C#与F#编程实践

Real-World Functional Programming

With Examples in F# and C#

（捷）Tomas Petricek （英）Jon Skeet 著
贾洪峰 译

MANNING

清华大学出版社

C#与 F#编程实践

(捷)Tomas Petricek (英)Jon Skeet 著

贾洪峰 译

清华大学出版社

北 京

内 容 简 介

本书旨在介绍用于解决现实问题的函数式思想及相关技巧。尽管本书给出了很多用 F#和 C#两种语言编写的示例，但其重点在于阐述思想，而不是介绍 F#或 C#语言。

本书的主要内容包括：如何用函数式思维方式来思考问题；如何将面向对象编程与函数式编程结合起来；如何编写高效的 F#代码。要阅读本书，最好具备 OOP 和 C# 2.0 编程经验，不需要具备有关函数式编程或 F#的基础知识。

本书的适用对象是具备上述经验并希望了解函数式编程的.NET 开发人员。计算机专业的学生也可以通过阅读本书，了解函数式概念的现实应用。

Real-World Functional Programming With Examples in F# and C# by Tomas Petricek and Jon Skeet (ISBN: 978-1-933988-92-4)

图书在版编目(CIP)数据

C#与 F#编程实践/(捷克)佩特里切克(Petricek, T.)，(英)斯基特(Skeet, J.)著；贾洪峰译.
--北京：清华大学出版社，2011.10

书名原文：Real-World Functional Programming With Examples in F# and C#
ISBN 978-7-302-26890-1
Ⅰ. ①C…　Ⅱ.①佩…　②斯…　③贾…　Ⅲ. ①程序语言—程序设计　Ⅳ. ①TP312
中国版本图书馆 CIP 数据核字(2011)第 193666 号

责任编辑：文开琪　汤涌涛
封面设计：杨玉兰
版式设计：北京东方人华科技有限公司
责任校对：周剑云
责任印制：王秀菊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**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 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185×230　**印　张**：36　**字　数**：806 千字
版　　次：2011 年 10 月第 1 版　　**印　　次**：2011 年 10 月第1 次印刷
印　　数：1～3500
定　　价：79.00 元

产品编号：037496-01

译者序

曾经有一位学生把自己的求职简历拿给我看，在“计算机能力”一栏中写着“精通 C、C++、Delphi、Java 等多种编程语言”。我非常了解这位学生，知道他不是天才，知道他经过四年的学习，有着怎样的编程能力。同样是因为我对他非常了解，所以也知道他并不是一位夸夸其谈的学生。他一定是认为自己真的“精通”这些语言，所以才会在简历中郑重写下这样的文字。

我于是问他：“你认为怎样才算是‘精通’一门语言，而不只是‘会’一门语言？”他回答：“我认为对于一门语言，如果只知道它的语法，或者了解它的一些特有结构，能够读用这种语言编写的基本程序，能够帮助自己完成一些简单的任务，就可以说是‘会’这门语言。而要达到‘精通’的水平，需要掌握这种语言里所有的代码库类库情况，能够根据任务选择最佳程序包。”能有这样的回答，表明他对编程还是有一定的体会和积累的。但我想起 Alan Perlis 的那篇著名文章“Epigrams on Programming”[①]，其中第 19 句是：

如果一种编程语言不能影响你对编程的思考方式，就不值得学习。(A language that doesn't affect the way you think about programming, is not worth knowing.)

我们能不能这样理解：“在学习一门语言时，只有在改变你的编程方式，甚至是改变你分析问题、解决问题的思维方式之后，才真正算得上‘精通’这门语言。”本书就是这样一本帮你改变思维方式的编程书籍。

随着多核处理器的发展，对高可靠性应用程序的需求增加，函数式编程思想正在受到越来越多的关注。一方面是出现了新的函数式语言，如微软的 F#；另一方面，在一些传统的面向对象语言(如 C#)中，也增加了许多函数式特性。本书希望通过 F#和 C#两种语言编写的示例，深入浅出地讲解那些能够解决现实问题的函数式思想及相关技巧，使读者体会到函数式编程思想与当今流行的面向对象式编程有什么不同，是什么独特

① Perlis A J. Epigrams on Programming. SIGPLAN Notices, 1982, 17(9): 7-13.

思想让 F#成为一门真正的函数式语言，这些思想又是如何成功地解决 C#中的编程问题的。

本书的重点不是介绍 F#的语法结构，而是将函数式编程思想从这些具体的语言之中剥离出来，为读者提供一种不同的思维方式，提供一种不同的视角。这些思维方式和看待问题的视角，并不局限于编程领域，还有可能改变你对周围世界的思考方式与看法。

由于出版周期原因，有多位译者参与了本书的翻译工作。主体翻译及全书统稿工作由贾洪峰完成。其他参与部分章节翻译的人员有：刘建卓、时朋泉、刘建静、许津、刘建喜、刘阿娜、韩斌、武首香、张丽蘋、孙绛琳、时楚楚、王亮、王彦、王山花、缪素芬、郭辉、李强丽、李晓晓、吴士珍、翟海川等。

由于译者翻译水平与技术水平有限，译文之中存在偏差甚至错误之处在所难免，请广大读者不吝指出！

序

在过去的几十年里，面向对象编程在这个行业中占据着主导地位，这种方式能够将复杂性隐藏起来，提供结构和直观性，从而促进了软件开发的繁荣发展。

并不是所有类型的复杂性都适合采用经过封装的共享状态和虚方法之类的机制进行处理。对于一些计算、分析和变换领域，采用对象并没有太大帮助。目前，对于并发性的需求日益增强，这对面向对象式的范例产生了新的压力。

函数式与声明性技术最初隐约存在于学术机构和研究实验室中，为了应对挑战，慢慢渗透到主流语言中。例如，在 C#中，我们增加了泛型、匿名函数和单子(monadic)查询表达式。但是还缺少一种具有丰富库和工具的成熟商业编程语言，这些库和工具对于开发人员的生产效率是至关重要的。我相信 F#是一个重要的里程碑，是众多转折点之一，有一天我们回顾过去时，会说，我们就是在这一刻改变了发展方向。

F#在编程上超越了语言分支，充分包含.NET 框架的完整对象模型。但现在，我们必须对函数式编程和函数式编程语言进行区分。由于编程语言变为多范例(multi-paradigmatic)语言，所以，过去仅有唯一选择的内容(面向对象的或函数式的)，现在都演变出更多的选择机会。尽管在 F#中实现函数式编程技术可能更容易、更自然一些，但在 C#中也可以使用这些技术获益。在选择编程语言时可以有自己的偏好，但解决问题的可选方法仍然是开放性的。

本书的卓越之处在于，它将函数式思维方式从 F#语言本身中剥离出来。书中以 C#和 F#两种语言实现了函数式编程模式，读者应当会从中受益匪浅。那些希望使用 F#编写代码的开发人员，利用自己已经熟悉的语言来研究模式用法，一定可以事半功倍。而那些有充分的理由需要继续使用 C#的程序员，也可以从 F#示例中感受到原汁原味的原理。

函数式编程是一种思想状态。它是对问题的一种不同思考方式。它不只是一个饱含技术的百宝箱(尽管它的确是这样一个百宝箱)，还是一种视角，从这种视角来看，再困难

的问题也是有方法可循的。在本书中，Tomas 和 Jon 温和而又勇敢地坚持突出、强调深层次的原理，不过，本书依然具有很强的实用性。阅读本书时，一定要做好心理准备，你可能再也不会以过去的方式来思考自己的代码了！

Mads Torgersen

微软 C#项目经理

前言

本书不同于目前已有的许多其他编程书籍。它没有仅专注于某一种特定的编程语言或程序库，而是使用现有的语言和程序库来解释一种思维方式，这种思维方式的重要性日益增加，而且已经对许多新兴技术产生了影响。

读者可能已经了解了本书介绍的一些概念，因为函数式思想在许多技术中都有所体现。.NET 领域的例子包括 C# 3.0 和 LINQ 项目、Microsoft Parallel Extensions to .NET 以及 Windows Presentation Foundation (WPF)中所使用的声明性编程模型。在本书中，我们将在读者已有的.NET 和 C#经验基础上，解释函数式编程范例。我们将介绍微软新推出的函数式编程语言 F#，并用它将抽象思想变得更具体一些。如果可能，我们还会给出 C#示例，这是因为在设计 C#应用程序时，函数式思想也可以提供一些帮助。

如果我们要编写一本专门介绍 F#的书，可能会完全根据它的各个语言特征来组织篇章结构，并对这些特征进行逐一解释。但本书整体上是关于函数式编程的，其结构基础比较松散，也就是那些构成函数式范例的思想。这种组织方式要更困难一些，因为思想之间没有清晰界限，经常是相互重叠的。

我们尝试选出了一些思想，我们认为这些思想对于函数式编程入门者来说是最为重要的，并围绕着这些思想来编写本书。这一点在第 II 部分尤其重要，这一部分系统地研究了函数式、高阶函数和函数式程序的体系结构。这就意味着，有一些可用于快速演示函数式编程的例子(如处理数据集合的例子)，不会仅在书中出现一次，而是在介绍各个新思想之后，在若干章节中逐渐开发这些例子。我们之所以决定采用这一方法，是因为它表明了函数式编程是如何从一小组简单概念中优雅地发展而来的，就像函数式程序本身一样。

致谢

“蝴蝶效应”，这个由 Edward Lorenz 创造的词语是以混沌理论为基础的：一个非常微小的事件，例如在亚洲某个地方的蝴蝶扇动了一下翅膀，可能会引起一个非常巨大的事件，如南美洲发生一次飓风。在概念上，蝴蝶扇动翅膀是固定的，而位置(亚洲或巴西)和结果(飓风或龙卷风)是变化的。其实说一句话就足够了：有太多我无法在此一一列举的人(和蝴蝶)，没有他们，就不可能有本书的存在。而且，即使我不相信混沌理论，我希望在此提到的人数也是非常庞大的。

如果不是遇到 Don Syme，我可能永远都不会对 F#和函数式编程发生兴趣。Don 是我在微软研究院两个实习期内的导师，和他一起工作是一件非常愉快的事情，特别是在长时间地讨论 F#(还有生活、宇宙和其他各种话题)时。我还要感谢微软研究院的 James Margetson，他教给我许多很酷的函数式编程技巧。但是，如果我没有成为微软的 MVP(最有价值专家)，没有遇到 Luke Hoban，那我就不可能遇到 Don 和 James，是 Luke Hoban 后来把我介绍给 Don 的。如果像这样继续下去，我还要提到 Michal Bláha、Jan Stoklasa、Božena Mannová，他们是 CodeProject.com 的作者，当然还要提到其他许多人。

如果没有来自 Manning 的 Mike Stephens，没有 Harry Pierson，这本书也不会存在，Mike Stephens 是最早与我联系的，没有 Harry Pierson 的最初参与，我们可能永远都不会着手编写本书。尽管我们一起合作的时间很短暂，但 Harry 的参与非常重要，而且使我备受鼓舞。

没有前面提到的这些人，就不可能开始编写本书，以下要提到的这些人，没有他们就不可能完成本书。

我非常感谢我的合著者 Jon，感谢他帮助我度过了将初稿和草图转变为真正书籍的漫长过程。如果读者感受到这本书就是为你而写的，那也应当归功于 Jon，因为他从读者的角度发出，将所有内容都调整为正确形式。最后还要说一声，Jon 是一个非常出色的合作者，无论是通过网络，还是当面和他讨论本书，都是一件非常愉快的事情。

现在，我要感谢 Manning 为本书做出贡献的每一个人。我已经提到了 Mike Stephens，他总是能够雪中送炭。Nermina Miller 和 Tara McGoldrick Walsh 对我在日常编写过程中遇到的问题提供总体的指导性意见，而 Mary Piergies 和 Liz Welch、Elizabeth Martin 一起，帮助我找到具体解决方法。我还和 Manning 的其他许多优秀员工一起短暂合作过，他们是 Gabriel Dobrescu、Steven Hong、Dottie Marsico、Christina Rudloff、Gordan Salinovic、Maureen Spencer 和 Karen Tegtmeyer。我还要感谢出版商 Marjan Bace，他提供了无数非常有用的真知灼见。

Manning 的人们还有一双幸运之手，他们在编写过程中的不同阶段，挑选出正确的人选进行评论。我们收到大量的评论、建议和提示，还有相当数量的正面反馈，这些反馈意见鼓励我尽可能多地考虑这些建议。除了我们的匿名评论者之外，我还要感谢我们的两位技术评论者——Matthew Podwysocki 和 Michael Giagnocavo。我过去也曾经担任过技术评论者，所以我非常了解他们的艰苦工作！还要特别感谢为本书作序的 Mads Torgersen。

另外一群对本书提供宝贵建议的人们是早期草稿的读者们。首先是我的同事 Jan Stoklasa 和 René Stein，还有那些通过“Manning 提前获取计划”(Manning Early Access Program)购买本书并在论坛中共享反馈的人们(如 Dave Novick、Peer Reynders、Vladimir Kelman 和 Michiel Borkent 等人)。其他一些对本书成文有所帮助的评价者包括：Marius Bancila、Freedom Dumlao、Eric Swanson、Walter Myers、Keith J. Farmer、Adam Tacy、Marc Gravell、Jim Wooley、Alessandro Gallo、Lester Lobo、Massimo Perga、Andrew Siemer、 Austin Ziegler、Dave McMahon、Jason Jung、Joshua Gan、Keith Hill、Mark Needham、Mark Ryall、Mark Seemann、Paul King 和 Stuart Caborn。

当然，我还要感谢我的朋友和家庭。他们可能并不知道，虽然他们的问题“你的书什么时候完成啊？”可能听起来不像在鼓励，但我非常了解他们，我衷心地感谢他们别样的鼓励。最后但绝非最不重要的一个，非常感谢我最亲爱的 Evelina，她不仅提供了非常宝贵的精神支持，还如此耐心地阅读和评论了大部分手稿。

Tomas Petricek

我首先要感谢 Tomas 和 Manning 的每一个人，使我有机会参与本书的撰写。能够亲身参与，尽绵薄之力，是非常有趣的，一方面从一本书中学习函数式编程，同时又将自己学到的东西融入一本书中，这一切让我乐在其中。我只希望我所做的微薄贡献能够有所帮助——我主要是充当一个富有热情但又无知的读者(当然还是一个 C#狂热者)，所

以在某些方面，您现在读到的这本书经过了一些调整，是用来教我学习函数式编程的。对我而言，它是一份值得感恩的礼物。Tomas 已经向 Manning 的所有编辑和其他员工表示了感谢，我希望再一次向他们表示谢意。

我的孩子们还太小，不能开始编程，而我的妻子则太……怎么说呢？大体来说就是太正常了吧——但当我疯狂工作时，我的孩子和妻子总是伴我左右，给我支持。我总是努力使自己保持头脑清醒，能够同时处理一本以上的书，而我的妻子(她撰写童话)似乎总是在为各种不同书目和出版社忙碌着计划书、详细目录、初稿、审稿、校稿和已提交的手稿。而且，她还嫁给了我——所有人都奇怪她是如何保持头脑清醒的。但是，我非常高兴她做到了，我要感谢她成为现在这样一个人。Tom、Robin 和 William 都很喜欢技术，在这方面大有前途，但当我从办公室回到家中时，最喜欢的还是他们的微笑和拥抱。

最后，我还要感谢我的所有英语老师，特别是 Simon Howells。我对编程语言的了解越多，我就越相信软件工程师应当最为关注的语言就是他用来与人们进行交流的语言，而不是计算机语言。就像我对计算充满激情一样，Simon Howells 对语言和文学也充满着激情，这种激情给他的学生们留下了深刻印象。他很可能从来都不会阅读我著作中的一个单词，但他的教导将陪伴我一生。

Jon Skeet

关于本书

如果您是一位.NET 开发人员，熟悉面向对象的技术，希望了解这场新的“函数式编程”运动都是关于什么内容的，还想知道如何从中获益，那这本书绝对是为您而写的。

本书是专门为那些拥有面向对象编程和 C# 2.0 工作经验的.NET 开发人员定制的。当然，一般来说，读者不需要了解函数式编程，甚至不需要了解 F#。事实上，如果已经习惯采用面向对象的方式来思考问题，那么在学习函数式编程时的难度会更大一些，因为许多函数式思想会显得非常陌生。我们在编写本书时已经考虑到这一点，所以在解释特定主题时，我们经常会从读者角度思考，并对 OOP 和函数式编程进行对比。

如果您是一位使用其他语言和工具(例如 Java、Python 或者 Ruby)的面向对象程序员，能够快速理解语言，那么也可以从本书中获益。虽然我们给出的例子是采用 C#编写的，但其中有许多例子在其他面向对象式语言中都是非常相似的。对于其他语言中所不存在的 C# 3.0 功能，本书给出了简要解释，所以不用担心会摸不到头脑。

本书的核心不是函数式编程的学术问题，但如果您是一位计算机科学系的学生，正在学习相关课程，也可以阅读本书，从中了解函数式概念的现实应用。

本书包含哪些主题

如果您还在犹豫本书是否适合自己，可以先看看本书包含的内容。

- 函数式编程概念。在阅读本书时，您将学习一种新的问题思考方式，将会看到复杂的面向对象设计模式如何在函数式编程中变为一个简单的概念。在使用任意一种语言进行编程时，都可以采用函数式思维方式，所以无论您使用哪种特定技术，它都是有用的。
- 实践中使用的基本函数式结构。现在，在 C# 3.0 中已经可以使用这些结构中的一部分，所以在纯粹的函数式 F#实施方式旁边，还会给出许多其他示例：用我们熟悉的 C#代码编写而成。我们还将解释 C# 3.0 中新增加的一些功能，以及它

们与函数式思想的联系。深入理解这些概念，可以帮助您从这些新功能中获取最大利益。

- 编写实用 F#代码。尽管本书并不是专门介绍 F#的，但仍然会教给您足够的知识，供您开始应用 F#。我们还将研究一些能够让 F#真正发挥实力的领域，包括异步编程和创建可组合库等。

我们并不会宣称本书在所有方面都是完美的：计算领域不存在“一体适万用”的良方。应当知道，本书并不奢望让所有读者都满意。

本书不涉及哪些内容

本书的编写目标不是作为一本参考书。如果您还不熟悉函数式编程，阅读本书的最佳方式就是从头开始，然后按各章顺序阅读。我们假定读者会这样阅读本书，所以后面的章节经常引用前面已经解释过的概念。如果从本书中间开始阅读，可能会发现理解起来比较困难。

本书也不是 F#编程的快速指南。要想成为一位优秀的 F#程序员，唯一的方法就是理解函数式编程思想。您可以学习所有 F#关键字，并用 F#重写 C#代码，但这样不会有太多好处。如果希望编写出合乎惯例的 F#代码，那就需要学习函数式编程中所使用的一种不同思考方式。而这正是可以通过阅读本书学到的东西。在习惯了函数式思维方式之后，可以阅读其他书籍来帮助您更深入地学习 F#。

我们的主要目的是编写一本书，可以帮助专业程序员编写一些解决方案，以解决现实世界的业务问题。但是，这并不意味着我们会为您提供一种现成的解决方案，用来解决您眼前需要解决的问题，而是主要介绍相关概念和技术。我们用许多例子来说明这些原理，但不可能涵盖 F#和函数式编程的全部适用领域。

路线图

本书采用了一种重复式结构。在第 I 部分(第 1 至 4 章)，我们将解释最重要主题中的一些内容，使您可以体会到学习的动力，并理解是什么使函数式编程不同于其他编程方式。第 II 部分(第 5 至 8 章)系统地讨论了函数式编程的基础。在第 III 部分(第 9 至 12 章)，我们将在此基础上，讨论函数式编程、优化的最佳实践，还要讨论大多数函数式程序员偶尔会用到的高级技巧。第 IV 部分(第 13 至 16 章)给出了一些更为复杂的例子，展示如何使用函数式编程来开发更大的现实世界项目。

第 1 章讨论了是什么原因使函数式概念变得日益重要。它给出一些现有技术的例子，您对这些技术可能已经有所了解，而且这些技术已经从函数式编程的某些方面获益。

这一章还给出了用 F#编写的第一个示例应用程序。

第 2 章介绍了函数式编程背后的概念。它没有给出任何细节，而且主要使用 C#，从而可以帮助读者理解这些概念之间的相互关系，以及它们对于程序结构意味着什么。

第 3 章终于给出了一些实际函数代码。它介绍了 F#中用到的一些数据类型，例如元组和列表。我们还会看到如何处理 F#中的类型，但也会用 C#来实现它们，以解释它们是如何工作的。这一章介绍了将函数作为值来使用的思想，它对于函数式编程是非常重要的。

第 4 章给出了第一个用 F#实现的现实世界应用程序。我们将使用各种.NET 和 F#库来实现一个用于绘制饼图的程序。在该章还将看到如何在开发过程中有效使用 F#所提供的工具。

第 5 章对值进行了讨论。函数式程序被编写为一些计算式，它们取得参数值，返回结果值，因此，很容易就可以看出，为什么在开始系统地评论函数式功能时，必须首先研究各种各样的值。

第 6 章描述了处理值的最常用方式，也就是使用高阶函数。直接对值进行处理，经常需要大量的重复代码，所以本章介绍了如何设计和实施可重复使用的操作。

第 7 章将注意力转移到体系结构方面。函数式应用程序的结构由其要处理的数据确定。我们将使用一个管理和绘制简单文档的应用程序来说明这一重要原理。

第 8 章主要关注一些应用程序的体系结构，这些应用程序需要在运行时动态改变其行为。这一功能可以使用函数来实现，所以我们将详细讨论它们，还将解释诸如闭包之类的相关主题。

第 9 章说明如何在 F#中混合使用面向对象式风格和函数式风格。它说明在编写函数式.NET 库时，如何将函数式功能(例如不变性)与面向对象概念(如封装)结合使用。

第 10 章主要介绍正确性和效率。我们将看到如何编写能够处理任意大小数据集的函数，以及如何高效率地编写这些函数。您还会学习如何使用诸如数组之类的命令式结构来优化代码。

第 11 章讨论重构、测试和延迟。采用函数式编程时，对已有代码的理解和改进将变得更为容易，在本章将解释是如何做到的。我们将研究单元测试，了解如何因为具有了可组合性和严格性而不再需要进行某些种类的测试。

第 12 章开始展示如何处理数据集合。我们将介绍 F#序列表达式，它是专门为这一目的

设计的。您还会看到它不是一个内置功能，这一点不同于 C#中与它最接近的对应内容，序列表达式是一种可以改变代码含义的更通用功能。

第 13 章展示了一种在 F#中处理数据时的常见场景。它首先使用一种公共 Web 服务下载数据，然后将其解析为一种结构化格式。最后将会看到如何使用 Excel 对其中有意义的内容进行可视化。

第 14 章介绍如何利用函数式概念来生成易于并行处理的代码。该章使用一个图像处理应用程序和一个模拟来进行展示，在这个模拟中存在一些动物和猎取这些动物的捕食者。

第 15 章介绍如何构建声明性函数库。该章表明，可以优美地组合那些经过精心设计的库。在示例中，我们将看到如何创建一个生成动画的库和一个表示经济合同的库。

第 16 章介绍如何生成 GUI 应用程序，笼统而言是如何生成由外部事务驱动的程序。在其他语言中，要实现诸如此类的控制流是非常困难的，我们将学习一些技巧，利用这些技巧，可以非常容易地在 F#中实现它。

印刷约定

本书包含大量代码示例，它们采用等宽字体排版。较长示例都展示在带有标题的代码清单中。由于本书混合采用了 C#和 F#，所以，在代码清单中，其标题还指出了所使用的语言。在展示用 F#编写的代码时，我们在两种代码清单形式之间进行了区分。标有“F#”的代码是无格式源代码，可以作为整体进行编译。而标有“F# Interactive”的代码清单给出了以 F#编写、输入到交互式外壳(shell)的代码段。由该外壳生成的输出使用斜体。在所有代码清单中，所有 C#和 F#关键字都使用等宽字体的粗体突出显示。

命名约定

在本书中，我们不仅混合使用了两种语言，还将函数式编程惯例与面向对象式惯例混合在一起。由于我们希望使用两种语言的自然风格，所以必须遵循 F#和 C#中的不同命名约定。

在 C#中，我们遵循通常的.NET 风格。在 F#中，我们在开发类或者编写可以从其他.NET 语言访问的组件时，也毫无例外地使用这种符号体系。而在展示仅作为私有实现的 F#代码时，我们将遵循函数式命名风格。更特别的是，对于变量和函数名称均采用 camelCase 风格。这是 F#中的标准风格，因为函数声明基本上与变量声明相同。

偶尔，我们会使用较短的名称和缩写，其中有两个原因。第一，函数式程序员经常使

用这种风格。有了更好的 IDE，使用这一风格的理由就比较少了，所以我们尽量减少它的使用。在某些情况下，较短名称是函数式编程的常见术语，所以我们还是保留了它。第二，有时我们会并排使用两个示例，这就意味着我们必须使用一种更紧凑的编码风格。在其他情况下，大多数命令遵循.NET 风格，也有几种例外，在正文中将对其进行讨论。

StyleCop 和 FxCop

如果读者熟悉诸如 StyleCop 和 FxCop 之类的代码分析工具，可能了解本书中的代码是否符合这些工具所要求的规则。我们遵循了大多数(但并非全部)通常的.NET 约定。如果通过这些工具来运行代码，会产生大量警告信息，其中同样有两个原因。

- 这些代码分析工具是为面向对象式语言开发的，而这些语言使用的命名规则与风格是符合面向对象惯例的。在本书中将会了解到，函数式世界在许多方面是有所不同的，我们不能遵循所有的面向对象式原则。F#是非常成功的，就是因为它非常不用于 C#和 Visual Basic。从语言关键字中看不出什么，但在它所采用的整体编程风格方面可以明显表现出来，在命名约定方面也有所体现，这种命名约定使代码更为紧凑。
- 本书的篇幅有限。我们使用源代码示例来演示重要的思想，所以我们不希望包含一些对于这些讨论并不重要而只是为了使代码符合约定而产生的干扰性内容。

源代码下载

本书示例的源代码可从出版商的网站上联机下载：http://www.manning.com/RealWorldFunctionalProgramming，也可以从作者创建的代码库中下载：http://code.msdn.microsoft.com/realworldfp。

作者在线

购买本书之后，还可以免费访问由 Manning 运转的一个专用 Web 论坛，在这里可以对本书进行评论，询问技术问题，并接受来自作者及其他用户的帮助。要访问该论坛并订阅它，可在 Web 浏览器中打开 http://www.manning.com/RealWorldFunctionalProgramming。该网页将提供一些信息，告诉您在注册之后如果进入该论坛，可以获得哪些方面的帮助，以及该论坛中的一些行为规则。

Manning 对读者的承诺是提供一种场合，供各位读者之间、读者与作者之间进行有意义的对话。关于作者的参与程度问题，并不做出承诺，作者们对本书论坛的贡献是自愿

的(也是没有报酬的)。我们建议读者询问一些富有挑战性的问题，以免作者失去留在这里的兴趣!

在本书付印时，“作者在线”论坛及前面讨论的归档文件，都可以从出版商的网站上下载。

其他在线资源

除了 Manning 的网站(http://www.manning.com/Real-WorldFunctionalProgramming)之外，我们还为本书创建了一个配套网站：http://functional-programming.net。其中包含了一些可能对读者有用的信息、各章的源代码以及没有收入本书中的内容。这个网页还链接到最近发表的一些与函数式编程有关的文章，读者可以阅读这些内容，了解有关这一主题的更多内容。

如果您对 F#感兴趣，可能还希望查看微软开发人员中心的官方网站：http://msdn.microsoft.com/fsharp。其中包含有关该语言的最新信息，以及指向文章、视频或其他 F#资源的链接。如果希望询问有关 F#的问题，并确保会得到满意的回答，请访问 F#社区论坛：http://cs.hubfs.net。