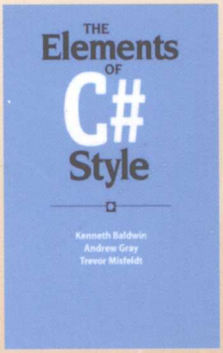
TURING 图灵程序设计丛书 微软技术系列

The Elements of C# Style

C#编程风格（英汉对照）

Kenneth Baldwin
[美] Trevor Misfeldt 著
Andrew Gray

韩磊 译



- 享誉全球的C#经典著作
- 顶级软件公司编程规范，
多位世界级专家经验结晶
- 卓越软件团队必由之路

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 微软技术系列

The Elements of C# Style
C#编程风格（英汉对照）

[美] Kenneth Baldwin
Trevor Misfeldt 著
Andrew Gray

韩磊 译

人民邮电出版社
北京

图书在版编目（CIP）数据

C# 编程风格：英汉对照 /（美）鲍德温（Baldwin, K.），（美）格雷（Gray, A.），（美）米斯费尔特（Misfeldt, T.）著；韩磊译．—北京：人民邮电出版社，2008.10
（图灵程序设计丛书）
ISBN 978-7-115-18438-2

Ⅰ. C… Ⅱ. ①鲍…②格…③米…④韩… Ⅲ. C 语言－程序设计－英、汉 Ⅳ. TP312

中国版本图书馆CIP数据核字（2008）第096396号

内 容 提 要

本书是一部久经考验、短小精悍的 C# 编程规范。给出的 C# 编码规则和建议主要涉及格式、命名、文档、设计、编程以及包等内容，能够帮助广大程序员编写出更易于理解、维护、扩展，更有效，更专业的 C# 代码。

本书适用于各层次 C# 程序员。

图灵程序设计丛书

C#编程风格（英汉对照）

◆ 著　　[美] Kenneth Baldwin, Andrew Gray, Trevor Misfeldt
　译　　韩　磊
　责任编辑　陈兴璐

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
　邮编　100061　　电子函件　315@ptpress.com.cn
　网址　http://www.ptpress.com.cn
　北京艺辉印刷有限公司印刷

◆ 开本：850×1168　1/32
　印张：7.25
　字数：268千字　　　2008年10月第1版
　印数：1－4 000册　　2008年10月北京第1次印刷

著作权合同登记号　图字：01-2007-5287号
ISBN 978-7-115-18438-2/TP

定价：29.00元

读者服务热线：(010)88593802　印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

Introduction

style: 1b. the shadow-producing pin of a sundial.
2c. the custom or plan followed in spelling, capitalization, punctuation, and typographic arrangement and display.

—*Webster's New Collegiate Dictionary*

The syntax of a programming language tells you what code it is possible to write — what machines will understand. Style tells you what you ought to write—what humans reading the code will understand. Code written with a consistent, simple style is maintainable, robust, and contains fewer bugs. Code written with no regard to style contains more bugs, and may simply be thrown away and rewritten rather than maintained.

Attending to style is particularly important when developing as a team. Consistent style facilitates communication, because it enables team members to read and understand each other's work more easily. In our experience, the value of consistent programming style grows exponentially with the number of people working with the code.

Our favorite style guides are classics: Strunk and White's *The Elements of Style*① and Kernighan and Plauger's *The Elements of Programming Style.*② These small books work because they are simple: a list of rules, each containing a brief explanation and examples of correct, and sometimes incorrect, use. We followed the same pattern in this book. This simple treatment—a series of rules—enabled us to keep this book short and easy to understand.

---

① Strunk, William Jr., and E. B. White. *The Elements of Style, Fourth Edition.* (Allyn & Bacon, 2000).

② Kernighan, Brian and P. J. Plauger. *The Elements of Programming Style.* (New York: McGraw-Hill, 1988).

引　　言

风格（style）：1b. 日晷上阴影产生的指针。

2c. 拼写、大小写、标点符号、排版和显示应该遵守的习惯或方法。

——《韦氏新版大学词典》

编程语言的语法告诉你可以写什么样的代码——机器可以理解的代码。而风格则告诉你应该编写怎样的代码——阅读代码的人可以理解的代码。采用一致、简单风格编写的代码可维护、健壮、更少缺陷。而不顾及一致风格的代码包含更多缺陷，最好是推翻重写而不是维护。

团队开发时，留心代码风格尤其重要。一致的代码风格有助于沟通，因为它让团队成员更容易阅读和理解他人的代码。以我们的经验来看，一致编程风格的价值随与代码有关的工作人员数量呈指数级递增。

我们钟爱经典的写作风格指南：Strunk和White的*The Elements of Style*①，还有Kernighan和Plauger的*The Elements of Programming Style*②。这些小书很有用，因为它们很简单：只是列出规则，每条规则都附上简洁的解释和正确用法范例，有时还举出错用的例子。我们这本书也如法炮制。这种简单的处理方式——只列出一系列规则——使本书言简意赅、容易理解。

① Strunk, William Jr., and E. B. White. *The Elements of Style, Fourth Edition*. (Allyn & Bacon, 2000).

② Kernighan, Brian and P. J. Plauger. *The Elements of Programming Style*. (New York: McGraw-Hill, 1988).

Some of the advice that you read here may seem obvious to you, particularly if you've been writing code for a long time. Others may disagree with some of our specific suggestions about formatting or indentation. The most important thing is consistency. What we've tried to do here is distill many decades of development experience into an easily accessible set of heuristics that encourage consistent coding practice (and hopefully help you avoid some coding traps along the way). The idea is to provide a clear standard to follow so programmers can spend their time on solving the problems of their customers instead of worrying about things like naming conventions and formatting.

The guidelines in this book complement the official .NET design guidelines in the ECMA C# specification[①] and Krzysztof Cwalina and Brad Abrams' excellent *Framework Design Guidelines*.[②] This book extends those guidelines to internal implementation and coding style.

Disclaimer

We have dramatically simplified the code samples used in this book to highlight the concepts related to a particular rule. In many cases, these code fragments do not conform to conventions described elsewhere in this book—they lack real documentation and fail to meet certain minimum declarative requirements. Do not treat these fragments as definitive examples of real code!

Acknowledgments

Books like these are necessarily a team effort. Major contributions came from the original authors of *The Elements of Java Style*: Al Vermeulen, Scott Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur, and Patrick Thompson, and the original authors of *The Elements of C++ Style*: Trevor Misfeldt, Greg Bumgardner, and Andrew Gray. Both of those books have some roots in "C++ Design, Implementation, and Style Guide," written by Tom Keffer, the "Rogue Wave Java Style Guide," and the "Ambysoft Inc. Coding Standards for Java," documents to which Jeremy Smith, Tom Keffer, Wayne Gramlich, Pete Handsman, and Cris Perdue all contributed.

① ECMA International, Standard ECMA-334: "C# Language Specification." 3rd Edition, June 2005. http://www.ecma-international.org/publications/standards/Ecma-334.htm.

② Cwalina, Krzysztof and Brad Abrams. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries.* Addison-Wesley, 2005. ISBN 0321246756.

对一些读者（尤其是已编程多年的读者）来说，本书中列出的一些建议也许是显而易见的。还有一些读者可能不同意我们关于代码格式化和缩进的建议。但最重要的是一致。我们尽力将数十年开发经验提炼成一套浅显近人、鼓励一致代码实践的总结性规则（希望能帮你规避一路上的编码陷阱）。本书旨在提出一种明晰的标准，以便程序员可将时间用在解决客户问题上，而无需挂虑于命名约定、代码格式化之类的琐事。

本书列出的指导是对ECMA C#规范[①]中的正式.NET设计指导以及Krzysztof Cwalina和Brad Abrams的优秀著作*Framework Design Guidelines*[②]一书的补充。本书将上述指导原则扩展到内部实现和编码风格。

声明

我们对本书所用的代码示例进行了极度简化，使之凸显具体规则的相关概念。这些代码片断多不适用于书中描述的其他约定规则——它们缺少文档说明，不满足最低的描述需求。别把这些片断错当成真实代码示例！

致谢

此类图书必是团队努力的成果。主要贡献来自于*The Elements of Java Style*一书的作者们，即Al Vermeulen、Scott Ambler、Greg Bumgardner、Eldon Metz、Trevor Misfeldt、Jim Shur和Patrick Thompson，以及*The Elements of C++ Style*一书的作者，即Trevor Misfeldt、Greg Bumgardner和Andrew Gray。这两本书都源自Tom Keffer所著*C++ Design, Implementation, and Style Guide*一书，以及Jeremy Smith、Tom Keffer、Wayne Gramlich、Pete Handsman和Cris Perdue共同编写的*Rogue Wave Java Guide*和*Ambysoft Inc. Coding Standards for Java*文档。

① ECMA International, Standard ECMA-334: “C# Language Specification.” 3rd Edition, June 2005. http://www.ecma-international.org/publications/standards/Ecma-334.htm.

② Cwalina, Krzysztof and Brad Abrams. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries.* Addison-Wesley, 2005. ISBN 0321246756.（中文版《.NET设计规范》，人民邮电出版社，2006）

Thanks also to the reviewers who provided valuable feedback on drafts of this book, particularly Brad Abrams, Krzysztof Cwalina, and Mark Vulfson of Microsoft Corporation; Mike Gunderloy of Larkware; and Michael Gerfen of Evolution Software Design.

This book would certainly never have happened without the help and encouragement of the folks at Cambridge University Press, particularly Jessica Farris and Lauren Cowles, who kept us on track throughout the writing and publication process.

还要感谢为本书初稿提出宝贵意见的审阅者们，特别是Brad Abrams、Krzysztof Cwalina和微软公司的Mark Vulfson、Larkware公司的Mike Gunderloy和Evolution Software Design公司的Michael Gerfen。

如果没有剑桥大学出版社同仁的帮助和鼓励，本书绝不可能问世。特别感谢Jessica Farris和Lauren Cowles，她们在整个写作和出版过程中让我们保持前进。

Preface

As commercial developers of software components, we always strive to have good, consistent style throughout our code. Since source code is usually included in our final products, our users often study our code to learn not just how the components work, but also how to write good software.

This fact ultimately led to the creation of *The Elements of Java Style*① and *The Elements of C++ Style*.② The positive reception of those books, coupled with recurring questions about C# and .NET style issues, resulted in this edition for C#.

If you've read the earlier books in this series (or even if you haven't), much of the advice in this book will probably be familiar. This is deliberate, as many of the programming principles described are timeless and valid across programming languages. However, the content has been reworked and expanded here to address the unique characteristics of the C# language.

Audience

We wrote this book for anyone writing C# code, but especially for programmers who are writing C# as part of a team. For a team to be effective, everyone must be able to read and understand everyone else's code. Having consistent style conventions is a good first step!

This book is not intended to teach you C#, but rather it focuses on how C# code can be written in order to maximize its effectiveness. We therefore assume you are already familiar with C# and object-oriented programming.

① Vermeulen, Al, et al. *The Elements of Java Style*. (Cambridge, UK: Cambridge University Press, 2000).

② Misfeldt, Trevor, Greg Bumgardner, and Andrew Gray. *The Elements of C++ Style*. (Cambridge, UK: Cambridge University Press, 2004).

前　　言

作为商业软件组件的开发者，我们总是努力在代码中保持良好、一致的风格。由于最终产品中总是会包括源代码，所以用户常常会研究我们的代码，不只是为了学习组件的工作机理，还为了学习如何编写好的软件。

这最终导致了*The Elements of Java Style*①和*The Elements of C++ Style*②两本书的诞生。这两本书深获好评，加上人们也总在问关于C#和.NET代码风格的问题，这样就促成了这个C#版本的问世。

无论你是否读过本系列图书的前两本，都可能会对本书中的许多建议耳熟能详。这是有意为之的，因为其中提到的许多编程原则在任何时候、任何编程语言中均适用。当然，我们重新整理了内容并做了扩展，以适应C#语言的独特个性。

读者对象

本书适用于编写C#代码的人，尤其是那些团队中的所有C#程序员。一个团队要想有成效，每个人都必须能阅读并理解其他人的代码。拥有一致的风格约定将是个良好的开始！

本书无意于教你C#，而是专注于应该如何编写最有效的C#代码。所以，我们假定你已经熟悉C#语言和面向对象编程。

---

① Vermeulen, Al, et al. *The Elements of Java Style*. (Cambridge, UK: Cambridge University Press, 2000).（英汉对照版《Java编程风格》，人民邮电出版社，2008）

② Misfeldt, Trevor, Greg Bumgardner, and Andrew Gray. *The Elements of C++ Style*. (Cambridge, UK: Cambridge University Press, 2004).（英汉对照版《C++编程风格》，人民邮电出版社，2008）

1

General Principles

While it is important to write software that performs well, many other issues should concern the professional developer. *Good* software gets the job done. But *great* software, written with a consistent style, is predictable, robust, maintainable, supportable, and extensible.

1. Adhere to the Style of the Original

When modifying existing software, your changes should follow the style of the original code.[①] Do not introduce a new coding style in a modification, and do not attempt to rewrite the old software just to make it match the new style. The use of different styles within a single source file produces code that is more difficult to read and comprehend. Rewriting old code simply to change its style may result in the introduction of costly yet avoidable defects.

2. Adhere to the Principle of Least Astonishment

The *Principle of Least Astonishment* suggests you should avoid doing things that would surprise other software developers. This implies that the means of interaction and the behavior exhibited by your software must be predictable and consistent,[②] and, if not, the documentation must clearly identify and justify any unusual patterns of use or behavior.

To minimize the chances that anyone would encounter something surprising in your software, you should emphasize the following characteristics in the design, implementation, packaging, and documentation of your software:

---

① Jim Karabatsos. "When does this document apply?" In "Visual Basic Programming Standards." (GUI Computing Ltd., 22 March 1996).

② George Brackett. "Class 6: Designing for Communication: Layout, Structure, Navigation for Nets and Webs." In "Course T525: Designing Educational Experiences for Networks and Webs."(Harvard Graduate School of Education, 26 August 1999).

第 1 章

一 般 原 则

编写性能良好的软件固然重要，但专业开发者还应注意其他许多问题。好的软件能完成任务，但以一致风格写成的卓越的软件则明晰、健壮、可维护、可支持且可扩展。

1. 保持原有风格

修改别人编写的软件时，应遵循原始代码的风格。[1] 修改时不要引入新风格，也不要仅为了吻合新风格而重写旧软件。在一个源代码文件中存在多种不同风格会使代码更难读懂。为修改风格而重写代码将产生本可避免的缺陷，增加软件成本。

2. 坚持最小惊奇原则

最小惊奇原则建议你避免做出可能让其他软件开发人员吃惊的事情。这意味着，软件所展示的互动及行为方式必须可预料并保持一致。[2] 如果不是这样，文档就必须清晰地指出所有非通常的用法或行为。

为了降低他人在使用你的软件时遭遇惊奇的可能性，你应当在软件的设计、实现、打包和文档中强调以下原则：

[1] Jim Karabatsos. “When does this document apply?” In “Visual Basic Programming Standards.” (GUI Computing Ltd., 22 March 1996).

[2] George Brackett. “Class 6: Designing for Communication: Layout, Structure, Navigation for Nets and Webs.” “Course T525: Designing Educational Experiences for Networks and Webs.”(Harvard Graduate School of Education, 26 August 1999).

Simplicity	Meet the expectations of your users with simple classes and simple methods.
Clarity	Ensure that each class, interface, method, variable, and object has a clear purpose. Explain where, when, why, and how to use each.
Completeness	Provide the minimum functionality that any reasonable user would expect to find and use. Create complete documentation; document all features and functionality.
Consistency	Similar entities should look and behave the same; dissimilar entities should look and behave differently. Create and apply consistent standards whenever possible.
Robustness	Provide predictable, documented behavior in response to errors and exceptions. Do not hide errors and do not force clients to detect errors.

3. Do It Right the First Time

Apply these rules to any code you write, not just code destined for production. More often than not, some piece of prototype or experimental code will make its way into a finished product, so you should anticipate this eventuality. Even if your code never makes it into production, someone else may still have to read it. Anyone who must look at your code will appreciate your professionalism and foresight at having consistently applied these rules from the start.

4. Document Any Deviations

No standard is perfect and no standard is universally applicable. Sometimes you will find yourself in a situation where you need to deviate from an established standard. Regardless, strive for clarity and consistency.

Before you decide to ignore a rule, you should first make sure you understand why the rule exists and what the consequences are if it is not applied. If you decide you must violate a rule, then document why you have done so.

This is the *prime directive*.

5. Consider Using a Code-Checking Tool to Enforce Coding Standards

A source code analysis tool enables you to check your code for compliance with

简单性	用简单的类和简单的方法满足用户期望。
清晰性	确保每个类、接口、方法、变量和对象都有清晰的目的。阐明何时、何处、如何使用它们。
完整性	提供任一可能的用户期望找到和使用的最小功能。创建完整的文档，描述所有特性和功能。
一致性	相似实体的外观和行为应该相同，不同实体的外观和行为应该不同。应该尽可能制定并遵守相关标准。
健壮性	对软件中可能出现的错误和异常做出预测，并将解决方法记入文档。不要隐藏错误，也不要等着用户去发现错误。

3. 第一次就做对

对所有代码实施这条规则，不仅限于正式产品的代码。原型或试验性代码多半会用到最终产品中，所以你该洞悉先机。即便代码永远不会采用到正式产品中，别人也有可能读到它。任何阅读你的代码的人，都会从你的专业性和贯彻这些规则的先见之明中获益匪浅。

4. 记录所有非规范行为

没有十全十美、普适一切的标准。有时你会有偏离某条既定标准的需要。无论如何也要努力保持清晰和一致。

在决定忽略某条规则之前，你应该先确信自己了解该条规则存在之理由，以及不采用该规则会引起的后果。如果你决定要违反某条规则，记录下这么做的原因。

这是第一守则。

5. 考虑采用代码检查工具强制遵循编码标准

可采用源代码分析工具检查代码是否符合编码标准和最佳实践。例

coding standards and best practices. For example, FxCop[①] is a popular .NET code analysis tool that uses reflection, MSIL parsing, and callgraph analysis to check for conformance to the .NET Framework design guidelines. FxCop is extensible, and can thus incorporate the particular coding standards used by your own organization.

① http://www.gotdotnet.com/team/fxcop/.