经典原版书库

C# 7.0 本质论

（英文版）

Essential C# 7.0

[美] 马克·米凯利斯（Mark Michaelis）著

机械工业出版社
China Machine Press

出版者的话

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson、McGraw-Hill、Elsevier、MIT、John Wiley & Sons、Cengage 等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出 Andrew S. Tanenbaum、Bjarne Stroustrup、Brian W. Kernighan、Dennis Ritchie、Jim Gray、Afred V. Aho、John E. Hopcroft、Jeffrey D. Ullman、Abraham Silberschatz、William Stallings、Donald E. Knuth、John L. Hennessy、Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

序

本书是C#最权威、最值得尊重的参考书之一，作者为此付出了非凡的努力！ Mark Michaelis的《 Essential C# 》系列多年来一直是畅销经典。而我刚认识Mark的时候，这本书还处于萌芽阶段。

2005年LINQ（语言集成查询，Language Integrated Query）公布时，我才刚加入微软公司，正好见证了PDC会议上令人激动的公开发布时刻。虽然我对技术本身几乎没有什么贡献，但它的宣传造势我可是全程参加了。那时人人都在谈论它，宣传小册子满天飞。那是C#和.NET的大日子，至今依然令人难忘。

但会场的实践实验室区域却相当安静，那儿的人可以按部就班地试验处于预览阶段的技术。我就是在那儿遇见Mark的。不用说，他一点儿都没有按部就班的意思。他在做自己的试验，梳理文档，和别人沟通，忙着鼓捣自己的东西。

作为C#社区的新人，我感觉自己在那次会议上见到了许多人。但老实说，当时太混乱了，我唯一记得清的就是Mark。因为当问他是否喜欢这个新技术时，他不像别人那样马上开始滔滔不绝，而是非常冷静地说：“还不确定，要自己搞一搞才知道。”他希望完整地理解并消化一种技术，之后才将自己的想法告知于人。

所以我们之间没像我本来设想的那样发生一次快餐式的对话。相反，我们的对话相当坦诚、颇有营养。像这样的交流好多年都没有过了。新技术的细节、造成的后果和存在的问题全都涉及了。对我们这些语言设计者而言，Mark是最有价值的社区成员。他非常聪明，善于打破砂锅问到底，能深刻理解一种技术对于真正的开发人员的影响。但是，最根本的原因可能还是他坦诚，他从不惧怕说出自己的想法。一样东西通过了Mark的测试，就没什么好担心的了！

这些特质也使Mark成为一名出色的作家。他的文字直指技术的本质，敏锐地指出技术的真正价值和问题，向读者提供最完整的信息且没有废话。没人能像这位大师一样帮你正确理解C# 7.0。

请好好享用本书！

——Mads Torgersen，微软公司C#项目经理

前　言

在软件工程的发展历史中，用于编写计算机程序的方法经历了几次思维模式的重大转变。每种思维模式都以前一种为基础，宗旨都是增强代码的组织，并降低复杂性。本书将带领你体验相同的思维模式转变过程。

本书开始几章会指导你学习顺序编程结构。在这种编程结构中，语句按编写顺序执行。该结构的问题在于，随着需求的增加，复杂性也指数级增加。为降低复杂性，将代码块转变成方法，产生了结构化编程模型。在这种模型中，可以从一个程序中的多个位置调用同一个代码块，不需要复制。但即使有这种结构，程序还是会很快变得臃肿不堪，需进一步抽象。所以，在此基础上人们又提出了面向对象编程的概念，这将在第 6 章开始讨论。在此之后，你将继续学习其他编程方法，比如基于接口的编程和 LINQ（以及它促使集合 API 发生的改变），并最终学习通过特性（attribute）进行初级的声明性编程（第 18 章）。

本书有以下三个主要职能。

- 全面讲述 C# 语言，其内容已远远超过了一本简单的教程，为你进行高效率软件开发打下坚实基础。
- 对于已熟悉 C# 的读者，本书探讨了一些较为复杂的编程思想，并深入讨论了语言最新版本（C# 7.0 和 .NET Framework 4.7/.NET Core 2.0）的新功能。
- 它是你永远的案头参考——即便在你精通了这种语言之后。

成功学习 C# 的关键在于，要尽可能快地开始编程。不要等自己成为一名理论“专家”之后才开始写代码。所以不要犹豫，马上开始写程序吧。作为迭代开发思想的追随者，我希望即使一名刚开始学习编程的新手，在第 2 章结束时也能动手写基本的 C# 代码。

许多主题本书没有讨论。你在本书中找不到 ASP.NET、ADO.NET、Xamarin、智能客户端开发以及分布式编程等主题。虽然这些主题与 .NET 有关，但它们都值得用专门的书分专题讲述。幸好市面上已经有丰富的图书供读者选择。本书重点在于 C# 及基类库中的类型。读完本书之后，你在上述任何领域继续深入学习都会有游刃有余的感觉。

本书面向的读者

写作本书时，我面临的一个挑战是如何在持续吸引高级开发人员眼球的同时，不因使用 assembly、link、chain、thread 和 fusion 等字眼而打击初学者的信心，否则许多人会以为这是一本讲冶金而不是程序设计的书。本书的主要读者是已经有一定编程经验，并想多学一种语言来“傍身”的开发者。但我还是小心地编排了本书的内容，使之对各种层次的开发者都有足够大的价值。

- **初学者：**假如你是编程新手，本书将帮助你从入门级程序员过渡为 C# 开发者，消除以后在面临任何 C# 编程任务时的害怕心理。本书不仅要教会你语法，还要教你养成良好的编程习惯，为将来的编程生涯打下良好基础。
- **熟悉结构化编程的程序员：**学习外语最好的方法就是“沉浸法”。类似地，学习一门计

算机语言最好的方法就是在动手中学习，而不是等熟知了它的所有“理论”之后再动手。基于这个前提，本书最开始的内容是那些熟悉结构化编程的开发者很容易上手的。到第 5 章结束时，这些开发者应该可以开始写基本的控制流程序。然而，要成为真正的 C# 开发者，记住语法只是第一步。为了从简单程序过渡到企业级开发，C# 开发者必须熟练从对象及其关系的角度来思考问题。为此，第 6 章的“初学者主题”开始介绍类和面向对象开发。历史上的 C、COBOL 和 FORTRAN 等结构化编程语言虽然仍在发挥作用，但作用会越来越小，所以，软件工程师们应该逐渐开始了解面向对象开发。C# 是进行这一思维模式转变的理想语言，因为它本来就是基于“面向对象开发”这一中心思想来设计的。

- **熟悉“基于对象”和“面向对象”理念的开发者：**C++、Python、TypeScript、Visual Basic 和 Java 程序员都可归于此类。对于分号和大括号，他们可是一点儿都不陌生！简单浏览一下第 1 章的代码，你会发现，从核心上讲，C# 类似于你熟知的 C 和 C++ 风格的语言。
- **C# 专家：**对于已经精通 C# 的读者，本书可供你参考不太常见的语法。此外，对于在其他地方强调较少的一些语言细节以及微妙之处，我提出了自己的见解。最重要的是，本书提供了编写可靠和易维护代码的指导原则及模式。在你教别人学 C# 时，本书也颇有助益。从 C# 3.0 到 C# 7.0 最重要的一些增强包括：
  - 字符串插值（第 2 章）
  - 隐式类型的变量（第 3 章）
  - 元组（第 3 章）
  - 模式匹配（第 4 章）
  - 扩展方法（第 6 章）
  - 分部方法（第 6 章）
  - 泛型（第 12 章）
  - Lambda 语句和表达式（第 13 章）
  - 表达式树（第 13 章）
  - 匿名类型（第 15 章）⊖
  - 标准查询操作符（第 15 章）
  - 查询表达式（第 16 章）
  - 动态编程（第 18 章）
  - 用任务编程库（TPL）和 async 进行多线程编程（第 19 章）
  - 用 PLINQ 进行并行查询处理（第 19 章）
  - 并发集合（第 20 章）

考虑到许多人还不熟悉这些主题，本书围绕它们展开了详细的讨论。涉及高级 C# 开发的还有“指针”这一主题，该主题将在第 21 章讨论。即使是有经验的 C# 开发者，也未必能很透彻地理解这一主题。

⊖ 英文原书此处有误，这里已根据正文内容（匿名类型包括在第 15 章中）修改。——编辑注

本书特色

本书是语言参考书，遵循核心《C# 语言 7.0 规范》(C# Language 7.0 Specification)。为了帮助读者理解各种 C# 构造，书中用大量例子演示了每一种特性，而且为每个概念都提供了相应的指导原则和最佳实践，以确保代码能顺利编译，避免留下隐患，并获得最佳的可维护性。

为增强可读性，所有代码均进行了特殊格式处理，而且每章内容都用思维导图来概括。

C# 设计规范

本书新版本最重大的改进之一就是增加了大量“设计规范”，下面是取自第 17 章的例子。

> 设计规范
> 要确保相等的对象有相等的散列码。
> 要确保对象在散列表中时散列码永不变化。
> 要确保散列算法快速生成良好分布的散列码。
> 要确保散列算法在任何可能的对象状态中的健壮性。

区分知道语法的程序员和能因地制宜写出最高效代码的专家的关键就是这些设计规范。专家不仅能让代码通过编译，还会遵循最佳实践，降低出现 bug 的概率，并使代码的维护变得更容易。设计规范强调了一些关键原则，开发时务必注意。

示例代码

本书大多数代码都能在公共语言基础结构（Common Language Infrastructure，CLI）的任何实现上运行，但重点还是 Microsoft .NET Framework 和 .NET Core 这两个实现。很少使用平台或厂商特有的库，除非需要解释只和那些平台相关的重要概念（例如，解释如何正确处理 Windows 单线程 UI）。

下面是一个示例代码清单。

代码清单 1-19　注释代码

```
class Comment Samples
{
  static void Main()
  {

      string firstName; //Variable for storing the first name
      string lastName;  //Variable for storing the last name

      System.Console.WriteLine("Hey you!");

      System.Console.Write /* No new line */ (
          "Enter your first name: ");
      firstName = System.Console.ReadLine();
```

```
        System.Console.Write /* No new line */ (
            "Enter your last name: ");
        lastName = System.Console.ReadLine();

        /* Display a greeting to the console
           using composite formatting. */

        System.Console.WriteLine("Your full name is {0} {1}.",
            firstName, lastName);
        // This is the end
        // of the program listing
    }
}
```

下面解释具体的格式：

- 注释使用斜体。

```
/* Display a greeting to the console
   using composite formatting */
```

- 关键字加粗。

```
static void Main()
```

- 有的代码突出显示，是为了指出这些代码与之前的有区别，或是为了演示正文介绍的概念。

```
System.Console.WriteLine(valerie);
miracleMax = "It would take a miracle.";
System.Console.WriteLine(miracleMax);
```

突出显示的可能是一整行，也可能是一行中的几个字符。

```
System.Console.WriteLine(
    "Your full name is {0} {1}.", firstName, lastName);
```

- 省略号表示无关代码已省略。

```
// ...
```

- 代码清单后列出了对应的控制台输出。由用户输入的内容加粗。

输出 1-7

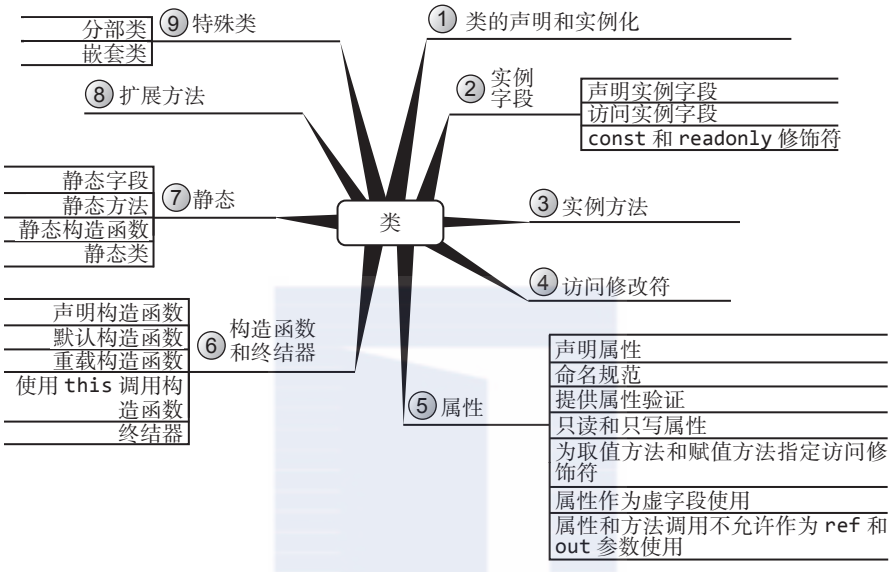
```
Hey you!
Enter your first name: Inigo
Enter your last name: Montoya
Your full name is Inigo Montoya.
```

虽然我也可以在书中提供完整代码以方便复制，但这样会分散大家的注意力。因此，你需要在自己的程序中修改示例代码。书中的代码主要省略了错误检查，比如异常处理。另外，代码没有显式包含 using System 语句，所有例子都需要该语句。

请访问 *https://github.com/IntelliTect/EssentialCSharp* 或 *http://bookzhou.com* 下载示例代码。

思维导图

每章开头都有一幅“思维导图”作为提纲，目的是为读者提供针对每章内容的快速参考。下面是一个例子（摘自第 6 章）。



每章主题显示在思维导图的中心，高级主题围绕中心展开。利用思维导图，读者可方便地搭建自己的知识体系，可以从一个主题出发，更清楚地理解其周边的各个具体概念，避免中途纠缠于一些不相干的枝节问题。

分类解说

根据编程水平的不同，可以利用书中的标志来帮助自己轻松找到适合自己的内容。

- **初学者主题**：特别针对入门级程序员提供的定义或解释。
- **高级主题**：可以让有经验的开发者将注意力放在他们最关心的内容上。
- **标注**：用有底纹的标注框强调关键点，引起读者的注意。
- **语言对比**：分散在正文中的补充内容描述了 C# 和其他语言的关键差异，为熟悉其他语言的读者提供指引。

本书内容组织

总体来说，软件工程的宗旨就是管理复杂性。本书基于该宗旨来组织内容。第 1 章 ~ 第 5 章介绍结构化编程，学习这些内容后，可以立即开始写一些功能简单的代码。第 6 章 ~ 第 10 章介绍 C# 的面向对象构造，新手应在完全理解这几章的内容之后，再开始接触本书其余部分更高级的主题。第 12 章 ~ 第 14 章介绍更多用于降低复杂性的构造，讲解当今几乎所有程序都要用到的通用设计模式。理解了它们之后，才可以更轻松地理解如何通过反射和特性

来进行动态编程。后续章节将广泛运用它们来实现线程处理和互操作性。

本书最后专门用一章（第 22 章）讲解 CLI。这一章在开发平台的背景下对 C# 语言进行了描述。之所以要放到最后，是因为它非 C# 特有，且不涉及语法和编程风格问题。不过，本章适合在任何时候阅读，或许最恰当的时机是在阅读完第 1 章之后。

下面是每一章的内容提要。（加黑的标题表明那一章含有 C# 6.0 和 C# 7.0 的内容。）

- **第 1 章——C# 概述**：本章在展示了用 C# 写的 HelloWorld 程序之后对其进行细致分析。目的是让读者熟悉 C# 程序的“外观和感觉”，并理解如何编译和调试自己的程序。另外，还简单描述了执行 C# 程序的上下文及其中间语言（intermediate language，IL）。
- **第 2 章——数据类型**：任何有用的程序都要处理数据，本章介绍了 C# 的基元数据类型。
- **第 3 章——更多数据类型**：本章深入讲解数据类型的两大类别：值类型和引用类型。然后讲解了可空修饰符以及 C# 7.0 引入的元组。最后深入讨论了基元数组结构。
- **第 4 章——操作符和控制流**：计算机最擅长重复性操作，为利用该能力，需知道如何在程序中添加循环和条件逻辑。本章还讨论了 C# 操作符、数据转换和预处理器指令。
- **第 5 章——方法和参数**：本章讨论了方法及其参数的细节，其中包括通过参数来传值、传引用和通过 `out` 参数返回数据。C# 4.0 新增了默认参数，本章将解释如何使用。
- **第 6 章——类**：前面已学过类的基本构成元素，本章合并这些构造，以获得具有完整功能的类型。类是面向对象技术的核心，它定义了对象模板。
- **第 7 章——继承**：继承是许多开发者的基本编程手段，C# 更是提供了一些独特构造，比如 new 修饰符。本章讨论了继承语法的细节，其中包括重写（overriding）。
- 第 8 章——接口：本章讨论如何利用接口来定义类之间的“可进行版本控制的交互契约”（versionable interaction contract）。C# 同时包含显式和隐式接口成员实现，可实现一个额外的封装等级，这是其他大多数语言所不支持的。
- **第 9 章——值类型**：尽管不如定义引用类型那么频繁，但有时确有必要定义行为和 C# 内置基元类型相似的值类型。本章介绍如何定义结构（struct），同时也强调其特殊性。
- **第 10 章——合式类型**：本章讨论了更高级的类型定义，解释如何实现操作符，比如 + 和转型操作符，并描述如何将多个类封装到一个库中。此外，还演示了如何定义命名空间和 XML 注释，并讨论如何基于垃圾回收机制来设计令人满意的类。
- **第 11 章——异常处理**：本章延伸讨论第 5 章引入的异常处理机制，描述了如何利用异常层次结构创建自定义异常。此外，还强调了异常处理的一些最佳实践。
- 第 12 章——泛型：泛型或许是 C# 1.0 最缺少的功能。本章全面讨论自 2.0 引入的泛型机制。此外，C# 4.0 增加了对协变和逆变的支持，本章将在泛型背景中探讨它们。
- 第 13 章——委托和 Lambda 表达式：正因为委托，才使 C# 与其前身语言（C 和 C++ 等）有了显著不同，它定义了在代码中处理事件的模式。这几乎完全消除了写轮询例程的必要。Lambda 表达式是使 C# 3.0 的 LINQ 成为可能的关键概念。通过学习本章，你将知道 Lambda 表达式是在委托的基础上构建起来的，它提供了比委托更优雅和简洁的语法。本章内容是第 14 章讨论的新的集合 API 的基础。本章还强调了匿名方法应该用

新的 Lambda 表达式代替。

- 第 14 章——事件：封装起来的委托（称为事件）是公共语言运行时（Common Language Runtime，CLR）的核心构造。
- 第 15 章——支持标准查询操作符的集合接口：通过讨论新的 Enumerable 类的扩展方法，介绍 C# 3.0 引入的一些简单而强大的改变。Enumerable 类造就了全新的集合 API，即“标准查询操作符”，本章对其进行详细讨论。
- **第 16 章——使用查询表达式的 LINQ**：如果只使用标准查询操作符，会形成让人难以辨认的长语句。查询表达式提供了一种类似 SQL 风格的语法，有效解决了该问题。本章会详细讨论这种表达式。
- **第 17 章——构建自定义集合**：构建用于操纵业务对象的自定义 API 时，经常需要创建自定义集合。本章讨论了具体做法，还介绍了能使自定义集合的构建变得更简单的上下文关键字。
- **第 18 章——反射、特性和动态编程**：20 世纪 80 年代末，程序结构的思维模式发生了根本性的变化，面向对象的编程是这个变化的基础。类似地，特性（attribute）使声明性编程和嵌入元数据成为可能，因而引入了一种新的思维模式。本章探讨了特性的方方面面，并讨论了如何通过反射机制来获取它们。本章还讨论了如何通过基类库（Base Class Library，BCL）中的序列化框架来实现文件的输入输出。C# 4.0 新增了 dynamic 关键字，能将所有类型检查都移至运行时进行，因而极大地扩展了 C# 的能力。
- **第 19 章——多线程处理**：大多数现代程序都要求用线程执行长时间运行的任务，同时确保对并发事件的快速响应。随着程序越来越复杂，必须采取其他措施来保护这些高级环境中的数据。多线程应用程序的编写比较复杂。本章讨论了如何操纵线程，并提供一些最佳实践来避免将多线程应用程序弄得一团糟。
- **第 20 章——线程同步**：本章以第 19 章为基础，演示如何利用一些内建线程处理模式来简化对多线程代码的显式控制。
- 第 21 章——平台互操作性和不安全的代码：必须意识到 C# 是相对年轻的一种语言，许多现有的代码是用其他语言写成的。为了用好这些现有代码，C# 通过 P/Invoke 提供了对互操作性（调用非托管代码）的支持。此外，C# 允许使用指针，也允许执行直接内存操作。虽然使用了指针的代码要求特殊权限才能运行，但它具有与 C 风格的 API 完全兼容的能力。
- **第 22 章——公共语言基础结构（CLI）**：事实上，C# 被设计成一种在 CLI 顶部工作的最有效的编程语言。本章讨论了 C# 程序与底层“运行时”及其规范的关系。

——Mark Michaelis

IntelliTect.com/mark

Twitter: @Intellitect, @MarkMichaelis

致　　谢

世上没有任何一本书是作者单枪匹马就能出版的，在此，我要向整个过程中帮助过我的所有人致以衷心感谢。这里排名不分先后，但家人必然第一。本书已出到第 6 版。在这 10 年的时间里（还不包括以前出的书），我的家人做出了巨大的牺牲。在 Banjamin、Hanna 和 Abigail 眼中，爸爸经常因为此书而无暇顾及他们，但 Elisabeth 承受得更多。家里大事小事全靠她一个人，她独自承担着家庭的重任。（2017 年在外面度假时，好几天我都在“闭门造书”，他们更想去海边。）我感到万分抱歉，谢谢你们！

为保证本书技术上的准确性，许多技术编辑对本书中的各章都进行了仔细审阅。我常常惊讶于他们的认真程度，任何不易察觉的小错误都逃不过他们的火眼金睛，他们是 Paul Bramsman、Kody Brown、Ian Davis、Doug Dechow、Gerard Frantz、Thomas Heavey、Anson Horton、Brian Jones、Shane Kercheval、Angelika Langer、Eric Lippert、John Michaelis、Jason Morse、Nicholas Paldino、Jon Skeet、Michael Stokesbary、Robert Stokesbary、John Timney、Neal Lundby、Andrew Comb、Jason Peterson、Andrew Scott、Dan Haley、Phil Spokas（第 22 章有一部分是他写的）和 Kevin Bost。

Eric Lippert 给了我太多惊喜。他对 C# 的掌握令人“望而生畏”，我很欣赏他的修改，尤其是追求术语完美性方面。本书第 2 版，他大幅改进了和 C# 3.0 相关的章节。那一版我唯一遗憾的就是未能让他审阅全部章节。但遗憾不再。Eric 兢兢业业审阅了《Essential C# 4.0》的每一章，《Essential C# 5.0》和《Essential C# 6.0》更是成为共同作者。在《Essential C# 7.0》中，我非常感谢他的技术编辑工作。谢谢你，Eric！我想象不出还有谁能比你干得更好。正因为你，本书才真正实现了从“很好”到“极好”的飞越。

就像 Eric 之于 C#，很少有人像 Stephen Toub 那样对 .NET Framework 多线程处理有如此深刻的理解。Stephen 专门审阅了（第三次了）重写的关于多线程的两章，并重点检查了 C# 5.0 中的 `async` 支持。谢谢你，Stephen！

感谢 Addison-Wesley 的所有员工，感谢他们在与我合作期间表现出来的极大耐心，容忍我将注意力频频转移到书稿之外的其他事情上。感谢 Trina Fletcher Macdonald、Anna Popick、Julie Nahil 和 Carol Lallier。Trina 值得颁发劳模奖章，在她明显还有其他好多事情的时候，还能容忍我这样的人。Carol 则非常严谨，她改进写作和挑错的本事令人称道（甚至能从代码清单中挑出文法错误）。

作者简介

Mark Michaelis 是高端软件工程和咨询公司 IntelliTect 的创办者、首席技术架构师和培训师。Mark 经常在开发者大会上发言，写过许多文章和书籍，目前是《 MSDN Magazine 》的《 Essential .NET 》专栏作家。

从 1996 年起，他一直是 C#、Visual Studio Team System 和 Windows SDK 的 MVP。2007 年被评选为微软的 Regional Director。他还服务于微软的几个软件设计评审团队，包括 C# 和 VSTS。

Mark 拥有伊利诺伊大学哲学专业文学学士学位和伊利诺伊理工大学计算机硕士学位。

他不是痴迷于计算机，就是忙于陪伴家人或者玩壁球（2016 年暂停铁人三项训练）。他居住在华盛顿州的斯波坎，他和妻子 Elisabeth 有三个孩子：Benjamin、Hanna 和 Abigail。

技术编辑简介

Eric Lippert 目前在 Facebook 负责开发者工具。之前是微软 C# 语言设计团队的一员。不在 StackOverflow 上回答用户的 C# 问题或者编辑程序书时，他总是喜欢玩他的小帆船。目前和妻子 Leah 居住在华盛顿州的西雅图。