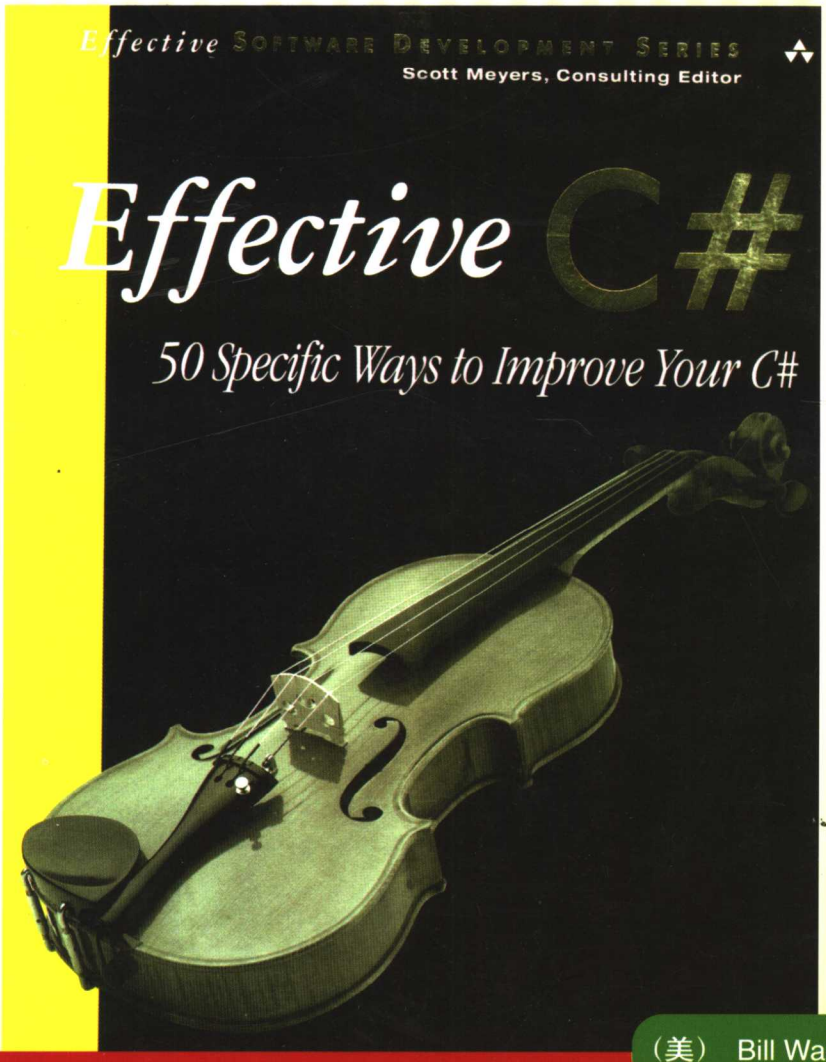
经 典 原 版 书 库

Effective C#

（英文版）



（美） Bill Wagner 著

机械工业出版社
China Machine Press

经 典 原 版 书 库

Effective C#

（英文版）

Effective C#
50 Specific Ways to Improve Your C#

（美） Bill Wagner 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2005-4832

图书在版编目（CIP）数据

Effective C#（英文版）/（美）瓦格纳（Wagner, B.）著. －北京：机械工业出版社，2006.1

（经典原版书库）

书名原文：Effective C#: 50 Specific Ways to Improve Your C#

ISBN 7-111-17473-9

Ⅰ.E… Ⅱ.瓦… Ⅲ.C语言－程序设计－英文 Ⅳ.TP312

中国版本图书馆CIP数据核字（2005）第112828号

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：迟振春

北京京北制版厂印刷 · 新华书店北京发行所发行

2006年1月第1版第1次印刷

718mm × 1020mm 1/16 · 20.25印张

印数：0 001-3 000册

定价：35.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

出版者的话

文艺复兴以降，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；也正是这样的传统，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，还揭橥了学术的源变，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、从业人员较少的现状下，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图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann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Tanenbaum，Stroustrup，Kernighan，Jim Gray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、研究及庋藏。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襄助，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，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，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，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步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步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔

滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

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，但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。教材的出版只是我们的后续服务的起点。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：

电子邮件：hzjsj@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

Praise for *Effective C#*

"This book really demonstrates Bill's strengths as a writer and programmer. In a very short amount of time, he is able to present an issue, fix it, and conclude it; each chapter is tight, succinct, and to the point."

—Josh Holmes, independent contractor

"The book provides a good introduction to the C# language elements from a pragmatic point of view, identifying best practices along the way and following a clear and logical progression from the basic syntax to creating components to improving your code-writing skills. Since each topic is covered in short entries, it is very easy to read and you'll quickly realize the benefits of the book."

—Tomas Restrepo, Microsoft MVP

"The book covers the basics well, especially with respect to the decisions needed when deriving classes from System.Object. It is easy to read with examples that are clear, concise, and solid. I think it will bring good value to most readers."

—Rob Steele, Central Region Integration COE and Lead Architect, Microsoft

"*Effective C#* provides the C# developer with the tools they need to rapidly grow their experience in Visual C# 2003 while also providing insight into the many improvements to the language that will be hitting a desktop near you in the form of Visual C# 2005."

—Doug Holland, Precision Objects

"Part of the point of the .NET Framework—and the C# Language, in particular—is to let the developer focus on solving customer problems and delivering a product, rather than spending hours (or even weeks) writing plumbing code. Bill Wagner's *Effective C#* not only shows you what's going on behind the scenes, but also shows you how to take advantage of particular C# code constructs. Written in a dispassionate style that focuses on the facts—and just the facts—of writing effective C# code, Wagner's book drills down into practices that will let you write C# applications and components that are easier to maintain as well as faster to run. I'm recommending *Effective C#* to all students of my "NET Boot-Camp" and other C#-related courses.

—Richard Hale Shaw, www.RichardHaleShawGroup.com

"*Effective C#* is very well organized and easy to read with a good mix of code and explanations that give the reader deep understanding of the topic. The author is an authority on C# and the .NET runtime, but keeps the content accessible and easy to read through a conversational tone while still imparting expert knowledge to the reader."

—Brian Noyes, Principal Software Architect, IDesign, Inc.

Introduction

This book is designed to offer practical advice for the programmer on how to improve productivity when using the C# language and the .NET libraries. In it, I have comprised 50 key items, or minitopics, related to the most–frequently-asked questions that I (and other C# consultants) have encountered while working with the C# community.

I started using C# after more than 10 years of C++ development, and it seems that many C# developers are following suit. Throughout the book, I discuss where following C++ practices may cause problems in using C#. Other C# developers are coming to the language with a strong Java background; they may find some of these passages rather obvious. Because some of the best practices change from Java to C#, I encourage Java developers to pay special attention to the discussions on value types (see Chapter 1, "C# Language Elements"). In addition, the .NET Garbage Collector behaves differently than the JVM Garbage Collector (see Chapter 2, ".NET Resource Management").

The items in this book are the collection of recommendations that I most often give developers. Although not all items are universal, most of the items can be easily applied to everyday programming scenarios. These include discussions on Properties (Item 1), Conditional Compilation (Item 4), Immutable Types (Item 7), Equality (Item 9), ICloneable (Item 27), and the new Modifier (Item 29). It has been my experience that, in most situations, decreasing development time and writing good code are the primary goals of the programmer. Certain scientific and engineering applications may place the highest premium on the overall performance of the system. Still, for others, it's all about the scalability. Depending on your goals, you might find particular information more (or less) relevant under certain circumstances. To address this, I have tried to explain the goals in detail. My discussions on readonly and const (Item 2), Serializable Types (Item 25), CLS Compliance (Item 31), Web Methods

(Item 34), and DataSets (Item 41) assume certain design goals. Those goals are spelled out in these items, so that you can decide what is most applicable for you in your given situation.

Although each item in *Effective C#* stands alone, it is important to understand that the items have been organized around major topics, such as C# language syntax, resource management, and object and component design. This is no accident. My goal is to maximize learning the material covered in the book by leveraging and building each item on earlier items. Don't let that keep you from using it as a reference, though. If you have specific questions, this book functions well as the ideal "ask-me" tool.

Please keep in mind that this is not a tutorial or a guide to the language, nor is it going to teach you C# syntax or structure. My goal is to provide guidance on the best constructs to use in different situations.

Who Should Read this Book?

Effective C# is written for professional developers, those programmers who use C# in their daily work lives. It assumes that you have some experience with object-oriented programming and at least one language in the C family: C, C++, C#, or Java. Developers with a Visual Basic 6 background should be familiar with both the C# syntax and object-oriented design before reading this book.

Additionally, you should have some experience with the major areas of .NET: Web Services, ADO.NET, Web forms, and Windows Forms. I reference these concepts throughout the book.

To fully take advantage of this book, you should understand the way the .NET environment handles assemblies, the Microsoft Intermediate Language (MSIL), and executable code. The C# compiler produces assemblies that contain MSIL, which I often abbreviate as IL. When an assembly is loaded, the Just In Time (JIT) Compiler converts that MSIL into machine-executable code. The C# compiler does perform some optimizations, but the JIT compiler is responsible for many more effective optimizations, such as inlining. Throughout the book, I've explained which process is involved in which optimizations. This two-step compilation process has a significant effect on which constructs perform best in different situations.

About the Content

Chapter 1, "C# Language Elements," discusses the C# syntax elements and the core methods of `System.Object` that are part of every type you write. These are the topics that you must remember every day when you write C# code: declarations, statements, algorithms, and the `System.Object` interface. In addition, all the items that directly relate to the distinction between value types and reference types are in this chapter. Many items have some differences, depending on whether you are using a reference type (class) or a value type (struct). I strongly encourage you to read the discussions on value and reference types (Items 6 through 8) before reading deeper into the book.

Chapter 2, ".NET Resource Management," covers resource management with C# and .NET. You'll learn how to optimize your resource allocation and usage patterns for the .NET managed execution environment. Yes, the .NET Garbage Collector makes your life much simpler. Memory management is the environment's responsibility, not yours. But, your actions can have a big impact on how well the Garbage Collector performs for your application. And even if memory is not your problem, nonmemory resources are still your responsibility; they can be handled through `IDisposable`. You'll learn the best practices for resource management in .NET here.

Chapter 3, "Expressing Designs with C#," covers object-oriented design from a C# perspective. C# provides a rich palette of tools for your use. Sometimes, the same problems can be solved in many ways: using interfaces, delegates, events, or attributes and reflection. Which one you pick will have a huge impact on the future maintainability of your system. Choosing the best representation of your design will help to make it easier for the programmers using your types. The most natural representation will make your intent clearer. Your types will be easier to use and harder to misuse. The items in Chapter 3 focus on the design decisions you will make and when each C# idiom is most appropriate.

Chapter 4, "Creating Binary Components," covers components and language interoperability. You'll learn how to write components that can be consumed by other .NET languages, without sacrificing your favorite C# features. You'll also learn how to subdivide your classes into components in order to upgrade pieces of your application. You should be able to release new versions of a component without redistributing the entire application.

Chapter 5, "Working with the Framework," covers underutilized portions of the .NET Framework. I see a strong desire in many developers to create their own software rather than use what's already been built. Maybe it's the size of the .NET Framework that causes this; maybe it's that the framework is completely new. These items cover the parts of the framework where I have seen developers reinvent the wheel rather than use what they've been given. Save yourself the time by learning how to use the framework more efficiently.

Chapter 6, "Miscellaneous," finishes with items that did not fit in the other categories and with a look forward. Look here for C# 2.0 information, standards information, exception-safe code, security, and interop.

A Word About the Items

My vision for these items is to provide you with clear and succinct advice for writing C# software. Some guidelines in the book are universal because they affect the correctness of a program, such as initializing data members properly (see Chapter 2). Others are not so obvious and have generated much debate in the .NET community, such as whether to use ADO.NET `DataSets`. While I personally believe that using them is a great timesaver (see Item 41), other professional programmers, whom I highly respect, disagree. It really depends on what you're building. My position comes from a timesaving stance. For others who write a great deal of software that transfer information between .NET- and Java-based systems, `DataSets` are a bad idea. Throughout the book, I support and have given justification for all the suggestions I make. If the justification does not apply to your situation, neither does the advice. When the advice is universal, I usually omit the obvious justification, which is this: Your program won't work otherwise.

Styles and Code Conventions

One difficulty in writing a book about a programming language is that language designers give real English words a very specific new meaning. This makes for passages that can be tough to understand: "Develop interfaces with interfaces" is an example. Any time I use a specific language keyword, it is in the `code` style. When I discuss general topics with specific C# concepts, the specific C# topic is capitalized, as in: "Create

`Interfaces` to represent the interface supported by your classes." It's still not perfect, but it should make many of these passages easier to read.

Many related C# terms are used in this book. When I refer to a member of type, it refers to any definition that can be part of a type: methods, properties, fields, indexers, events, enums, or delegates. When only one applies, I use a more specific term. A number of terms in this book might or might not already be familiar to you. When these terms first appear in the text, they are set in **bold** and defined.

The samples in this book are short, focused sections of code that are meant to demonstrate the advice of that particular item. They are designed to highlight the advantages of following the advice. They are not complete samples to incorporate into your current programs. You cannot just copy the listings and compile them. I've omitted many details from each listing. In all cases, you should assume the presence of common `using` clauses:

```
using System;
using System.IO;
using System.Collections;
using System.Data;
```

When I use less common namespaces, I make sure that you can see the relevant namespace. Short samples use the fully qualified class names, and long samples include the less common `using` statements.

I take similar liberties with code inside the samples. For example, when I show this:

```
string s1 = GetMessage( );
```

I might not show the body of the `GetMessage()` routine if it's not relevant to the discussion. Whenever I omit code, you can assume that the missing method does something obvious and reasonable. My purpose in this is to keep us focused on the particular topic. By omitting code that is not part of that topic, we don't become distracted. It also keeps each item short enough that you should be able to finish an item in one sitting.

Regarding C# 2.0

I say little about the upcoming C# 2.0 release; there are two reasons for this. First and foremost, most of the advice in this book applies just as

well for C# 2.0 as it does for the current version. Although C# 2.0 is a significant upgrade, it is built on C# 1.0 and does not invalidate most of today's advice. Where the best practices will likely change, I've noted that in the text.

The second reason is that it's too early to write the most effective uses of the new C# 2.0 features. This book is based on the experience I've had—and the experience my colleagues have had—using C# 1.0. None of us has enough experience with the new features in C# 2.0 to know the best ways to incorporate them into our daily tasks. I'd rather not mislead you when the simple fact is that the time to cover the new C# 2.0 features in an Effective book has not yet arrived.

Making Suggestions, Providing Feedback, and Getting Book Updates

This book is based on my experiences and many conversations with colleagues. If your experience is different, or if you have any questions or comments, I want to hear about it. Please contact me via email: `wwagner@srtsolutions.com`. I'll be posting those comments online as an extension to the book. See `www.srtsolutions.com/EffectiveCSharp` for the current discussion.

Acknowledgments

Although writing seems like a solitary activity, this book is the product of a large team of people. I was lucky enough to have two wonderful editors, Stephane Nakib and Joan Murray. Stephane Nakib first approached me about writing for Addison Wesley a little more than a year ago. I was skeptical. The bookstore shelves are filled with .NET and C# books. Until C# 2.0 has been around long enough to cover thoroughly, I could not see the need for another reference, tutorial, or programming book about C# and .NET. We discussed several ideas and kept coming back to a book on C# best practices. During those discussions, Stephane told me that Scott Meyers had begun an Effective series, modeled after his Effective C++ books. My own copies of Scott's three books are very well worn. I've also recommended them to every professional C++ developer I know. His style is clear and focused. Each item of advice has solid justifications. The Effective books are a great resource, and the format makes it easy to

remember the advice. I know many C++ developers who have copied the table of contents and tacked it to the cubicle wall as a constant reminder. As soon as Stephane mentioned the idea of writing *Effective C#*, I jumped at the opportunity. This book contains in one place all the advice I've been giving to C# developers. I am honored to be a part of this series. Working with Scott has taught me a great deal. I only hope this book improves your C# skills as much as Scott's books improved my C++ skills.

Stephane helped pitch the idea of an *Effective C#* book, reviewed outlines and manuscripts, and graciously championed the book through the early writing process. When she moved out of acquisitions, Joan Murray picked up and shepherded the manuscript through production without missing a beat. Ebony Haight provided a constant presence as an editorial assistant through the entire process. Krista Hansing did all the copyediting, turning programmer jargon into English. Christy Hackerd did all the work to turn the word documents into the finished book you now hold.

Any errors that remain are mine. But the vast majority of errors, omissions, and unclear descriptions were caught by a wonderful team of reviewers. Most notably, Brian Noyes, Rob Steel, Josh Holmes, and Doug Holland made the text you have in front of you more correct and more useful than the earlier drafts. Also, thank you to all the members of the Ann Arbor Computing Society, the Great Lakes Area .NET User Group, the Greater Lansing User Group, and the West Michigan .NET User Group, who all heard talks based on these items and offered great feedback.

Most of all, Scott Meyers' participation had a huge, positive impact on the final version of this book. Discussing early drafts of this book with him made me clearly understand why my copies of the *Effective C++* books are so worn. Very little, if anything, escapes his reviews.

I want to thank Andy Seidl and Bill French from MyST Technology Partners (`myst-technology.com`). I used a secure MyST-based blogsite to publish early drafts of each item for reviewers. The process was much more efficient and shortened the cycle between drafts of the book. We've since opened parts of the site for the public so you can see parts of the book in an online format. See `www.srtsolutions.com/EffectiveCSharp` for the online version.

I've been writing magazine articles for several years now and I need to publicly thank the person who got me started: Richard Hale Shaw. He

gave me (as an untested author) a column in the original *Visual C++ Developer's Journal* he helped found. I would not have discovered how much I enjoy writing without his help. I also would not have had the opportunity to write for *Visual Studio Magazine*, *C# Pro*, or *ASP.NET Pro* without the initial help he gave.

Along the way, I've been fortunate to work with many wonderful editors at different magazines. I'd like to list them all, but space does not permit it. One does deserve a special mention: Elden Nelson. I've enjoyed all our time working together and he has had a strong positive effect on my writing style.

My business partners, Josh Holmes and Dianne Marsh, put up with my limited involvement in the company while taking the time to write this book. They also helped review manuscripts, ideas, and thoughts on items.

Throughout the long process of writing, the guidance of my parents, Bill and Alice Wagner, to always finish what I start could be the only reason you are now holding a completed book.

Finally, the most important thanks go to my family: Marlene, Lara, Sarah, and Scott. Writing a book takes an incredible amount of time from all those activities we enjoy. And after all this time on the book, their continued patience has never wavered.

Contents

1 C# Language Elements

Why should you change what you are doing today if it works? The answer is that you can be better. You change tools or languages because you can be more productive. You don't realize the expected gains if you don't change your habits. This is harder when the new language, C#, has so much in common with a familiar language, such as C++ or Java. It's easy to fall back on old habits. Most of these old habits are fine. The C# language designers want you to be able to leverage your knowledge in these languages. However, they also added and changed some elements to provide better integration with the Common Language Runtime (CLR), and provide better support for component-oriented development. This chapter discusses those habits that you should change—and what you should do instead.

Item 1: Always Use Properties Instead of Accessible Data Members

The C# language promoted properties from an ad-hoc convention to a first-class language feature. If you're still creating public variables in your types, stop now. If you're still creating `get` and `set` methods by hand, stop now. Properties let you expose data members as part of your public interface and still provide the encapsulation you want in an object-oriented environment. **Properties** are language elements that are accessed as though they are data members, but they are implemented as methods.

Some members of a type really are best represented as data: the name of a customer, the x,y location of a point, or last year's revenue. Properties enable you to create an interface that acts like data access but still has all the benefits of a function. Client code accesses properties as though they are accessing public variables. But the actual implementation uses methods, in which you define the behavior of property accessors.