STEPS IN SCALA

An Introduction to Object-Functional Programming

Object-functional programming is already here. Scala is the most prominent representative of this exciting approach to programming, both in the small and in the large. In this book we show how Scala proves to be a highly expressive, concise, and scalable language, which grows with the needs of the programmer, whether professional or hobbyist.

Read the book to see how to:

- leverage the full power of the industry-proven JVM technology with a language that could have come from the future;
- learn Scala step-by-step, following our complete introduction and then dive into specially chosen design challenges and implementation problems, inspired by the real-world, software engineering battlefield;
- embrace the power of static typing and automatic type inference;
- use the dual object and functional oriented natures combined at Scala's core, to see how to write code that is less "boilerplate" and to witness a real increase in productivity.

Use Scala for fun, for professional projects, for research ideas. We guarantee the experience will be rewarding.

Christos K. K. Loverdos is a research inclined computer software professional. He holds a B.Sc. and an M.Sc. in Computer Science. He has been working in the software industry for more than ten years, designing and implementing flexible, enterprise-level systems and making strategic technical decisions. He has also published research papers on topics including digital typography, service-oriented architectures, and highly available distributed systems. Last but not least, he is an advocate of open source software.

Apostolos Syropoulos is a computer scientist. He holds a B.Sc. in Physics, an M.Sc. in Computer Science, and a Ph.D. in Theoretical Computer Science. His research interests focus on computability theory, category theory, fuzzy set theory, and digital typography. He has authored or co-authored six books, was co-editor of a multi-author volume, and has published more than 50 papers and articles.

STEPS IN SCALA

An Introduction to Object-Functional Programming

CHRISTOS K. K. LOVERDOS
APOSTOLOS SYROPOULOS

CAMBRIDGE
UNIVERSITY PRESS

Shaftesbury Road, Cambridge CB2 8EA, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi – 110025, India

103 Penang Road, #05–06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of Cambridge University Press & Assessment, a department of the University of Cambridge.

We share the University's mission to contribute to society through the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9780521762175

First published 2010

A catalogue record for this publication is available from the British Library

ISBN 978-0-521-76217-5 Hardback
ISBN 978-0-521-74758-5 Paperback

Additional resources for this publication at www.cambridge.org/9780521762175

To Katerina, who is always here and is constantly making me a better person

CKKL

Στους γονείς μου Γεώργιο και Βασιλική και στον γιο μου Δημήτριο-Γεώργιο

AS

Contents

Preface

What is Scala?

Scala is a relatively new programming language that was designed by Martin Odersky and released in 2003. The distinguishing features of Scala include a seamless integration of functional programming features into an otherwise object-oriented language. Scala owes its name to its ability to *scale*, that is, it is a language that can *grow* by providing an infrastructure that allows the introduction of new constructs and data types. In addition, Scala is a concurrent programming language, thus, it is a tool for today as well as tomorrow! Scala is a compiled language. Its compiler produces bytecode for the Java Virtual Machine, thus allowing the (almost) seamless use of Java tools and constructs from within scala. The language has been used to rewrite Twitter's[1] back-end services. In addition, almost all of Foursquare's[2] infrastructure has been coded in Scala. This infrastructure is used by several companies worldwide (for example, Siemens, Sony Pictures Imageworks).

Who should read this book?

The purpose of this book is twofold: first to teach the basics of Scala and then to show how Scala can be used to develop *real* applications. Unlike other books on Scala, this one does not assume any familiarity with Java. In fact, no previous knowledge of Java is necessary to read this book, though some knowledge of Java would be beneficial, especially in the chapter on GUI applications. On the other hand, the book assumes that readers do have a very basic understanding of programming concepts. In particular, we expect readers to be familiar with terms like compiler, interpreter, (character) string, etc. Thus, the book can be used by anyone who has done some high school computer programming. However, the book covers a number of subjects that are quite advanced and so are appropriate for readers with

[1] `http://www.twitter.com` (a.k.a. Twitter) is a free social networking and micro-blogging service.
[2] `http://foursquare.com` is a location-based social networking service.

a good background in both programming and mathematics. Sections describing such topics are marked with an asterisk (*).

The intended audience of this book includes computer science students as well as computing professionals. Obviously, students and practitioners of related fields and areas (for example, mathematics, physics, electrical and computer engineering, etc.) will find this book quite beneficial.

The book in detail

Essentially, the book is divided in two parts – the first seven chapters introduce most of the language constructs and related software modules, while the remaining six chapters present various applications of Scala. In particular, the first chapter of the book is an introduction to the basic ideas described in the rest of the book. In particular, it describes the basic ideas behind object-orientation, functional programming, and language extensionality, while it concludes with a comparison and discussion of programming languages similar to Scala.

In Chapter 2 we gradually introduce the various basic concepts and ideas of Scala. In particular, we present the "basic" data-types, classes and objects, methods and operators, and functions. Then we introduce some important predefined types-classes: sets, hash tables, lists and strings. In addition, we discuss other important features such as memo functions, regular expressions, annotations, etc.

Pattern matching is an another important feature of Scala that can be used to define useful structures like trees. In Chapter 3 we introduce traits in order to show how behaviors can be mixed in using them. Next, we discuss function objects. Polymorphism is an important characteristic of object-orientation and therefore is an important part of Scala. We discuss all aspects of Scala polymorphism, even higher-order polymorphism. We also discuss streams, setters and getters, memo functions, and we conclude with a discussion of monads.

Chapter 4 is about parser builders, that is, tools that can be used to implement language processors. After introducing the so-called basic parser combinators, we show how they can be used to build the interpreter of a relatively simple programming language. The chapter concludes with a short description of domain-specific languages and monadic parsing.

XML processing is a basic characteristic of Scala. In Chapter 5 we discuss how one can create and manipulate XML content using Scala. In addition, we show how to perform a number of important operations such as searching and printing. Also, we show how to produce XHTML content with Scala.

In Chapter 6 we show how to program GUI applications using Scala. In particular, we show how to use a number of GUI components such as frames, all sorts of

buttons, labels, text fields, dialogs, menus, tabs, and tables. In addition, we show how to implement applets in Scala. The chapter concludes with a short discussion of functional graphics.

It was said above that Scala is a concurrent programming language in the sense that it includes a number of features that facilitate concurrent programming. Chapter 7 is dedicated to these facilities. In particular, we discuss threads and synchronization, animation using threads, mailboxes (a precursor of actors), and actors.

Chapter 8 deals with a ubiquitous abstraction, that of paths. Paths are used mainly to describe file locations. Although our design and implementation is heavily influenced by file-based APIs, we place paths in a more general algebraic context.

Chapter 9 moves from path modeling to file system hierarchies modeling by using the now widely accepted notion of a virtual file system (VFS). We build three VFS implementations: a traditional file system, one based on zip (compressed archives) files and finally a memory-based VFS.

Chapter 10 introduces the concept of file matching, inspired by Unix-related terminology and tools. But instead of just reproducing known behavior, we take full advantage of Scala's DSL definition abilities and make file searches more user-friendly than ever.

Chapter 11 extends the basic idea of the previous chapter, regarding the methodologies to search for the appropriate files. It presents two complementary techniques: one based on the classical notion of iteration and the other based on the emerging notion of traversal. During the course of study, we discover not so traditional ways to abstract over our data and clearly show how a pre-order depth-first search can share almost the same codebase with a breadth-first search. We conclude with a set of thought provoking remarks on the interplay between iteration and traversal.

Chapter 12 introduces and analyzes the expression problem, a not so widely known software design problem. Since its essence lies at the frontier of combining data with operations, we feel that this particular problem should be brought to the attention of a wider audience. Based on work by well-known researchers (including the creator of Scala, Martin Odersky) we build a small code library that follows a consistent set of naming conventions in order to help us tackle the expression problem.

Chapter 13 is a short chapter that shows how one can easily construct a relatively simple computer algebra system.

There are four appendices. The first one briefly discusses how one can construct multimedia applications with Scala. The second one shows how we can use the open-source tool Proguard to package Scala applications along with the Scala

runtime in order to avoid any prerequisite when distributing Scala applications. The third appendix presents the Scala grammar. The fourth and last appendix presents the wealth of command line options of Scala's compiler and interpreter.

Each chapter contains a number of exercises that have been designed to help readers obtain a deeper understanding of the topics presented. There are no solutions to exercises, though in some cases material that follows the exercises contains the solution. In addition, there are some suggestions for programming projects. In most cases these are not easy and some of them are quite challenging.

In the book we use the term Unix, but since this may mean different things, we need to calarify the meaning. The term Unix means either the operating system originally created by Bell Labs or an operating system certified as Unix by the Open Group (for example, Solaris 10 is such a system). We use the term Unix to denote any system that seems sufficiently Unix-like (for example, OpenSolaris, Linux, MacOS, etc.) or is a certified Unix system.

All of the examples presented in the following pages have been tested to work under OpenSolaris and MacOS X [Snow] Leopard. We do not expect that readers who use different computer platforms will encounter any kind of problem, as long as they use the latest version of Java's JDK from Oracle. All examples are available from the book's web site.

Preparing the book

The book has been typeset using a Unicode-aware extension of LaTeX that runs atop of a novel typesetting engine created by Jonathan Kew. We have used Minion Pro to set the text of the book and GFS Neohellenic (by the Greek Font Society) to set captions. In addition, we have used UM Typewriter (created by Apostolos) to typeset code snippets. Asana Math (also by Apostolos) has been used to set mathematical text in this book. Our working platforms were MacOS X [Snow] Leopard (Christos) and OpenSolaris (Apostolos).

Thoughts before delving into the book

Sometimes, when introducing a new language, a new technology, a new approach, we may hear a great deal of technical arguments in favor. Scala can be introduced like that. For the reader who seeks technical ability and excellence in the everyday tools, Scala will provide a solid work-field. For the language enthusiast, the exploring student, the hobbyist programmer, the geek, the most important thing is that Scala can increase your enjoyment of programming. This has been our feeling while preparing this book and while using Scala in our everyday work.

Acknowledgments

First of all we would like to thank Heather Bergman, a former computer science editor of CUP's New York branch, who believed in this project. Heather has helped us during the writing of the book in every possible way! Also, we would like to thank Clare Dennison, our pre-production editor, and Jonathan Ratcliffe, our production editor. In addition, we would like to thank Michael Drig, Tony Morris, Bruno Oliveira, George Georgiou, and the anonymous reviewers for their help.