Introduction

This book is a tutorial for the Scala programming language, written by people directly involved in the development of Scala. Our goal is that by reading this book, you can learn everything you need to be a productive Scala programmer. All examples in this book compile with Scala version 2.7.2.

Who should read this book

The main target audience for this book is programmers who want to learn to program in Scala. If you want to do your next software project in Scala, then this is the book for you. In addition, the book should be interesting to programmers wishing to expand their horizons by learning new concepts. If you're a Java programmer, for example, reading this book will expose you to many concepts from functional programming as well as advanced object-oriented ideas. We believe learning about Scala, and the ideas behind it, can help you become a better programmer in general.

General programming knowledge is assumed. While Scala is a fine first programming language, this is not the book to use to learn programming.

On the other hand, no specific knowledge of programming languages is required. Even though most people use Scala on the Java platform, this book does not presume you know anything about Java. However, we expect many readers to be familiar with Java, and so we sometimes compare Scala to Java to help such readers understand the differences.

How to use this book

Because the main purpose of this book is to serve as a tutorial, the recommended way to read this book is in chapter order, from front to back. We have tried hard to introduce one topic at a time, and explain new topics only

in terms of topics we've already introduced. Thus, if you skip to the back to get an early peek at something, you may find it explained in terms of concepts you don't quite understand. To the extent you read the chapters in order, we think you'll find it quite straightforward to gain competency in Scala, one step at a time.

If you see a term you do not know, be sure to check the glossary and the index. Many readers will skim parts of the book, and that is just fine. The glossary and index can help you backtrack whenever you skim over something too quickly.

After you have read the book once, it should also serve as a language reference. There is a formal specification of the Scala language, but the language specification tries for precision at the expense of readability. Although this book doesn't cover every detail of Scala, it is quite comprehensive and should serve as an approachable language reference as you become more adept at programming in Scala.

How to learn Scala

You will learn a lot about Scala simply by reading this book from cover to cover. You can learn Scala faster and more thoroughly, though, if you do a few extra things.

First of all, you can take advantage of the many program examples included in the book. Typing them in yourself is a way to force your mind through each line of code. Trying variations is a way to make them more fun and to make sure you really understand how they work.

Second, keep in touch with the numerous online forums. That way, you and other Scala enthusiasts can help each other. There are numerous mailing lists, discussion forums, a chat room, a wiki, and multiple Scala-specific article feeds. Take some time to find ones that fit your information needs. You will spend a lot less time stuck on little problems, so you can spend your time on deeper, more important questions.

Finally, once you have read enough, take on a programming project of your own. Work on a small program from scratch, or develop an add-in to a larger program. You can only go so far by reading.

EBook features

This book is available in both paper and PDF eBook form. The eBook is not simply an electronic copy of the paper version of the book. While the content is the same as in the paper version, the eBook has been carefully designed and optimized for reading on a computer screen.

The first thing to notice is that most references within the eBook are hyperlinked. If you select a reference to a chapter, figure, or glossary entry, your PDF viewer should take you immediately to the selected item so that you do not have to flip around to find it.

Additionally, at the bottom of each page in the eBook are a number of navigation links. The "Cover," "Overview," and "Contents" links take you to the front matter of the book. The "Glossary" and "Index" links take you to reference parts of the book. Finally, the "Discuss" link takes you to an online forum where you discuss questions with other readers, the authors, and the larger Scala community. If you find a typo, or something you think could be explained better, please click on the "Suggest" link, which will take you to an online web application where you can give the authors feedback.

Although the same pages appear in the eBook as the printed book, blank pages are removed and the remaining pages renumbered. The pages are numbered differently so that it is easier for you to determine PDF page numbers when printing only a portion of the eBook. The pages in the eBook are, therefore, numbered exactly as your PDF viewer will number them.

Typographic conventions

The first time a *term* is used, it is italicized. Small code examples, such as `x + 1`, are written inline with a mono-spaced font. Larger code examples are put into mono-spaced quotation blocks like this:

```
def hello() {
  println("Hello, world!")
}
```

When interactive shells are shown, responses from the shell are shown in a lighter font.

```
scala> 3 + 4
res0: Int = 7
```

Content overview

- Chapter 1, "A Scalable Language," gives an overview of Scala's design as well as the reasoning, and history, behind it.
- Chapter 2, "First Steps in Scala," shows you how to do a number of basic programming tasks in Scala, without going into great detail about how they work. The goal of this chapter is to get your fingers started typing and running Scala code.
- Chapter 3, "Next Steps in Scala," shows you several more basic programming tasks that will help you get up to speed quickly in Scala. After completing this chapter, you should be able to start using Scala for simple scripting tasks.
- Chapter 4, "Classes and Objects," starts the in-depth coverage of Scala with a description of its basic object-oriented building blocks and instructions on how to compile and run a Scala application.
- Chapter 5, "Basic Types and Operations," covers Scala's basic types, their literals, the operations you can perform on them, how precedence and associativity works, and what rich wrappers are.
- Chapter 6, "Functional Objects," dives more deeply into the object-oriented features of Scala, using functional (*i.e.*, immutable) rational numbers as an example.
- Chapter 7, "Built-in Control Structures," shows you how to use Scala's built-in control structures: `if`, `while`, `for`, `try`, and `match`.
- Chapter 8, "Functions and Closures," provides in-depth coverage of functions, the basic building block of functional languages.
- Chapter 9, "Control Abstraction," shows how to augment Scala's basic control structures by defining your own control abstractions.
- Chapter 10, "Composition and Inheritance," discusses more of Scala's support for object-oriented programming. The topics are not as fundamental as those in Chapter 4, but they frequently arise in practice.
- Chapter 11, "Scala's Hierarchy," explains Scala's inheritance hierarchy and discusses its universal methods and bottom types.

- Chapter 12, "Traits," covers Scala's mechanism for mixin composition. The chapter shows how traits work, describes common uses, and explains how traits improve on traditional multiple inheritance.

- Chapter 13, "Packages and Imports," discusses issues with programming in the large, including top-level packages, import statements, and access control modifiers like `protected` and `private`.

- Chapter 14, "Assertions and Unit Testing," shows Scala's assertion mechanism and gives a tour of the various tools available for writing tests in Scala.

- Chapter 15, "Case Classes and Pattern Matching," introduces twin constructs that support you when writing regular, non-encapsulated data structures. Case classes and pattern matching are particularly helpful for tree-like recursive data.

- Chapter 16, "Working with Lists," explains in detail lists, which are probably the most commonly used data structure in Scala programs.

- Chapter 17, "Collections," shows you how to use the basic Scala collections, such as lists, arrays, tuples, sets, and maps.

- Chapter 18, "Stateful Objects," explains stateful (*i.e.*, mutable) objects, and the syntax Scala provides to express them. The chapter concludes with a case study on discrete event simulation, which shows some stateful objects in action.

- Chapter 19, "Type Parameterization," explains some of the techniques for information hiding introduced in Chapter 13 by means of a concrete example: the design of a class for purely functional queues. The chapter builds up to a description of variance of type parameters and how it interacts with information hiding.

- Chapter 20, "Abstract Members," describes all kinds of abstract members that Scala supports. Not only methods, but also fields and types can be declared abstract.

- Chapter 21, "Implicit Conversions and Parameters," covers two constructs that can help you omit tedious details from source code, letting the compiler supply them instead.

- Chapter 22, "Implementing Lists," describes the implementation of class `List`. It is important to understand how lists work in Scala, and furthermore the implementation demonstrates the use of several of Scala's features.
- Chapter 23, "For Expressions Revisited," shows how `for` expressions are translated to invocations of `map`, `flatMap`, `filter`, and `foreach`.
- Chapter 24, "Extractors," shows how to pattern match against arbitrary classes, not just case classes.
- Chapter 25, "Annotations," shows how to work with language extension via annotation. The chapter describes several standard annotations and shows you how to make your own.
- Chapter 26, "Working with XML," explains how to process XML in Scala. The chapter shows you idioms for generating XML, parsing it, and processing it once it is parsed.
- Chapter 27, "Objects As Modules," shows how Scala's objects are rich enough to remove the need for a separate modules system.
- Chapter 28, "Object Equality," points out several issues to consider when writing an `equals` method. There are several pitfalls to avoid.
- Chapter 29, "Combining Scala and Java," discusses issues that arise when combining Scala and Java together in the same project, and suggests ways to deal with them.
- Chapter 30, "Actors and Concurrency," shows you how to use Scala's actors concurrency library. Although you use the Java Platform's concurrency primitives and libraries from Scala programs, actors can help you avoid the deadlocks and race conditions that plague the traditional "threads and locks" approach to concurrency.
- Chapter 31, "Combinator Parsing," shows how to build parsers using Scala's library of parser combinators.
- Chapter 32, "GUI Programming," gives a quick tour of a Scala library that simplifies GUI programming with Swing.
- Chapter 33, "The `SCells` Spreadsheet," ties everything together by showing a complete spreadsheet application written in Scala.

Resources

At `http://www.scala-lang.org`, the main website for Scala, you'll find the latest Scala release and links to documentation and community resources. For a more condensed page of links to Scala resources, visit the website for this book: `http://booksites.artima.com/programming_in_scala`. To interact with other readers of this book, check out the Programming in Scala Forum, at: `http://www.artima.com/forums/forum.jsp?forum=282`.

Source code

You can download a ZIP file containing the source code of this book, which is released under the Apache 2.0 open source license, from the book's website: `http://booksites.artima.com/programming_in_scala`.

Errata

Although this book has been heavily reviewed and checked, errors will inevitably slip through. To see a (hopefully short) list of errata for this book, visit `http://booksites.artima.com/programming_in_scala/errata`. If you find an error, please report it at the above URL, so that we can be sure to fix it in a future printing or edition of this book.