经 典 原 版 书 库

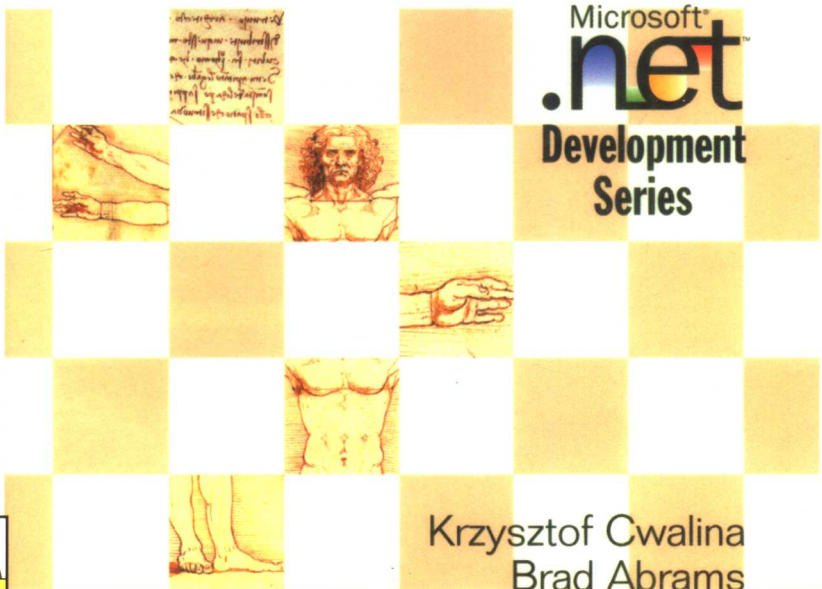
.NET设计规范

（英文版）

Foreword by Anders Hejlsberg
Distinguished Engineer, Microsoft Corporation

Framework Design Guidelines

Conventions, Idioms, and Patterns for Reusable .NET Libraries



Microsoft .net Development Series

Krzysztof Cwalina
Brad Abrams

附赠 DVD

（美） Krzysztof Cwalina
Brad Abrams 著

机械工业出版社
China Machine Press

经 典 原 版 书 库

.NET设计规范

（英文版）

Framework Design Guidelines

Conventions, Idioms, and Patterns for Reusable .NET Libraries

（美） Krzysztof Cwalina
Brad Abrams 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2006-5655

图书在版编目（CIP）数据

.NET设计规范（英文版）/（美）夸利纳（Cwalina, K.）等著. -北京：机械工业出版社，2007.1

（经典原版书库）

书名原文：Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries

ISBN 7-111-20203-1

Ⅰ.N… Ⅱ.夸… Ⅲ.计算机网络-程序设计-英文 Ⅳ.TP393

中国版本图书馆CIP数据核字（2006）第126023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

北京京北制版印刷厂印刷 · 新华书店北京发行所发行

2007年1月第1版第1次印刷

170mm × 242mm · 23.75印张

定价：56.00元（附光盘）

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

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范　明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅　宏	程　旭	程时端	谢希仁
裘宗燕	戴　葵			

To my wife, Ela,
for her support throughout the long process of writing this book,
and to my parents,
Jadwiga and Janusz, for their encouragement.
—Krzysztof Cwalina

■

To my wife, Tamara:
Your love and patience strengthen me.
—Brad Abrams

■

Foreword

In the early days of development of the .NET Framework, before it was even called that, I spent countless hours with members of the development teams reviewing designs to ensure that the final result would be a coherent platform. I have always felt that a key characteristic of a framework must be consistency. Once you understand one piece of the framework, the other pieces should be immediately familiar.

As you might expect from a large team of smart people, we had many differences of opinion—there is nothing like coding conventions to spark lively and heated debates. However, in the name of consistency, we gradually worked out our differences and codified the result into a common set of guidelines that allow programmers to understand and use the Framework easily.

Brad Abrams, and later Krzysztof Cwalina, helped capture these guidelines in a living document that has been continuously updated and refined during the past six years. The book you are holding is the result of their work.

The guidelines have served us well through three versions of the .NET Framework and numerous smaller projects, and they are guiding the development of WinFX, the next generation of APIs for the Microsoft Windows operating system.

With this book, I hope and expect that you will also be successful in making your frameworks, class libraries, and components easy to understand and use.

Good luck and happy designing.

Anders Hejlsberg
Redmond, WA
June 2005

Preface

This book, *Framework Design Guidelines*, presents best practices for designing frameworks, which are reusable object-oriented libraries. The guidelines are applicable to frameworks ranging in size and in their scale of reuse:

- Large system frameworks, such as the .NET Framework, usually consisting of thousands of types and used by millions of developers.
- Medium-size reusable layers of large distributed applications or extensions to system frameworks, such as the Web Services Enhancements.
- Small components shared among several applications; for example, a grid control library.

It is worth noting that this book focuses on design issues that directly affect the programmability of a framework (publicly accessible APIs). As a result, we generally do not cover much in terms of implementation details. Just like a user interface design book doesn't cover the details of how to implement hit testing, this book does not describe how to implement a binary sort, for example. This scope allows us to provide a definitive guide for framework designers instead of being yet another book about programming.

These guidelines were created in the early days of .NET Framework development. They started as a small set of naming and design conven-

tions but have been enhanced, scrutinized, and refined to a point where they are generally considered the canonical way to design frameworks at Microsoft. They carry the experience and cumulative wisdom of thousands of developer hours over three versions of the .NET Framework. We tried to avoid basing the text purely on some idealistic design philosophies, and we think its day-to-day use by development teams at Microsoft has made it an intensely pragmatic book.

The book contains many annotations that explain trade-offs, explain history, amplify, or provide critiquing views on the guidelines. These annotations are written by experienced framework designers, industry experts, and users. They are the stories from the trenches that add color and setting for many of the guidelines presented.

To make them more easily distinguished in text, namespace names, classes, interfaces, methods, properties, and types are set in `monospace` font.

The book assumes basic familiarity with .NET Framework programming. A few guidelines assume familiarity with features introduced in version 2.0 of the Framework. If you are looking for a good introduction to Framework programming, there are some excellent suggestions in the Suggested Reading List at the end of the book.

Guideline Presentation

The guidelines are organized as simple recommendations using **Do**, **Consider**, **Avoid**, and **Do not**. Each guideline describes either a good or bad practice and all have a consistent presentation. Good practices have a ✓ in front of them, and bad practices have an ✗ in front of them. The wording of each guideline also indicates how strong the recommendation is. For example, a **Do** guideline is one that should always[1] be followed (all examples are from this book):

✓ **DO** name custom attribute classes with the suffix "Attribute."

1. Always might be a bit too strong a word. There are guidelines that should literally be always followed, but they are extremely rare. On the other hand, you probably need to have a really unusual case for breaking a "Do" guideline and still have it be beneficial to the users of the framework.

```
public class ObsoleteAttribute : Attribute { … }
```

On the other hand, **Consider** guidelines should generally be followed, but if you fully understand the reasoning behind a guideline and have a good reason to not follow it anyway, you should not feel bad about breaking the rules:

✓ **CONSIDER** defining a struct instead of a class if instances of the type are small and commonly short-lived or are commonly embedded in other objects.

Similarly, **Do not** guidelines indicate something you should almost never do:

✗ **DO NOT** assign instances of mutable types to read-only fields.

Less strong, **Avoid** guidelines indicate that something is generally not a good idea, but there are known cases where breaking the rule makes sense:

✗ **AVOID** using `ICollection<T>` or `ICollection` as a parameter just to access the `Count` property.

Some more complex guidelines are followed with additional background information, illustrative code samples, and rationale:

✓ **DO** implement `IEquatable<T>` on value types.

The `Object.Equals` method on value types causes boxing and its default implementation is not very efficient because it uses reflection. `IEquatable<T>.Equals` can offer much better performance and can be implemented so it does not cause boxing.

```
public struct Int32 : IEquatable<Int32> {
 public bool Equals(Int32 other){ … }
}
```

Language Choice and Code Examples

One of the goals of the Common Language Runtime is to support a variety of programming languages: those provided by Microsoft, such as C++, VB, and C#, as well as third-party languages such as Eiffel, COBOL, Python, and others. Therefore, this book was written to be applicable to a broad set

of languages that can be used to develop and consume modern frameworks.

To reinforce the message of multilanguage framework design, we considered writing code examples using several different programming languages. However, we decided against this. We felt that using different languages would help to carry the philosophical message, but it could force readers to learn several new languages, which is not the objective of this book.

We decided to choose a single language that is most likely to be readable to the broadest range of developers. We picked C#, because it is a simple language from the C family of languages (C, C++, Java, and C#), a family with a rich history in framework development.

Choice of language is close to the hearts of many developers, and we offer apologies to those who are uncomfortable with our choice.

About This Book

This book offers guidelines for framework design from the top down.

Chapter 1 is a brief introduction to the book, describing the general philosophy of framework design. This is the only chapter without guidelines.

Chapter 2, "Framework Design Fundamentals," offers principles and guidelines that are fundamental to overall framework design.

Chapter 3, "Naming Guidelines," contains naming guidelines for various parts of a framework, such as namespaces, types, members, and common design idioms.

Chapter 4, "Type Design Guidelines," provides guidelines for the general design of types.

Chapter 5, "Member Design," takes it a step further and presents guidelines for the design of members of types.

Chapter 6, "Designing for Extensibility," presents issues and guidelines that are important to ensure appropriate extensibility in your framework.

Chapter 7, "Exceptions," presents guidelines for working with exceptions, the preferred error reporting mechanisms.

Chapter 8, "Usage Guidelines," contains guidelines for extending and using types that commonly appear in frameworks.

Chapter 9, "Common Design Patterns," offers guidelines and examples of common framework design patterns.

Appendix A contains a short description of coding conventions used in this book.

Appendix B describes a tool called FxCop. The tool can be used to analyze framework binaries for compliance with the guidelines described in this book. A link to the tool is included on the DVD that accompanies this book.

Appendix C is an example of an API specification that framework designers within Microsoft create when designing APIs.

Included with the book is a DVD that contains several hours of video presentations covering topics presented in this book by the authors, a sample API specification, and other useful resources.

Acknowledgments

This book, by its nature, is the collected wisdom of many hundreds of people, and we are deeply grateful to all of them.

Many people within Microsoft have worked long and hard, over a period of years, proposing, debating, and finally, writing many of these guidelines. Although it is impossible to name everyone who has been involved, a few deserve special mention: Chris Anderson, Erik Christensen, Jason Clark, Joe Duffy, Patrick Dussud, Anders Hejlsberg, Jim Miller, Michael Murray, Lance Olson, Eric Gunnerson, Dare Obasanjo, Steve Starck, and Kit George.

We also need to thank the many people who both reviewed and provided annotations to this book: Mark Alcazar, Chris Anderson, Christopher Brumme, Jason Clark, Steven Clarke, Joe Duffy, Patrick Dussud, Michael Fanning, Jan Gray, Brian Grunkemeyer, Eric Gunnerson, Anders Hejlsberg, Rico Mariani, Anthony Moore, Vance Morrison, Dare Obasanjo, Brian Pepin, Jon Pincus, Brent Rector, Chris Sells, Steve Starck, Herb Sutter, Clemens Szyperski, Jeffrey Richter, and Paul Vick.

Sheridan Harrison actually wrote and edited Appendix B on FxCop, which would not have been done without her time and skill.

For all of the help, reviews, and support, both technical and moral, we thank Martin Heller, the series editor for Addison-Wesley's Microsoft .NET Development Series. And for their insightful and helpful comments, we appreciate Pierre Nallet, George Byrkit, Khristof Falk, Paul Besley, Bill Wagner, and Peter Winkler.

John Montgomery sponsored this project, and it could not have been done without him.

We would also like to give special thanks to Susann Ragsdale who turned this book from a semi-random collection of disconnected thoughts into seamlessly flowing prose. Her flawless writing, patience, and fabulous sense of humor made the process of writing this book so much easier.

About the Authors

Krzysztof Cwalina is a program manager on the Common Language Runtime team at Microsoft. He started his career at Microsoft designing APIs for the first release of the .NET Framework. He has been responsible for several namespaces in the Framework, including `System.Collections`, `System.Diagnostics`, `System.Messaging`, and others. He was also one of the original members of the FxCop team. Currently, he is leading a companywide effort to develop, promote, and apply the design guidelines to the .NET Framework and WinFX. Krzysztof graduated with a B.S. and an M.S. in computer science from the University of Iowa.

Brad Abrams was a founding member of both the Common Language Runtime and .NET Framework teams at Microsoft Corporation, where he is a currently Lead Program Manager and has been designing parts of the .NET Framework since 1998. Brad started his framework design career building the Base Class Library (BCL) that ships as a core part of the .NET Framework. Brad was also the lead editor on the Common Language Specification (CLS), the .NET Framework Design Guidelines, and the libraries in the ECMA\ISO CLI Standard.

Brad has been involved from the beginning with the work on WinFX and Windows Vista. His primary role is ensuring that the consistency and developer productivity of the .NET Framework continues throughout Windows Vista and beyond.

Brad coauthored *Programming in the .NET Environment*, and was editor of *.NET Framework Standard Library Annotated Reference* (Volumes 1 and 2).