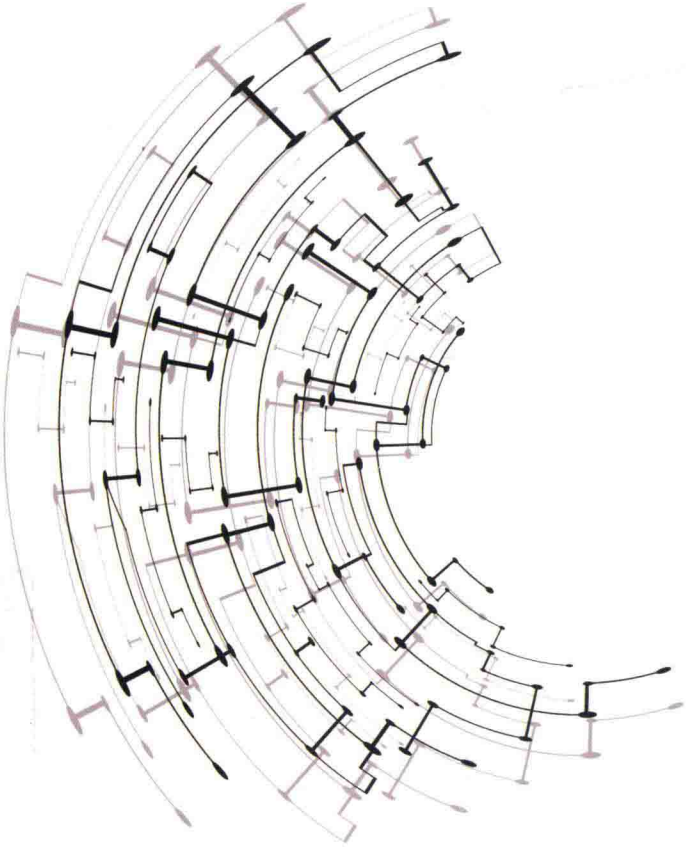
FUNCTIONAL PROGRAMMING IN C#, HOW TO WRITE BETTER C# CODE

C#函数式编程
编写更优质的C#代码

[美] 恩里科·博南诺(Enrico Buonanno) 著
张久修 译



MANNING

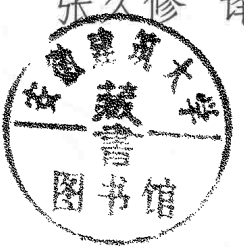
清华大学出版社

C#函数式编程

编写更优质的C#代码

[美]恩里科·博南诺(Enrico Buonanno) 著

张久修 译



清华大学出版社

北 京

Enrico Buonanno
Functional Programming in C#, How to Write Better C# Code
EISBN: 978-1-61729-395-5

北京市版权局著作权合同登记号　图字：01-2017-8973

图书在版编目(CIP)数据

C#函数式编程 编写更优质的 C#代码 / (美)恩里科·博南诺(Enrico Buonanno) 著；张久修译. —北京：清华大学出版社，2019
书名原文：Functional Programming in C#, How to Write Better C# Code
ISBN 978-7-302-51055-0

Ⅰ. ①C…　Ⅱ. ①恩…　②张…　Ⅲ. ①C 语言－程序设计　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2018)第 192028 号

责任编辑： 王　军　韩宏志
封面设计： 周晓亮
版式设计： 思创景点
责任校对： 牛艳敏
责任印制： 丛怀宇

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 三河市龙大印装有限公司
经　　销： 全国新华书店
开　　本： 170mm×240mm　　**印　　张：** 24　　**字　　数：** 484 千字
版　　次： 2019 年 1 月第 1 版　　**印　　次：** 2019 年 1 月第 1 次印刷
定　　价： 98.00 元

产品编号：078166-01

译者序

历经数月，终于完成了本书的翻译工作。没有过多的轻松和兴奋之情，反而更多的是一份疼惜，一份不舍。从事技术书籍的翻译工作，既是一个付出的过程，也是一个收获的过程，更是对自己技术的一种沉淀。每当一本书的翻译工作结束后，我便会无所适从，仿佛少了些什么似的。每天都要花费几个小时的翻译工作，可让我的内心真正沉静下来，只专注于字里行间。那种状态、那种感觉真的很好，也正是如此，内心便充满依依不舍之情。

生活中充满了未知以及不确定性，谁也不能保证你的内心每时每刻都是清静的。在翻译本书的这段时间内，我也经历了或多或少会影响心境的一些琐事。可见，想要专心地做一件需要长期坚持且耗费精力的事情是多么艰难。因此，虽未曾写过书，但即使是翻译一本书，也能深深体会到作者付出的心血，可以想象到写书人夜以继日工作的情景。同时，对于能成为本书的译者，我感到特别荣幸。

正所谓“工欲善其事，必先利其器”。函数式编程作为主流编程的重要组成部分，必将是高级程序员手中的一大利器。而想要掌握并使用这件利器，首先需要了解和培养函数式思维，以不同的视角来看待代码。我们首先需要从宏观上了解函数式编程，了解它是什么，及其定义、特性、优点分别是什么，甚至其所存在的缺点或顾虑又有哪些。

顾名思义，函数式编程是一种特定的编程方式，将计算机运算视为函数的计算。简单来说它是一种“编程范式”，也就是如何编写程序的方法论。同时，它又属于“结构化编程”的一种，主要思想是将运算过程尽量写成一系列嵌套的函数调用。与指令式编程相比，函数式编程强调函数的计算比指令的执行重要；与过程化编程相比，函数式编程里函数的计算可随时调用。而函数编程语言最重要的基础是λ演算(lambda calculus)，而且λ演算的函数可接受函数当作输入(参数)和输出(返回值)。

函数式编程具有三个特性：闭包和高阶函数，惰性计算，以及递归。并有五个鲜明特点：函数是“一等公民”(first class)、只用“表达式”而非“语句”、没有“副作用”(side effect)、不修改状态、具有引用透明性。函数式编程的优点为：代码简洁且开发快速、接近自然语言而且易于理解、方便了代码管理、易于“并发编程”，且利于代码的热升级。

凡事有利必有弊。在良好的特性和优点的背后，必然也存在顾虑。函数式编程常被认为严重耗费CPU和存储器资源，主因有二：其一是，早期的函数式编程语言实现时并未考虑过效率问题；第二点是，有些非函数式编程语言为提升速度，不提供自动边界检查或自动垃圾回收等功能。同时，惰性求值亦为语言(如Haskell)增加了额外的管理工作。

以上只是对函数式编程的一个宏观介绍；而通过本书，你可以更加系统、详细地了解并掌握函数式编程。本书由浅入深地讲解函数式编程的基本原理和技术，并通过现实中的各种示例场景带领你完成非函数式编程代码到函数式编程代码的不断重构，以逐渐让你树立函数式编程的思想，引领你从一个全新的视角看待代码。本书必将让你受益匪浅。

作为本书的译者，我本着“诚惶诚恐”的态度投入工作，为避免误导读者，文中的一词一句皆反复斟酌。但是鉴于译者水平有限，错误和失误在所难免，如有任何意见和建议，请不吝指正，感激不尽！

关于本书

如今，函数式编程(Functional Programming，FP)已成为主流编程的一个重要且令人兴奋的组成部分。近十年来创建的大多数语言和框架都是函数式的，这导致人们纷纷预测编程的未来也将是函数式的。与此同时，诸如 C#和 Java 的面向对象主流语言，在其每个新版本中都会引入更多函数式特性，从而实现多范式编程风格。

然而，C#社区关于函数式编程的推行却十分缓慢。为何会这样呢？我认为，其中一个原因是缺乏优秀的文献：

- 大多数 FP 文献都是用函数式语言(尤其是 Haskell)编写的。而对于具有 OOP(Oriented Object Programming，面向对象编程)背景的开发人员来说，要学习其概念就必须跨越编程语言的障碍。尽管许多概念适用于诸如 C#的多范式语言，但同时学习一种新范式和新语言是一项艰巨的任务。
- 更重要的是，文献中的大部分书籍倾向于用数学或计算机科学领域的例子来阐明函数式编程的技术和概念。对于大部分终日从事业务(LOB)应用开发的程序员来说，这会产生一个领域差异，并使得他们难以知悉这些技术与实际应用间的相关性。

这些缺陷是我学习 FP 的主要绊脚石。有些书籍试图解释什么是“柯里化”，通过利用数字 3，来创建一个可将 3 添加到任意数字的函数，以展示 add 函数是如何被柯里化的(在你能想到的所有应用中，它有一点点实际用处吗？)。放弃此类书籍后，我决定追寻一条属于自己的研究之路。这包括学习 6 种函数式语言(最优秀的几种语言)，并探究 FP 中的哪些概念可在 C#中有效地应用，以及研究由大量开发人员有偿撰写的该类型的应用，并且最终撰写了本书。

本书展示如何利用 C#语言的函数式技术来弥补 C#开发人员的语言差异。还展示如何将这些技术应用于典型的业务场景来弥补领域差异。我采取了一种务实的方法，并且涵盖了函数式技术，使其在典型的 LOB 应用场景中非常有用，并省略了 FP 背后的大部分理论。

最终，你应该关注 FP，因其赋予了以下优势：

- **高效率**——这意味着可用更少的代码完成更多工作。FP 提高了抽象层级，使你可编写高级代码，同时将你从那些只是增加复杂性，却没有任何价值

的低级技术层面解放出来。

- **安全性**——在处理并发性时尤其如此。一个用命令风格编写的程序可能在单线程实现中运行良好，但当并发发生时会导致各种错误。函数式代码在并发场景中提供了更好的保障，因此，在多核处理器时代，我们很自然会看到开发人员对 FP 的兴趣激增。
- **清晰性**——相对于编写新代码，我们会花费更多时间来维护和使用现有的代码，所以让我们的代码清晰明了并且意义明确是非常重要的。当你转向函数式思维时，达到这种清晰性将水到渠成。

如果你已经用面向对象的风格进行了一段时间的编程，在本书的概念实现之前，可能需要做出一些努力和意愿去尝试。为确保学习 FP 是一个愉快而有益的过程，我有两个建议：

- **耐心**——你可能需要多次重复阅读一些章节。你可能会把这本书放下几个星期，当你再次拿起这本书时，突然间发现有些模糊的东西开始变得有意义了。
- **用代码进行实验**——实践出真知。本书提供了许多示例和练习，许多代码片段可在 REPL 中进行测试。

你的同事可能比你更不愿意去探索新东西。预料到他们可能会抗议你采用这种新风格，并对你的代码感到困惑，然后发问“为什么不只是做 x？”(其中 x 是枯燥的、过时的，并且通常是有害的)。不必过多地讨论。坐下来，看着他们最终转身，并用你的技术来解决他们屡次遇到的问题。

致　谢

感谢 Paul Louth，他不但通过自己编写的 LanguageExt 库赋予我灵感(我从中借鉴了很多很棒的想法)，而且亲力亲为地在各个阶段对本书进行了审阅。

Manning 出版社的详尽编辑过程确保了本书的质量。为此，我要感谢与本书合作的团队，包括 Mike Stephens、开发编辑 Marina Michaels、技术编辑 Joel Kotarski 技术校对员 Jürgen Hoetzel，以及版权编辑 Andy Carroll。

特别感谢 Daniel Marbach 和 Tamir Dresher 所给予的技术见解，以及所有参与同行评审的人，包括 Alex Basile、Aurélien Gounot、Blair Leduc、Chris Frank、Daniel Marbach、Devon Burriss、Gonzalo Barba López、Guy Smith、Kofi Sarfo、Pauli Sutelainen、Russell Day、Tate Antrim 和 Wayne Mather。

感谢 Scott Wlaschin 在 http://fsharpforfunandprofit.com 上分享的文章，感谢所有通过文章、博客和开源代码分享自己的知识和热情的其他 FP 社区成员。

前　言

本书旨在展示如何利用C#中的函数式技术编写简洁、优雅、健壮和可维护的代码。

本书读者对象

本书是为那些具有雄心壮志的开发人员所编写的。你需要了解C#语言和.NET框架。你需要具备开发实际应用的经验，熟悉OOP的概念、模式和最佳实践。并且，你正在寻求通过学习函数式技术来扩展编程技能，以便可以充分利用C#的多范式语言特性。如果你正在尝试或正在计划学习一门函数式语言，那么本书也将是非常有价值的，因为你将学习如何在一门你所熟悉的语言上进行函数式思考。改变自己的思考方式是很难的；而一旦做到，那么学习任何特定语言的语法将变得相对容易。

本书的组织结构

全书共15章，分为3个部分：

- 第Ⅰ部分介绍函数式编程的基本技术和原理。我们将初窥函数式编程是什么，以及C#是如何支持函数式编程风格的。然后，将研究高阶函数的功能、纯函数及其与可测性的关系、类型和函数签名的设计，以及如何将简单的函数组合到复杂的程序中。在第Ⅰ部分的最后，你将很好地感受到一个用函数式风格所编写的程序是什么样的，以及这种风格所带来的好处。
- 第Ⅱ部分将加快速度，转向更广泛的关注点，例如函数式的错误处理、模块化和组合应用，以及理解状态和表示变化的函数式方法。到第Ⅱ部分结束时，你将掌握一系列工具的用法，将能利用函数式方法来有效地完成许多编程任务。
- 第Ⅲ部分将讨论更高级的主题，包括惰性求值、有状态计算、异步、数据流和并发性。第Ⅲ部分的每章都介绍一些重要技术，它们可能彻底改变你编写软件的方式和思考方式。

你会在每章中找到更详细的主题分类，并在阅读任何特定章节之前，都能从本书的内封了解到需要预先阅读哪些章节。

为实际应用编码

本书旨在让实际场景保持真实。为此，很多例子都涉及实际任务，例如读取配置、连接数据库、验证HTTP请求；对于这些事情，你可能已经知道如何做了，但你将用函数式思维的新视角来重新看待它们。

在本书中，我使用了一个长期运行的例子来说明在编写LOB应用时，FP是如何提供帮助的。为此，我选择了一个在线银行应用，它是虚拟的Codeland银行(BOC)——我知道这或许有些生搬硬套了，但至少它有了必需的三个字母的缩写。由于大多数人都可访问在线银行设施，因此很容易想象其所需的功能，并且清楚地看到所讨论的问题是如何与实际应用关联的。

我也使用了场景来说明如何解决函数式风格中典型的编程问题。在实际的例子和FP概念之间的不断反复，将帮助我们弥合理论与实践之间的差异。

利用函数式库

诸如C#的语言具有函数式特性，但为了充分利用这些特性，你将经常使用便于实现常见任务的库。Microsoft已经提供了几个库，以便进行函数式风格的编程，包括：

- **System.Linq**——这是一个功能库。我假定你是熟悉它的，因为它是.NET的一个重要组成部分。
- **System.Collections.Immutable**——这是一个不可变集合的库，第9章将开始使用它。
- **System.Reactive**——这是.NET的Reactive Extensions的实现，允许你使用数据流，第14章将讨论这些数据流。

当然还有其他许多重要的类型和功能未列举，这些都是FP的主要部分。因此，一些独立的开发人员已经编写了一些开源的代码库来填补这些空白。到目前为止，其中最完整的是LanguageExt，这是由Paul Louth编写的一个库，用于在进行函数式编码时改进C#开发人员的体验。[1]

本书并没有直接使用LanguageExt；相反，将向你展示如何开发自己的函数式实用工具库，且将其命名为LaYumba.Functional，尽管它与LanguageExt在很大程

1 LanguageExt是开源的，可在GitHub和NuGet上找到：https://github.com/louthy/language-ext。

度上是重叠的，但这在教学方面会更有用，原因有如下几点：

- 在本书出版后，将保持代码的稳定。
- 你可以透过现象看本质，将看到看似简单实则强大的函数式构造。
- 你可以专注于基本要素：我将以最纯粹的形式向你展示这些构造，这样你就不会被一个完整的库所处理的细节和边缘情况分散注意力。

代码约定和下载

代码示例使用了 C# 7，大部分与 C# 6 兼容。C# 7 中专门介绍的语言特性仅用于第 10 章及之后章节(另外，1.2 节的几个示例中明确地展示了 C# 7)。可在 REPL 中执行许多较短的代码片段，从而获得动手练习的实时反馈。更多的扩展示例可通过 https://github.com/la-yumba/functional-csharp-code 下载，其中还配有练习的设置和解决方案。

本书中的代码清单重点讨论了正在讨论的主题，因此可能会省略命名空间(namespace)、using 语句、简单的构造函数，或先前代码清单中出现的并保持不变的代码段。如果你想查看代码清单的完整编译版本，可在代码存储库中找到它：https://github.com/la-yumba/functional-csharp-code。

另外，读者也可扫描封底的二维码下载相关资料。

图书论坛

购买本书后，可免费访问由 Manning 出版社运行的私人网络论坛，你可在这里提交有关本书的评论，询问技术问题，并获得作者和其他用户的帮助。可通过 https://forums.manning.com/forums/functional-programming-in-c-sharp 访问该论坛。你也可通过 https://forums.manning.com/forums/about 了解更多关于 Manning 论坛及论坛行为准则的信息。

Manning 出版社为读者提供一个场所，在这里，读者之间以及读者和作者之间可以进行有意义的对话。但不承诺作者的任何具体参与度，作者对论坛的贡献是自愿的(并且是无偿的)。我们建议你尝试向作者提出一些具有挑战性的问题，以免他的兴趣流失！只要本书还在市场上销售，论坛和之前所讨论的内容存档将可从出版商的网站上直接访问。

目　录

第 I 部分　核心概念