshelf website for this book.[1] You can also provide feedback by submitting errata entries or posting your comments and questions in the forum.

If you're reading the book in PDF form, you can click on the link above a code listing to view or download the specific examples.

A number of web resources referenced throughout the book are collected in Appendix 2, *Web Resources*, on page ?. Here are a few that will help you get started with this book:

To download Scala visit the official website for the language.[2] You can find documentation for the Scala library at the documentation page.[3]

Let's ascend Scala.

1. http://www.pragprog.com/titles/vsscala2
2. http://www.scala-lang.org/download
3. http://www.scala-lang.org/api