HZ BOOKS 华章IT

EFFECTIVE 系列丛书

Pearson

经典畅销书《Effective C#》的进阶篇，世界知名专家Bill Wagner倾力撰写，针对C# 7.0全面更新

紧贴C#语言的设计理念，既从正面阐释如何编写高效代码，又从反面入手指出容易出错之处，涵盖C#语言的各个方面

More Effective C#

50 Specific Ways to Improve Your C#

Second Edition

More Effective C#

改善C#代码的50个有效方法

（原书第2版）

[美] 比尔·瓦格纳（Bill Wagner） 著

爱飞翔 译

机械工业出版社
China Machine Press

EFFECTIVE
系列丛书

More Effective C#
50 Specific Ways to Improve Your C#
Second Edition

More Effective C#

改善C#代码的50个有效方法

（原书第2版）

[美] 比尔·瓦格纳（Bill Wagner） 著
爱飞翔 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

More Effective C#: 改善 C# 代码的 50 个有效方法（原书第 2 版）/（美）比尔·瓦格纳（Bill Wagner）著；爱飞翔译．—北京：机械工业出版社，2019.3
（Effective 系列丛书）
书名原文：More Effective C#:50 Specific Ways to Improve Your C#, Second Edition

ISBN 978-7-111-62071-6

I. M… II. ① 比… ② 爱… III. C 语言 - 程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字（2019）第 035131 号

本书版权登记号：图字 01-2017-7866

More Effective C#

改善 C# 代码的 50 个有效方法（原书第 2 版）

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：张志铭 责任校对：李秋荣
印 刷：三河市宏图印务有限公司 版 次：2019 年 3 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：17
书 号：ISBN 978-7-111-62071-6 定 价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

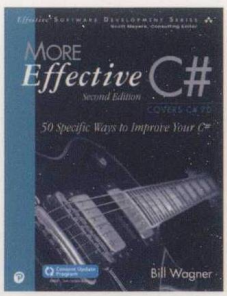
本书是经典畅销书《Effective C#》的进阶篇，世界知名专家Bill Wagner倾力撰写，针对C# 7.0全面更新。

在内容编排上，与上一版相比，新版删掉了与当前C#语言和应用开发无关的内容，新增的条目涵盖语言与框架的新特性以及在使用C#语言进行开发的过程中总结出来的经验。之前版本的《Effective C#》中的部分条目也移到了本书中。总之，本书中的50条方法可以帮助大家更高效地使用C#语言，成为更加专业的开发者。

全书分为6章，50条。第1章介绍如何根据不同的编程范式来选用合适的写法。第2章讲解如何利用C#语言的众多特性来准确表达自己的设计思路，如怎样利用惰性初始化机制、怎样创建易于拼接的接口等。第3章探讨怎样使用C#语言中基于任务的异步特性来创建合适的API，以便准确地告诉调用方这个API会如何利用各种服务及资源。第4章讲解多线程并行执行，这是异步编程中的一个领域。第5章讨论怎样把C#当成动态语言使用，怎样使用这些动态特性，以及怎样避免动态类型在程序中过于泛滥。第6章讲解如何更好地与全世界的C#开发者交流，为C#语言的发展做出贡献。

作者简介

Bill Wagner 世界知名的C#专家，ECMA C#标准委员会成员，畅销书《Effective C#》的作者。他在 Humanitarian Toolbox 项目中担任主管，并荣获微软公司.NET MVP 称号长达11年，最近开始在 .NET Foundation Advisory Council 任职。Wagner 在工作中与各种规模的公司合作，帮助这些创业公司或企业改进软件开发流程，并培养软件开发团队。目前，他是微软.NET核心内容团队的员工，写过很多与C#语言及.NET框架有关的学习资料。他拥有伊利诺伊大学厄巴纳-尚佩恩分校计算机科学专业的学士学位。



原书封面

EFFECTIVE 系列丛书

译者序 · THE TRANSLATOR'S WORDS

Effective

本书与作者写的另一本书（即《 Effective C# 》(第 3 版）[⊖]）可以结合起来阅读。二者之间的关系也与 Effective 书系中针对其他语言而写的那些作品一样，是相辅相成的。如果说《 Effective C# 》讲的是基本技巧，那么《 More Effective C# 》就是在此基础上的拓展和延伸，从而帮助我们把自己的知识体系打造得更加完备。

C# 语言经历了较长的发展过程，使用 C# 做开发的人越来越多，在这个过程中，大家也总结出了许多心得。本书作者 Wagner 先生有丰富的经验，他在分享这些经验时所遵循的理念很值得注意。

首先，作者经常提醒大家，应该优先采用现有的语言、程序库及开源项目中的机制与工具来实现相关的功能，而不要总是想着从头去写。他通过范例演示了前者的好处，让我们看到这种做法不易出错，而且能跟其他开发者所写的代码兼容。

其次，作者告诉我们，如果现有的工具无法满足需求，那么应该怎样编写正确的代码。为此，作者讲述了 C# 语言的许多特性，并通过实际的代码演示了在使用这些特性的过程中所要注意的问题，以及应该避开的陷阱。

本书讲解这些内容时很善于在两种视角之间切换。有时它讲的是某个功能怎样实现会比较简单，有时它又提醒你这项功能应该通过什么样的接口开放给外界使用，以便让其他开发者能够准确理解你的设计意图，从而顺畅地使用这套接口。这在多人协作的环境中尤其重要，而且也有助于提升软件设计水平。

⊖ 中文版已由机械工业出版社引进出版，全名为《 Effective C# ：改善 C# 代码的 50 个有效方法（原书第 3 版）》，ISBN 为 978-7-111-59719-3。——编辑注

另外，作者还特别关注 C# 引入的新特性，并通过详细的范例演示了这些特性的优势以及它们的正确用法。这将促使你逐渐习惯新的写法，而不会总是停在某些过时的资料所给出的旧方案上。

本书虽然介绍了许多新的特性，但依然是按照 C# 的固有理念来讲述这些特性的。你可以由此了解到，怎样才能在尽量发挥 C# 自身优势的前提下，合理运用这些特性及工具实现出更灵活、更高效的解决方案。这种运用方式也能让 C# 开发界在传承的过程中有所创新。

翻译过程中，我得到了机械工业出版社华章公司诸位工作人员的帮助，在此深表谢意。尤其感谢关敏编辑给我机会，让我能够遇见这些优秀的技术书籍。

书中的术语参考了 Microsoft 的语言门户网站（www.microsoft.com/Language/zh-cn/Search.aspx）、技术文档网站（docs.microsoft.com/zh-cn）以及其他一些技术文章。

由于译者水平有限，文中难免有错误与疏漏之处，请读者发邮件至 eastarstormlee@gmail.com ，或访问 github.com/jeffreybaoshenlee/mecs2-errata/issues 留言，给我以批评和指教。

爱飞翔

2018 年 11 月

本书假设你是使用 C# 7 来开发程序的，然而笔者并不会详细地讲到这一版 C# 语言所具备的每一种新特性。与 Effective 软件开发书系的其他作品类似，本书关注的也是怎样运用语言特性来解决日常工作中的实际问题。C# 7 中有一些新特性，可以实现出比旧式做法更为高效的新方案，本书尤其关注这些特性。大家在网上找到的某些解决办法可能是几年前的旧方案，针对这种情况，笔者会专门指出用新特性实现出的方案为什么比早前的那些办法要好。

你可以用基于 Roslyn 的分析器（analyzer）和代码修复程序（code fix）来判断某段 C# 代码有没有遵循书中所提到的某些建议。笔者把相关资源放在了 https://github.com/BillWagner/MoreEffectiveCSharpAnalyzers 上。如果你有任何想法，或是打算给这份代码库提供新的内容，请点击该网页上的 issue 或 pull request。

目标读者

本书适合以 C# 为首选编程语言的专业开发者阅读。你应该熟悉 C# 语言的写法及各项特性，并且能够熟练地运用 C# 语言的一般功能。书中不会再教你如何利用这些特性，而是要告诉你怎样把当前这一版 C# 语言所具备的特性正确地运用于日常开发工作中。

除了要熟悉 C# 本身的特性外，你还应该了解 CLR（Common Language Runtime，公共语言运行时）与 JIT（Just-In-Time，即时）编译器。

每章概述

当今世界，数据无处不在，但对待数据的方式却各有不同。面向对象的编程范式把数据与代码都当成类型及其职责的一部分。函数式的编程范式将方法视为数据。面向服务的编程范式则把数据与操纵数据的代码分隔开来。C# 语言在演变的过程中把这些编程范式用到的习惯写法全都包括了进来，这就要求开发者在选择设计方式时必须多加考虑。第 1 章会告诉你怎样根据不同的编程范式来选用合适的写法。

编写程序在很大程度上是在设计 API。使用 API 的人可以通过 API 看出 API 的设计者所规划的用法，还可以看出设计者是怎样理解其他开发者的需求和期望的。第 2 章介绍如何利用 C# 语言的众多特性来准确表达自己的设计思路，例如，怎样利用惰性初始化机制，怎样创建易于拼接的接口，以及怎样避免公有接口中的各种语言特性给人们带来困惑，等等。

基于任务的异步编程要求开发者采用一些新的写法，把这些基本的异步单元组合成完整的应用程序。掌握这些异步特性，有助于创建良好的异步操作 API，这些 API 要能够准确地

反映出代码的执行方式，从而令调用者使用起来更加顺畅。第 3 章讲解怎样使用 C# 语言中基于任务的异步特性来创建合适的 API，以便准确地告诉调用方这个 API 会如何利用各种服务及资源。

第 4 章专门讲解异步编程中的一个领域：多线程并行执行。你会看到怎样利用 PLINQ 方便地拆解复杂的算法，从而令其能够运行在多个处理器核心及多个 CPU 上。

第 5 章讨论怎样把 C# 当成动态语言来使用。C# 本身是强类型的静态类型语言，然而当今很多程序都在同时运用动态类型与静态类型这两种机制。C# 一方面可以继续发挥静态类型的优势，另一方面又允许开发者在程序中同时运用动态编程的一些写法。第 5 章会讲解如何使用这些动态特性，以及怎样避免动态类型在整个程序中过于泛滥。

第 6 章会提出一些建议，告诉你怎样更好地与全世界的 C# 开发者交流。你可以通过各种办法为 C# 语言的发展做出贡献，并帮助大家把这门日常语言打造得更加优秀。

代码约定

在书中展示代码需要兼顾版面与清晰度。笔者尽量将范例代码写得较为精简，使其能够专注于该段代码所要讲解的问题，这就意味着类或方法中与本问题无关的部分可能会省略，而且错误恢复代码可能也不会写出，以求节省篇幅。公有方法应该验证调用方所传入的参数及其他输入数据，不过，由于篇幅的限制，这些内容同样会省略。此外，复杂的算法通常会对方法调用做出验证并编写 try/finally 结构，这些内容也会因篇幅过大而被略去。

笔者假设你能够从代码中看出它用到的几个常见命名空间。你可以认为每段范例代码都采用了下面几条 using 语句：

```
using System;
using static System.Console;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

提供反馈

笔者尽量保证书中的文字与代码准确，这些内容也经过了其他人审阅，然而其中或许还是会有一些错误。如果你发现了错误，那么请发送邮件到 bill@thebillwagner.com，或通过 Twitter 号 @billwagner 联系我。本书的勘误表发布在 http://thebillwagner.com/Resources/

More EffectiveCS 上。书中有很多条目都是笔者通过电子邮件或 Twitter 与其他 C# 开发者讨论时想出来的。如果对这些条目所给出的建议有想法或评论，也请联系我。更一般的话题则可以在博客 http://thebillwagner.com/blog 上讨论。

致谢

本书能够写成，得益于很多朋友所提供的帮助。这些年来，笔者有幸结识了很多优秀的 C# 开发者。C# Insiders 邮件列表中的每个人（无论是否在 Microsoft 公司）都提供了见解，并跟我交流，让我能把这本书写得更好。

其中，有几位 C# 开发者不仅直接提供了思路，而且还帮我把这些思路落实成具体的条目。感谢 Jon Skeet、Dustin Campbell、Kevin Pilch、Jared Parsons、Scott Allen 与我讨论本书内容，尤其要多谢 Mads Torgersen，这一版中有很多新的想法都源自他的见解。

这一版的技术评审团队也很棒。Jason Bock、Mark Michaelis 与 Eric Lippert 认真检查了书里的文字和范例，使得本书质量大幅提高。他们细致而详尽的态度令笔者特别佩服。此外，他们还给出了一些建议，使我能把书中的许多话题解释得更加清楚。

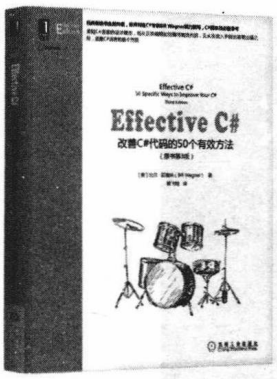
与 Addison-Wesley 团队共事相当愉快。Trina Macdonald 是位出色的编辑与督导，是她敦促我写成这本书。Mark Renfrow 与 Olivia Basegio 极力支持 Trina 的编辑工作，他们也给我提供了许多帮助，以确保整份书稿都包含高品质的内容。Curt Johnson 依然很好地完成了本书的营销工作，无论你现在读到的是哪种格式，都离不开他的努力。

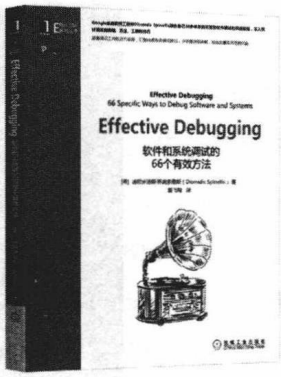
本书能够收录在 Scott Meyers 的 Effective 书系中，笔者深感荣幸。他读过每一本书稿，并提出了修改意见。Scott 是一位水平高超、经验丰富的软件开发者，他虽然不做 C#，但却能在书稿中发现那些解释得不够清楚、论证得不够充分之处。他为这一版所提供的建议与早前一样，都十分宝贵。

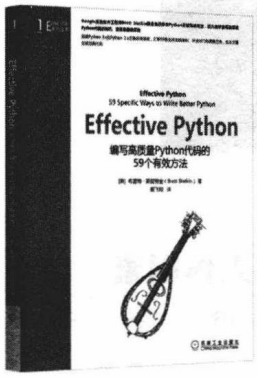
最后，感谢家人给我留出时间，让我能够写完本书。妻子 Marlene 总是能耐心地等我把文字或范例代码安排好。我能够写出包括本书在内的许多作品，并且写得如此顺畅，全靠她支持。

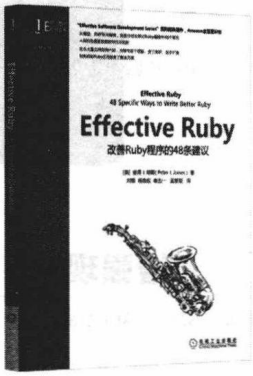
推荐阅读

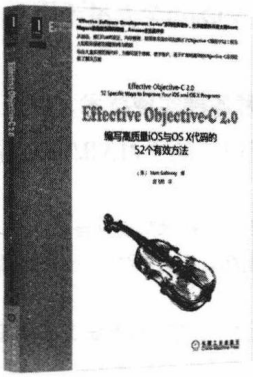
Effective系列

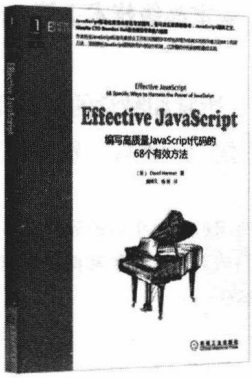




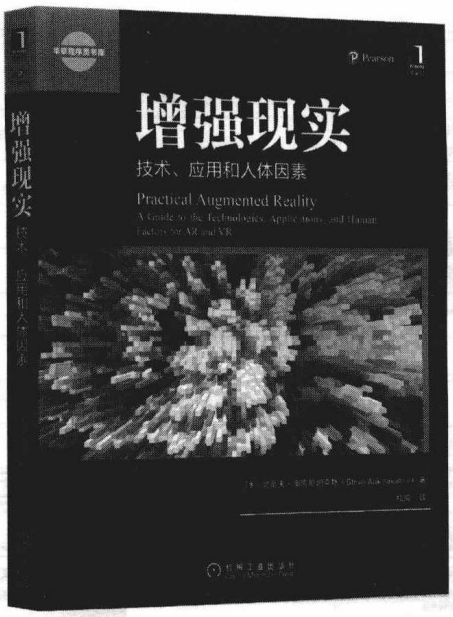








推荐阅读



增强现实：技术、应用和人体因素

作者：Steve Aukstakalnis ISBN：978-7-111-58168-0 定价：79.00元

美国国家科学基金会AR/VR技术资深专家亲笔撰写，美国国家航空航天局喷气推进实验室高级技术主管Victor Luo作序推荐，Amzon全五星评价

从介绍视觉、听觉和触觉的机制开始，深入浅出地讲解各种实现技术，以及AR/VR技术在游戏、建筑、医疗、航空航天和教育等领域的应用，是学习AR/VR的必读之作

我们已经使用过本书里讨论到的很多技术，并为即将到来的更多新技术感到兴奋……

让本书成为帮助大家理解和拓宽增强现实与虚拟现实领域的指南，因为AR技术已经如同电视机和互联网一样无处不在……

—— 维克多·罗 美国国家航空航天局喷气推进实验室软件系统工程学高级技术主管

EFFECTIVE 系列丛书

目录 · CONTENTS

Effective

EFFECTIVE 系列丛书

Effective

CHAPTER 1 · 第 1 章

处理各种类型的数据

C# 语言原本是设计给面向对象的开发者使用的，这种开发方式会把数据与功能合起来处理。在 C# 逐渐成熟的过程中，它又添加了一些新的编程范式，以便支持其他一些常用的开发方式。其中有一种开发方式强调把数据存储方法与数据操作方法分开，这种方式随着分布式系统而兴起，此类系统中的应用程序分成多个小的服务，每个服务只实现一项功能，或者只实现一组相互联系的功能。如果要把数据的存储与操作分开，那么开发者就得有一些新的编程技术可供使用，正是这些需求促使 C# 语言添加了与之相应的一些特性。

本章会介绍怎样把数据本身与操纵或处理该数据的方法分开。此处所说的数据不一定都是对象，也有可能是函数或被动的数据容器。

第 1 条：使用属性而不是可直接访问的数据成员

属性一直是 C# 语言的特色，目前的属性机制比 C# 刚引入它的时候更为完备，这使得开发者能够通过属性实现很多功能，例如，可以给 getter ⊖与 setter 设定不同的访问权限。与直接通过数据成员来编程的方式相比，自动属性可以省去大量的编程工作，而且开发者可以通过该机制轻松地定义出只读的属性。

⊖ 本书中的 get accessor（get 访问器）与 set accessor（set 访问器）也可以称为 getter 和 setter。——译者注

此外还可以结合以表达式为主体的（expression-bodied）写法将代码变得更紧凑。有了这些机制，就不应该继续在类型中创建公有（public）字段，也不应该继续手工编写 get 与 set 方法。属性既可以令调用者通过公有接口访问相关的数据成员，又可以确保这些成员得到面向对象式的封装。在 C# 语言中，属性这种元素可以像数据成员一样被访问，但它们其实是通过方法来实现的。

类型中的某些成员很适合用数据来表示，如顾客的名字、点的（x, y）坐标以及上一年的收入等。

如果用属性来实现这些成员，那么在调用你所创建的接口时，就可以像使用方法那样，通过这些属性直接访问数据字段。这些属性就像公有字段一样，可以轻松地访问。而另一方面，开发这些属性的人则可以在相关的方法中定义外界访问该属性时所产生的效果。

.NET Framework 会做出这样一种预设：它认为开发者都是通过属性来表达公有数据成员的。这可以通过其中与数据绑定（data binding）有关的类而得到印证，因为这些类是通过属性而非公有数据字段来提供支持的。WPF（Windows Presentation Foundation）、Windows Forms 以及 Web Forms 在把对象中的数据与用户界面中的控件绑定时，都以相关的属性为依据。数据绑定机制会通过反射在类型中寻找与名称相符的属性：

```
textBoxCity.DataBindings.Add("Text",
    address, nameof(City));
```

这段代码会把 textBoxCity 控件的 Text 属性与 address 对象的 City 属性绑定。假如你在 address 对象中用的是名为 City 的公有数据字段，而不是属性，那么这段代码将无法正常运作，因为 Framework Class Library 的设计者本来就没打算支持这种写法。直接使用公有数据成员是一种糟糕的编程方式，Framework Class Library 不为这种方式提供支持。这也是促使开发者改用属性来编程的原因之一。

数据绑定机制只针对那些其元素需要显示在用户界面（UI）中的类，然而，属性的适用范围却不仅仅局限于此。在其他的类与结构中，也应该多使用属性，这样可以让你在发现新的需求时，更为方便地修改代码。比方说，如果你现在决定 Customer 类型中的 name（名字）数据不应出现空白值，那么只需修改 Name 属性的代码即可：

```
public class Customer
{
    private string name;
    public string Name
    {
        get => name;
        set
        {
            if (string.IsNullOrWhitespace(value))
```