TURING 图灵程序设计丛书

MANNING

C# in Depth Third Edition

深入理解C#（第3版）

[英] Jon Skeet 著
姚琪琳 译

- 资深C# MVP扛鼎之作
- 深入理解语言特性，探究本源
- .NET开发人员必读经典

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

C# in Depth Third Edition

深入理解C#（第3版）

[英] Jon Skeet 著
姚琪琳 译

人民邮电出版社
北京

版权声明

献给我的爱子Tom、 Robin和William。

对本书第1版的赞誉

总之，本书可以算是我读过的最好的计算机图书。

——Craig Pelkie，作家，System iNetwork

多年来我一直使用C#进行开发，但本书依然让我惊喜连连。它对委托、匿名方法和协变逆变的绝妙介绍让我印象特别深刻。即使你是一名经验丰富的开发者，本书仍然能让你学到C#语言中一些不为人知的东西。本书之“深入”，是其他书籍无法企及的。

——Adam J. Wolf，Southeast Valley .NET用户组

阅读本书是一大享受。它编排精妙，示例通俗易懂。我非常喜欢Lambda表达式这一章，并且很容易就被这一话题吸引。

——Jose Rolando Guay Paz，CSW Solutions公司Web开发者

作者将关于C#内部机理的丰富知识，汇集成了你手上这本文笔流畅、简洁实用的书。

——Jim Holmes，*Windows Developer Power Tools*作者

措辞严谨，示例精确，用最少的代码展示最全面的特性……阅读本书真是难得的享受啊！

——Franck Jeannin，Amazon评论员

如果你用C#进行了多年的开发，并且想了解一些内部原理，那么本书绝对适合你。

——Golo Roden，作家、演说家、.NET相关技术培训师

我所读过的最好的C#图书。

——Chris Mullins，C# MVP

对第2版的赞誉

一本关于C#的杰作。

——Kirill Osenkov，微软C#团队

如果你想精通C#，那么本书是必读之作。

——Tyson S. Maxwell，Raytheon资深软件工程师

我们打赌这是最好的C# 4图书。

——Nikander Bruggeman和Margriet Bruggeman，Lois & Clark IT Serivces的.NET顾问

对C# 4的独到见解实用且引人入胜。

——Joe Albahari，*LINQPad and C# 4.0 in a Nutshell*的作者

我所读过的最好的C#书籍之一。

——Aleksey Nudelman，C# Computing LLC的CEO

所有专业的C#开发者都应该阅读的书。

——Stuart Caborn，BNP Paribas资深开发者

C#所有主要版本中语言更新方面高度集中的、专家级的资源。对于所有想掌握C#语言最新动态的专业开发人员来说，本书必不可少。

——Sean Reilly，Point2 Technologies的程序员/分析师

为什么要一遍又一遍地阅读基础知识？Jon关注的是有嚼劲儿的新东西！

——Keith Hill，Agilent Technologies的软件架构师

所有你还没意识到需要掌握的C#知识。

——Jared Parsons，微软资深软件开发工程师

序

世上有两类钢琴家。

一类钢琴家弹琴并不是因为他们喜欢，而是因为父母强迫他们上钢琴课。另一类钢琴家弹琴是因为他们喜欢音乐，想创作音乐。他们不需要被强迫，相反，他们陶醉其中，时常忘记什么时候要停下来。

后一类人中，有人是把弹钢琴当作一种爱好。而有人则是为了生活，因此更需要投入、技巧和天赋。他们有一定的灵活性来选择弹奏哪些音乐流派和风格，不过这些选择主要还是由雇主的需要或者听众的口味来决定的。

后一类人中，有人主要就是为了钱，但也有一些专业人士即便没有报酬，也愿意在公共场合弹奏钢琴。他们喜欢运用自己的技巧和天赋为别人演奏音乐。在这个过程中，他们能找到许多乐趣。如果同时还有报酬，当然更是锦上添花。

后一类人中，有人是自学成材的，他们演奏乐曲是不看谱的。这些人有极高的天赋和能力，但除非通过音乐本身，否则无法向别人传递那种直观的感受。还有一些人无论在理论还是实践上都经过了正统的训练，他们能清楚地理解作曲家是用什么手法得到预期的情绪效果，并相应地改进自己的演绎手法。

后一类人中，有人从来没有打开钢琴看它的内部构造。还有一些人则对钢琴的发声原理好奇不已，最后发现是由于杠杆装置和绞盘在音锤敲击琴弦前的瞬间，牵引制音器的擒纵器，他们为弄明白由5 000 ~ 10 000个运动机件组成的这个乐器装置而感到高兴和自豪。

后一类人中，有人会对自己的手艺和成就心满意足，对它们带来的心灵上的愉悦和经济上的收入感到非常满意。但是，还有一些人不仅仅是艺术家、理论家和技师，他们会抽时间以导师的身份，将那些知识传授给其他人。

我不知道Jon Skeet是哪一类钢琴家。但是，我与这位微软C# MVP有多年的电子邮件交流，并经常看他的博客。我本人至少3遍逐字读完他的这本书，我清楚地知道Jon是后一种软件开发者：热情、博学、天资极高、有好奇心以及善于分析——是其他人的好老师。

C#是一种极为实用和快速发展的语言。通过添加查询能力、更丰富的类型推断、精简的匿名函数语法，等等，一种全新风格的编程语言已出现在我们的面前。与此同时，它代表的仍然是一种静态类型的、面向组件的开发方式，C#取得成功的立足之本没有变。

许多新元素会让人有矛盾的感觉。一方面，它们会显得比较“旧”（Lambda表达式可以追溯到20世纪上半叶计算机科学奠基的年代）。与此同时，对于那些习惯了现代面向对象编程的开发

者，它们又可能显得太新和太不熟悉。

Jon掌控了一切。对于需要理解C#最新版本“是什么”和“怎么做”的专业开发者，本书是理想的选择。此外，如果开发者还探索语言为什么要这样设计，从而加深他们对语言的理解，那么本书更是独一无二的。

为了利用语言提供的所有新能力，需要以全新的方式思考数据、函数以及它们之间的关系。这有点儿像经过多年的古典乐训练之后，开始尝试演奏爵士乐——或者相反。不管怎样，我期待下一代C#程序员能够“谱写”出优秀的乐章。祝你“谱曲”愉快，并感谢你选用了C#这个“主调”。

Eric Lippert
Coverity C#分析架构师

前　言

哦，天哪。在撰写这篇前言时，我打开了第2版的前言，第一句话就是感觉距离撰写第1版的前言已经过去很长时间。对于现在来说，第2版是个遥远的记忆，而第1版则恍若隔世。我不知道是因为世界变化快，还是我不明白（记性差），但不管从哪方面来说，都值得静下来思考。

从第1版甚至是第2版到现在，发展格局已经发生了翻天覆地的变化。这里面有很多原因，而移动设备的崛起是最显而易见的因素。但很多挑战依旧没有改变。编写正确的国际化应用程序依然很难。在所有情况下优雅地处理错误依然很难。编写正确的多线程应用程序也依然很难，尽管多年来语言和库的改进已经大大地简化了这个任务。

最重要的是，在这篇前言的背景下，我认为开发人员对语言的掌握即使到了能够确定语言行为的程度，也仍需要了解他们正使用的这种语言。他们也许十分了解每个用到的API，甚至了解一些不经常使用的细枝末节[①]，但语言的核心应该是开发人员忠实的朋友，开发人员可以依靠它们按可预测的方式编写代码。

除了你所敲下的语言的字母，我相信理解它的精髓会有更大的好处。也许有时不管怎么努力，你都气得想砸键盘，但如果按照语言设计者的意图来组织代码，将获得前所未有的愉快体验。

① 我要说明：我对C#中的不安全代码和指针知之甚少。我完全不需要弄清楚它们。

关于封面插图

本书封面上的插图的标题是“音乐家”。插图来自一本土耳其奥斯曼帝国的服饰画册，由伦敦老邦德街的William Miller于1802年1月1日出版。画册的扉页已经丢失，因此我们很难推断准确的创作时间。此书的目录同时使用英语和法语标识插图，每张图片都有创作它的两位艺术家的名字，他们一定会为自己的作品出现在两百年后的一本计算机编程类图书的封面上而倍感惊讶。

Manning出版社的一个编辑在位于曼哈顿西26街“Garage”的古董跳蚤市场买到了这本画册。卖主是住在土耳其安卡拉的一个美国人，交易时间是在那天他准备收摊的时候。这位编辑没带够买这本画册所需的现金，并且卖主礼貌地拒绝了他使用信用卡和支票。卖主当天晚上要飞回安卡拉，看起来好像没什么希望了。该怎么解决呢？两个人最后通过握手约定的老式君子协议解决了。卖主提议通过银行转账付款，编辑在纸上抄下了收款银行的信息，随后画册就到他手里了。不用说，第二天我们就把款付给了卖主。我们感谢这位陌生人能如此信任我们的同事。这让我们回忆起了那个很久以前的美好时代。

我们Manning人崇尚创造性和主动性，而以两个世纪以前的丰富多彩的地区生活作为图书封面的素材，使得计算机商业充满趣味性，这本画册中的这张图片，把我们带到了那时的生活中。

致　谢

你可能认为组织这本书的第3版（新增了两章）十分简单。的确，编写第15章和第16章的“绿色内容”[1]非常容易。但真正的工作远不止这些。我需要全书调整语言的细节，检查当年没问题现在却没道理的部分，还要确保整书能达到读者期望的高标准。幸好，有一群人支持我，让本书没有走改旗易帜的邪路。

最重要的是，我的家人一如既往地令人赞叹。我妻子Holly本身也是个儿童读物作家，所以孩子们已经习惯了我俩关起房门赶工期。但他们仍然自始至终都快乐地鼓励我们。Holly将这一切转化成了动力。她从来没跟我提起过在我撰写第3版的时候，她总共从头到尾写完了多少本书，对此我只能默默点赞。

稍后我会列出正式的审稿人名单，在此我要特别感谢那些订购了第3版预览版的朋友们，他们指出了我的笔误并提出了修改建议，而且不断地询问什么时候能够正式出版。有一批热切期望早日看到本书出版的读者，对我来说是莫大的鼓舞。

我与Manning的团队相处得十分融洽，不管是因第1版的出版而熟识的朋友还是新加入者，与他们一同工作都非常愉快。关于内容的增删，Mike Stephens和Jeff Bleiel给出了极具参考性的意见，他们能把所有事都安排妥当。Andy Carroll和Katie Tennant分别进行了专业的编辑和校对，而且从来不曾对我的英式作风、吹毛求疵或一些常见困惑表现出任何的不耐烦。出版部门一如既往地在背后出色地工作着，无论如何，我要对他们表示感谢：Dottie Marsico、Janet Vail、Marija Tudor和Mary Piergies。最后，我要感谢出版人Marjan Bace，他使我得以撰写本书第3版，并且还就未来我们可能会采取的合作方式提出了一些有趣的建议。

同行评审也十分重要，这不仅可以规范技术细节，还能平衡并协调书中内容。有时我们得到的建议仅涉及书的整体架构，而有时则是关于极为具体的细节（这时我都作了相应的修改）。无论哪种形式的反馈，我都非常欢迎。因此要感谢以下审稿人，他们让本书的质量更上一层楼：Andy Kirsch、Bas Pennings、Bret Colloff、Charles M. Gross、Dror Helper、Dustin Laine、Ivan Todorovic、Jon Parish、Sebastian Martín Aguilar、Tiaan Geldenhuys和Timo Bredenoort。

特别感谢Stephen Toub和Stephen Cleary，他们对本书第15章的早期评审非常宝贵。异步是一个特别复杂的主题，要想写得既清楚又准确十分不易。他们的专业建议使这一章的质量有了极大提升。

① 在大多数版本控制工具中，绿色都代表新增的内容。作者借此来指代新增的两章内容。——译者注

当然，如果没有C#团队，本书根本不可能存在。他们在语言的设计、实现和测试方面的贡献无与伦比，我期待他们接下来提出的内容。第2版出版之后，Eric Lippert就离开了C#团队，开始了新的传奇之旅[①]。令我万分感激的是，他仍然愿意做第3版的技术评审。还要感谢他为本书第1版作序，这一版仍然沿用了它。本书自始至终都有提到他的独到见解，如果你没有读过他的博客文章（http://ericlippert.com），那么你真应该读一读。

① Eric Lippert的博客名称即为代码传奇之旅（Fabulous Adventure in Coding）。——译者注

关于本书

这是一本关于C# 2及后续版本的书——就是这么简单。关于C# 1我几乎没讲什么。至于.NET Framework库和CLR（公共语言运行时），我也只是讨论了它们同语言有关的那一部分。我故意如此，结果就是这本书显然有别于市面上的大多数C#和.NET书籍。

假定读者已经具备了一定的C# 1相关知识，我就不必再花几百页的篇幅来讲述大多数人都已经知道的知识。这样就可以有更多的篇幅来深挖C# 2、C# 3和C# 4的细节，这正是我希望你阅读本书的目的。我写本书第1版时，有些读者对C# 2还相当陌生。现在，几乎所有的C#开发人员用过了C# 2中引入的特性，但这一版仍然保留了对C# 2的介绍，因为它是后续内容的基础。

目标读者

本书面向对C#有所了解的开发人员。最起码要十分了解C# 1，对后续版本略知一二即可。能达到这个要求的读者已经不多了，但我相信仍有很多开发者会通过深入分析C# 2和C# 3而受益，即使他们已经用了一段时间。而且，有很多开发人员根本没有用过C# 4和C# 5。

如果你对C#一无所知，那本书可能不适合你阅读。你可以通过查找你不熟悉的部分来苦读本书，但这并不是有效的学习方法。你最好选另外一本书，然后循序渐进地将本书加入到你的书单中。有很多从头介绍C#的图书，这些书的风格千差万别。*C# in a nutshell*系列（O'Reilly）一直都是这方面的好书，*Essential C# 5.0*（Addison-Wesley Professional）也是不错的选择。

我不是说阅读本书你就会变成一位出色的编码者（coder）。除了知道你正用到的语法，在软件工程中还有许多东西要学。我会在书中适时提供一些指导，但在开发过程中，我们不得不承认，在很多地方，基本上都是要依赖于某种直觉的。我想说的是，如果你阅读并理解了这本书，那么在使用C#时会感觉更加顺手，而且会更加自然地凭自己的直觉行事，而不会有太多的犹豫。这不是说因为用到语言的一些“生僻”的功能，你写出来的代码别人便看不懂了。相反，我是说你将拥有强大的自信，知道自己有哪些选择，并知道C#语言本身在鼓励你选择哪条路走下去。

路线图

本书的结构很简单，共有5个部分和3个附录。第一部分相当于简介，包括对C# 1重点主题的回顾，这些主题对于理解C#的后续版本非常重要，而且经常被人误解。第二部分讲述了C# 2的新特性。第三部分讲述了C# 3的新特性，等等。

某些时候，像这样组织全书内容，意味着一个主题会被反复讲到——尤其是委托在C# 2中有所增强，在C# 3中则进一步增强。但是，我的疯狂中却蕴藏着深意。我估计许多读者在不同的项目中会使用不同版本的C#：例如，在工作中使用C# 4研发，而回到家则使用C# 5实验。所以，我觉得有必要明确哪个版本中有哪些内容。这样还可以营造出一种“现场感”和“进化感”——可以体会到随着时间的推移，语言是如何进化的。

第1章的作用是搭建起一个舞台，从一段简单的C# 1代码开始，让它不断演变，观察更高的版本如何使代码变得越来越易读，越来越强大。我们介绍了C#成长的历史背景，以及它作为一个完整平台的一部分而工作的技术背景。具体地说，C#作为一种语言，它的基础是各种各样的“框架库”（.NET Framework中的各种库）以及一个强大的运行时（runtime）。借助它们，我们可以将抽象的东西转变成现实。

第2章回顾了C# 1的3个特定方面：委托、类型系统的特征以及值类型和引用类型的差异。许多C# 1开发者对这些主题有了“足够”的认识，但由于C#对它们进行了极大的拓展，所以要充分利用那些新特性，就需要一个坚实的基础。

第3章探讨了C# 2最大、同时也有可能最难掌握的一个特性：泛型。方法和类型可以用泛型的方式来写，调用代码中指定的真实类型将被泛型定义中的“类型参数”代替。刚开始，这个说法确实会令人迷惑，但理解了泛型后，就会感觉自己再也离不开它们了。

如果你想过怎样表示一个空整数，第4章就是为你准备的。这一章介绍了可空类型，这是基于泛型建立起来的一个特性，利用了语言、运行时和框架所提供的支持。

第5章介绍了C# 2对委托的增强。在此之前，你也许只用委托处理过像按钮单击这样的事件。C# 2使委托更容易创建，而库的支持使它们在除了事件之外的其他场合更加有用。

第6章讨论了迭代器，以及在C# 2中实现它们的简易方式。很少有开发者使用迭代器块，但由于LINQ to Objects是建立在迭代器基础上的，所以它们会变得越来越重要。它们执行时的“惰性” 也是LINQ的一个关键部分。

第7章对C# 2引入的许多较小的特性进行了描述，它们使我们的编程工作变得更加令人愉悦。语言的设计者对C# 1的一些粗糙的设计进行了改进，现在能够更灵活地与代码生成器交互，能更好地支持工具类，能更细致地访问属性，等等。

第8章同样研究了一些相对简单的特性，但这一次是针对C# 3的。几乎所有新语法都针对LINQ这个共同的目标进行了调整。但是这些基本的构建单元本身也相当有用。使用匿名类型、自动实现的属性、隐式类型的局部变量以及得到显著增强的初始化支持，C# 3变成一个内容更加丰富的语言，使你的代码可以更好地表达程序的行为。

第9章探讨了C# 3的第一个重要主题——Lambda表达式。语言的设计者并不满足于第5章介绍得已经相当精简的语法，而是将委托变得比在C# 2中还要容易创建。Lambda表达式还能做更多的事情——它们可以转换成表达式树：这是以数据形式来表示代码的一种强大的方式。

第10章探讨了扩展方法，它提供了一种完美的方式来欺骗编译器，使编译器相信在一个类型中声明的方法实际上从属于另一个类型。从表面上看，这似乎会带来可读性上的灾难，但经过仔细考虑之后，你就会发现它是一个相当强大的特性——而且对LINQ来说至关重要。

第11章以查询表达式的形式合并了前面3章的内容。查询表达式是一种简单而又强大的数据查询方式。我们先是将重点放在LINQ to Objects上，但所谓举一反三，通过观察如何应用查询表达式模式，你就明白LINQ to Objects能轻松地替换成其他数据提供器。

第12章简要介绍了LINQ的不同用法。首先，我们学习了LINQ to SQL如何将看似普通的C#转换为SQL语句，以此来展示查询表达式和表达式树相结合的好处。然后，以LINQ to XML为例，继续介绍了如何将库设计得能够与LINQ相契合。Parallel LINQ（并行LINQ）和Reactive Extensions（响应式扩展）展示了进程内查询的两种替代方法。本章最后讨论了如何用自己的LINQ操作符扩展LINQ to Objects。

从第13章开始介绍C# 4，包括命名实参和可选参数、COM互操作的改进和泛型可变性。在某种程度上，这些特性彼此互不相干，但COM互操作以及用于处理COM对象的其他功能，都受益于命名实参和可选参数。

第14章介绍了C# 4中最重要的特性：动态类型。用执行时的动态成员绑定代替编译时的静态绑定，对C#来说是一个巨大的尝试。但它在应用时是有选择性的：只有那些与动态值相关的代码才会动态地执行。

第15章介绍的是异步。C# 5只包含一个主要特性——编写异步函数的能力。这个特性非常复杂，不能一下子掌握，但掌握以后，使用起来非常优雅。

在接近尾声的第16章介绍了C# 5的其他特性（两个小特性），又展望了一下未来。

附录列出了所有的参考资料。附录A通过一些示例，介绍了LINQ标准查询操作符。附录B展示了核心的泛型集合类和接口。附录C简要介绍了.NET的不同版本，包括精简框架和Silverlight。

术语、排版约定和下载

本书大多数术语都会在出现时予以解释，但有的定义值得在这里重点讲一讲。我会相当明确地使用C# 1、C# 2、C# 3、C# 4和C# 5，但在其他书籍和网站中，你也许会看到C# 1.0、C# 2.0、C# 3.0、C# 4.0和C# 5.0这样的写法。我认为“.0”是多余的，因此省略了它们，希望意思清晰明确。

我从Mark Michaelis的一本C#书中借用了两个术语。runtime这个词有两个意思，一个是执行环境（如公共语言运行时），另一个是某个时间点（如“覆盖在运行时发生”）。为避免混淆，Mark换用“执行时”表示后一种概念，通常与“编译时”对应。这对我来说是一个很好的思路，我希望在社区中推广这种区分方法。本书从我做起，沿用了Mark的思路。

我经常会简单地说“语言规范”或者只是“规范”，除特别说明外，它们表示的是“C#语言规范”。但是，规范实际上有多个版本，一部分原因是由于语言本身有不同的版本，另一部分原因是标准化的过程如此。本书提到的规范的节序号都来自微软的《C# 5.0语言规范》。

本书有大量代码片段，它们是用等宽字体印刷的。代码清单的输出同样如此。有的代码清单添加了旁注，而且有时一些特定的代码段会加粗，以突出内容的变化、改进或增加。几乎所有代码都以代码段的形式出现，目的是保持精简，同时仍然可以在合适的环境中运行。那个环境就是

Snippy，这是1.8节将介绍的一个定制工具。Snippy可以从csharpindepth.com站点下载，本书所有代码也可以从此站点下载，有的是代码段形式，有的是完整的Visual Studio解决方案的形式，更多时候这两种形式都有。也可以从Manning出版社的网站manning.com/CSharpinDepthThirdEdition上下载。

Author Online和本书网站

购买本书，你可以访问由Manning出版社运行的一个内部论坛，可以在这里发表关于本书的评论，询问技术问题，并得到作者和其他用户的帮助。为了访问论坛并注册，请打开www.manning.com/CSharpinDepth ThirdEdition。网页上介绍了注册后进入论坛的方法，可以获得什么帮助，以及论坛的规则是什么。

只要书一出版，Author Online论坛和以前的讨论内容就可以从出版社的网站访问。

除了Manning出版社网站，我还为本书建立了一个配套网站，网址是www.csharpindepth.com，上面有许多不适合放到书中的内容和本书所有代码清单以及其他资源的链接。

目　录

第一部分　基础知识

第二部分　C# 2：解决C# 1 的问题