

谁能够让自己像浑浊的大水一样安静下来，
慢慢得到澄清？
——《老子》

更锋利的C#代码

——编写高质量C#程序



- 一个好的程序，不仅仅是能得出正确的运行结果。
- 每个章节的内容似乎都为大家所熟悉，然而视角却完全不同。
- 通过对那些几乎被人们忽视了的细节的精心处理，不断地提高每一行代码的质量。
- 它们为什么是必须的，而并非形式主义。
- C#提供的每种语言机制的功能背后，体现了怎样的逻辑含义。
- 读完此书，你会站在更高的角度对C#体系拥有更深的认识和把握。

包善东 著

清华大学出版社



原创经典

更锋利的 C# 代码

——编写高质量 C# 程序

包善东 著

清华大学出版社
北 京

内 容 简 介

一个好的程序，不仅仅是能得出正确的运行结果，而且还应在其内部保持清晰的代码逻辑和语义，否则，跟随在正常结果之后的也许是艰难的代码维护工作，对程序进行一处修改往往会牵一发而动全身，一不小心就会埋下深深的隐患。从另一个角度来说，如果每一行代码的质量都很高，那么这个软件产品也一定是高质量的。这就像 ISO 9000 的质量体系认证一样，与其在产品生产完成之后再进行检查，不如控制每一步生产环节的质量。

本书由浅入深、由表及里地讲述存在于 C# 编码开发中的各种质量问题，让读者清楚地了解什么是应该做的，什么是不应该做的。C# 提供的每种语言机制的功能背后，体现了怎样的逻辑含义。当遇到具体的问题时，应该如何选择与取舍。阅读完此书的每一个章节，都会让读者站在更高的角度 C# 体系拥有更深的认识和把握，不断向软件开发的更高层次迈进。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

图书在版编目 (CIP) 数据

更锋利的 C# 代码——编写高质量 C# 程序/包善东著. —北京：清华大学出版社，2008.10
ISBN 978-7-302-17942-9

I. 更… II. 包… III. C# 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2008) 第 092612 号

责任编辑：陈 冰

责任校对：徐俊伟

责任印制：王秀菊

出版发行：清华大学出版社

地 址：北京清华大学学研大厦 A 座

<http://www.tup.com.cn>

邮 编：100084

社 总 机：010-62770175

邮 购：010-62786544

投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者：北京鑫丰华彩印有限公司

装 订 者：三河市金元印装有限公司

经 销：全国新华书店

开 本：203×260 印 张：21.5 字 数：618 千字

版 次：2008 年 10 月第 1 版 印 次：2008 年 10 月第 1 次印刷

印 数：1~5000

定 价：49.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103 产品编号：027008-01

前言

这是一本以 C# 语言为基础，着眼于代码质量的编程指导。虽然没有砖头书的厚重，但却汇集了许许多多开发人员大量的实践经验。每个章节的内容似乎都为大家所熟悉，然而视角却完全不同，通过对那些几乎被人们忽视了的细节的精心处理，不断地提高每一行代码的质量。相信无论是 C# 初学者，还是具有 .NET 经验的开发人员，都能从本书中得到启发，写出质量更好的代码，开发出更加专业的程序。

为什么要写这本书？

很多人在编程时，仅仅追求其可以运行出正确的结果。这也许与学校中长期的数学式的编程教学方式有关，教学中涉及的都是些局部问题，即使有完整的程序，规模也非常有限，通常只考虑理想状况下算法与逻辑的正确性，很少去全面地考察它的工程质量。然而，当软件开发成为一种工业时，团队成员之间的高效合作成为首要的因素。任何一个具有一定规模的软件公司都深知：软件的工程质量往往比代码的算法效率更为重要。而现代化的质量管理并非仅仅是对产品的检验，更多的来自于过程的质量控制——用最佳的方式编写每一行代码，确保每一行代码都是高质量的。

可惜的是，涉及这方面知识的著作并不多，即使提及，也只是非常肤浅的一些关于缩进之类的书写格式上的建议。深入到代码之间的语义与逻辑关系层面，专门阐述代码质量的书可谓是凤毛麟角。难得的几部专著也大多是以 C/C++ 语言为主，很多规则及其背后的理由对于 C# 语言及 .NET 类库设计并不适用，读者在使用 C# 中的许多新特性（如索引器、事件、委托、异常等）之时也无法得到针对性的指导与帮助。本书不但紧密地结合了 C# 自身的特点，给出了丰富的原则性指导并进行了详细的解释，而且更具有实用操作性。

什么样的人应该读这本书？

本书虽然遍及 C# 的各种语法现象，但目的并不在于语言本身的教学，而是帮助读者发现那些影响代码质量的细节，并着手进行改进。对于初学者来说，可以配合相关的 C# 教材一同阅读；对于有经验的 C# 使用者来说，则可以作为一本开发过程中的手边参考。

如果您能遵照本书建议的原则进行开发，那么当别人看到您写出的代码或者发布的类库时，会相信该程序出自一位经验丰富的专业编程人员或团队，从而进一步对其产生充分的好感和信任。同时，代码阅读起来也非常易于理解，其他开发人员也能以最快的速度学习并掌握如何使用您开发的类库。因为您在编写每一行代码、定义每一个类型、设计每一个接口的时候，都从它的使用者的角度进行了充分的考

虑——即使这个使用者是您自己。良好的开发习惯不仅仅是为了方便他人，大多数的时间内，更重要的是方便了开发人员自己。这就如同对函数、类这些机制的使用一样，就算离开了团队开发，我们仍然会使用函数来理清结构并重用代码。同样，我们自己在一处写下的函数或类，终究还是会被自己在程序的另一个地方使用。高质量的程序编码无疑会给自己带来极大的便利——这种便利并不是可以让你在写程序时省事偷懒，而是可以不必将大量的精力耗费在弄懂一个星期之前写好的代码逻辑之上。

如果您是开发团队中的负责人、项目经理或总设计师，但你的团队缺少统一的开发规范，那么有了本书的内容作基础，您将可以轻松地建立起团队开发的编码标准和开发守则。并且您可以使用本书中所描述的充分的理由告诉您的团队成员，它们为什么是必须的，而并非形式主义。

本书内容

本书共 16 章，涵盖从 C# 代码风格（Coding Style）到国际化开发中的种种细节问题。第 1 至 3 章深入阐述了通用的代码书写格式与命名规范，第 4 至 7 章从面向过程的角度对开发质量准则进行了讨论，第 8 至 10 章则转向关于面向对象设计的指导原则，第 11 至 15 章针对 C# 及 .NET 所特有的机制进行相关讨论，第 16 章专门介绍了国际化应用开发时应当考虑的一些问题。由于前后章节的内容并没有严格的依赖关系，因此读者完全可以根据自己的兴趣与需要来决定阅读的顺序，而不必遵照原始章节顺序。

术语使用说明

术语的统一向来是计算机书刊的一个重要问题。同一个概念在不同时期、不同领域都可能有不同的具体表述方式。C# 是 Microsoft 公司专有的产品，并不像 C/C++ 那样开放，术语的统一相对较为容易。然而即便如此，仍有一些“官方”术语可能令人产生混淆。

重载与重写是最容易引起误解的一对术语，它可能与其他语言中类似概念的表述存在偏差。重载（Overload）指的是多个拥有不同函数签名的同名函数，相互通过参数的个数、数据类型、顺序上的差异进行区分。而重写（Override）指的是在派生类中对基类型中的虚成员或抽象成员提供自己的实现版本。

属性是另一个容易引起误解的术语，事实上，它是两个不同概念在中文里的重名翻译。大多数情况下，“属性”指“Property”，即结构或类的一种成员，通过 get/set 访问器进行读写。而“属性”有时所指的是“Attribute”，表示所有从 System.Attribute 派生的类型，用于对编程元素进行描述性声明。在下面的代码中：

```
[Serializable]
public class Employee
{
    private string name = null;
```

```
public string Name
{
    get { return name; }
    set { name = value; }
}
```

`Name` 是 `Employee` 类的一个 `Property`，而 `Serializable` 则是用于对 `Employee` 类进行补充声明的一个 `Attribute`。遗憾的是，官方并没有提供区分这两个概念的中文解决方案，好在 `Property` 使用的频率远远高于 `Attribute`。如果没有特殊的说明或者上下文的提示，本书中所有的属性均指 `Property`。

为了准确地表达术语，在本书中，所有术语在正文中被首次提及时，或者其他必要的位置，都会附带相应的英文表述，供读者参考。

排版约定

本书中包含大量的 `C#` 代码示例，为了便于阅读，在排版时使用了类似代码加亮的方式：关键字以**粗体**方式显示，而伪代码则用*斜体*显示。注意：为了加以区别，非 `C#` 语言的代码示例不使用代码加亮格式。

在整体上，本书还采用了带底纹与无底纹两种格式。带底纹的代码示例表示正确或推荐的做法，并附有示例编号与小标题；而无底纹的代码则表示错误或不当的做法，以及一般的补充说明之用。请看下面的示例：

```
Console.WriteLine("这是不带底纹的代码示例。");
```

以及，

代码示例 x-x：带底纹的代码示例

```
Console.WriteLine("这是带底纹的代码示例。");
```

关于截图的使用

书中所有 Microsoft Visual Studio 软件的屏幕截图的使用符合微软公司的相关许可规定。

致谢

这本书的创作过程真可谓是一场战斗，回想我写过的大大小小的程序、网站和文章，这几乎是时间

跨度最大的一桩。从 2006 年底开始构思、进行内容和框架整理，到 2007 年夏天开始正式动工，再到现在整本书的完稿、出版，自己都觉得不可思议。在我看来，时间如此之长主要有两个原因。一是这本书的内容主要来源于经验的总结、归纳与提炼，并不像一般的技术教程写起来那么顺畅。很多时候，内容要点本身的整理就占据了很多时间，更不要说如何将它们表述清楚了。另一个原因是客观的，这本书开工后不久，工作就越来越忙，每天能够用于专心写书的时间越来越少。以至于到 2007 年底时，实际进度已经比计划慢了整整一个月。回想这半年多的时间里，多少次觉得精疲力尽难以继续，而又是在亲人和朋友们的多少次鼓励中坚持了下来。毫不夸张地说，他们对这本书的完稿起到了决定性的作用。

在此，我要感谢我的父亲包宗顺、母亲瞿忆玲、以及我的女友应筠之。他们与我朝夕相处，对我的写书工作给予了全力的支持。

我想对清华大学出版社的编辑陈冰，以及参与本书编辑出版的所有工作人员表示致意。如果没有他们对我的信任与支持、以及认真负责的工作，我就无法通过这本书来与大家分享我的经验。

感谢胡妍妍同学，她帮我指出并改正了许多语言表述上的错误，为这本书的“中文编码质量”作出了极大的贡献。我还要感谢杜文远、郑明鉴、以及从未谋面的何晓杰，他们提出了许多宝贵的意见和建议，并在本书创作最困难的阶段给予我关心与鼓励。

包善东

2008 年 6 月 30 日

目 录

第 1 章 基本的代码风格

1.1	换行的讲究	1
1.1.1	寻找最佳的断行位置.....	2
1.1.2	每行只写一条语句.....	4
1.1.3	分行定义变量.....	4
1.2	避免代码过于拥挤	5
1.2.1	使用空行分隔代码块.....	5
1.2.2	使用空格降低代码密度.....	8
1.3	如何缩进	10
1.3.1	嵌套或包含关系引起的缩进.....	11
1.3.2	因换行而产生的缩进.....	14
1.3.3	使用空格还是 Tab 键.....	15
1.4	大括号	15
1.4.1	大括号的位置.....	16
1.4.2	空的大括号结构.....	17
1.4.3	仅包含单个语句的结构体.....	19
1.5	保持项目文件的条理性	21
1.5.1	解决方案的结构呼应.....	21
1.5.2	代码文件的结构.....	22
1.5.3	使用#region 标记来隐藏细节.....	24

第 2 章 养成良好的注释习惯

2.1	何时需要注释	25
2.1.1	解释代码的意图.....	26
2.1.2	对局部变量的说明.....	26

2.1.3	充当代码标题	27
2.1.4	指出例外情况	30
2.1.5	开发过程的提示	30
2.2	注释的格式	31
2.2.1	单行注释	32
2.2.2	多行注释	33
2.3	正确使用 XML 文档注释	34
2.3.1	结构与类的 XML 文档注释	36
2.3.2	属性的 XML 文档注释	37
2.3.3	方法的 XML 文档注释	38
2.3.4	构造函数的 XML 文档注释	39
2.3.5	事件的 XML 文档注释	40
2.3.6	枚举类型的 XML 文档注释	41
2.3.7	泛型的 XML 文档注释	42
2.3.8	其他标记	43

第 3 章 一般命名规范

3.1	选用合适的名称	45
3.1.1	使用字符的限制	46
3.1.2	使用含义明确的英语	47
3.2	大小写规则	49
3.2.1	Pascal 规则	49
3.2.2	Camel 规则	49
3.2.3	首字母缩写词与简写词	50
3.2.4	应在何时使用何种大小写规则	52
3.3	考虑跨语言编程	54
3.3.1	不要通过大小写区分标识符	54
3.3.2	避免与其他语言的关键字重复	55
3.3.3	避免使用特定语言的术语	55
3.4	命名一致与冲突	57
3.4.1	大小写无关原则	57
3.4.2	对基类型的命名暗示	58

3.4.3 对参数与属性的关系暗示	60
3.4.4 属性名称与自身类型同名	62
3.4.5 与命名空间相关的命名冲突	63
3.5 匈牙利命名法	65
3.5.1 匈牙利命名法的弊端	66
3.5.2 考虑为控件应用匈牙利命名法	67

第 4 章 处理数据

4.1 关于数据类型	69
4.1.1 整数	69
4.1.2 浮点数	71
4.1.3 布尔类型	74
4.1.4 字符与字符串	74
4.2 变量的使用	77
4.2.1 尽可能使用内置关键字	77
4.2.2 初始化一切变量	78
4.2.3 集中使用变量	79
4.3 使用枚举	83
4.3.1 何时使用枚举	84
4.3.2 如何为枚举命名	87
4.3.3 关于枚举项	89
4.3.4 标记枚举	90
4.4 魔数——以字面数值出现在代码中的常量	92
4.5 复杂的表达式	93
4.5.1 运算符的副作用	94
4.5.2 简化表达式	95

第 5 章 分支结构

5.1 使用 if 结构	98
5.1.1 “==”与“=”的问题	98

5.1.2	如何处理复杂的条件	100
5.2	使用 switch 结构	103
5.2.1	break 语句	103
5.2.2	使用 default 子句要注意的问题	105
5.3	选择 if 还是 switch?	107
5.4	关于判断顺序的设计	109
5.4.1	先判断最有可能成立的条件	110
5.4.2	预防因条件短路而丢失操作	113
5.5	慎用 goto 语句	114

第 6 章 循环结构

6.1	使用 for 还是 while	117
6.1.1	for 和 while 的语义比较	117
6.1.2	简单的数值迭代——for 和 while 的思维模式的差异	118
6.1.3	预知循环次数——微波炉加热的启示	121
6.1.4	集合迭代——独特的 foreach 结构	122
6.2	循环变量的使用	123
6.2.1	循环变量的命名	123
6.2.2	循环变量的定义	124
6.2.3	避免循环变量的非常规应用	125
6.3	提高循环效率	127
6.3.1	避免不必要的重复劳动	127
6.3.2	避免不必要的循环	127

第 7 章 如何使用函数

7.1	为什么要使用函数	129
7.1.1	函数与方法	129
7.1.2	代码复用	130
7.1.3	隐藏细节——使用函数进行抽象	132

7.2	函数重载	135
7.2.1	重载的语义——为调用者提供方便	135
7.2.2	保持核心代码唯一	139
7.3	参数的设计	142
7.3.1	参数的命名	142
7.3.2	不要使用保留项	142
7.3.3	何时使用变长参数列表	143
7.3.4	何时使用 <code>ref</code> 参数和 <code>out</code> 参数	145
7.3.5	参数的顺序	146
7.3.6	重载函数的参数一致性体现	146
7.4	参数检查的必要性	150
7.4.1	检查零值及空引用	150
7.4.2	检查枚举类型	151
7.4.3	防止数据被篡改	152
7.4.4	在何处检查合法性	153
7.5	函数的出口——离开函数的三种方式	154
7.5.1	返回值的用法	155
7.5.2	离开函数的时机	156

第 8 章 结构与类

8.1	结构与类的比较	159
8.1.1	值类型与引用类型	159
8.1.2	何时应当使用结构	160
8.1.3	何时应当使用类	160
8.2	结构与类的命名	161
8.2.1	措辞	161
8.2.2	避免与命名空间冲突	161
8.2.3	不要使用“C”前缀	162
8.2.4	后缀的使用	162
8.3	如何搭建一个典型的结构	164
8.3.1	找准核心数据	164

8.3.2	数据的表现形式	165
8.3.3	定义等价原则	166
8.3.4	实现基本运算	168
8.4	如何真正面向对象	169
第 9 章 封装		
9.1	构造函数	175
9.1.1	构造函数的语义	175
9.1.2	何时使用静态构造方法	177
9.1.3	构造函数的参数及其初始化	178
9.2	Finalize 函数	181
9.2.1	垃圾回收器	181
9.2.2	IDisposable 接口——显式释放资源的方法	181
9.2.3	释放资源的一般范式	184
9.3	何时应该使用字段	185
9.3.1	存储核心数据	186
9.3.2	维持中间结果	187
9.3.3	常量字段	188
9.4	如何使用字段	189
9.4.1	字段的命名	189
9.4.2	访问控制	190
9.5	何时应该使用属性	192
9.5.1	属性的语义	192
9.5.2	数据访问控制	193
9.5.3	需要的后续操作	194
9.5.4	简单的数据处理	194
9.5.5	预定义的对象实例	195
9.6	如何使用属性	195
9.6.1	属性的命名	195
9.6.2	访问控制	198
9.6.3	提供合理的默认值	199
9.6.4	保持轻量级的操作	199

9.6.5 在属性中抛出异常	199
9.7 何时应该使用方法	200
9.7.1 表示某种操作	200
9.7.2 耗时的任务——方法在形式上的暗示作用	201
9.7.3 有副作用的操作	202
9.7.4 返回不确定的值	203
9.7.5 返回数组或集合对象	203
9.8 如何使用方法	205
9.8.1 方法的命名	206
9.8.2 检查传入的参数	206
9.9 静态类型及成员	206
9.10 嵌套类型及其适用场合	206
9.11 可变类型的安全性	208
9.12 使用程序集与命名空间	210
9.12.1 程序集的划分	210
9.12.2 为什么要使用命名空间	210
9.12.3 命名空间的命名	211
9.12.4 命名空间的管理	212

第 10 章 继承与多态

10.1 如何利用类继承	214
10.1.1 自上而下逐步细化	214
10.1.2 自下而上逐步抽象	217
10.2 继承限制	222
10.2.1 强制继承的抽象类型	222
10.2.2 密封类型	222
10.2.3 扩展方法——直接向已有类型添加功能	223
10.3 关于接口	224
10.3.1 接口的语义	224

10.3.2	接口的命名	225
10.3.3	使用接口还是类继承	225
10.4	何时应当显式实现接口	227
10.4.1	解决接口之间的命名冲突	227
10.4.2	提供强类型操作	228
10.4.3	隐藏仅用于通过接口访问的成员	229
10.5	使用多态	230
10.5.1	何时应进行重写	230
10.5.2	应当重写哪个成员	230
10.5.3	保持参数名称一致	233
10.6	运算符重载	233
10.6.1	可重载的运算符	233
10.6.2	符合运算符的本意	235
10.6.3	运算符的关联性	235
10.6.4	类型转换运算符的重载	236

第 11 章 泛型机制

11.1	装箱与取消装箱	237
11.2	何时使用泛型	238
11.3	泛型的类型参数设计	239
11.3.1	类型参数的命名	239
11.3.2	使用类型参数的时机	241

第 12 章 事件与委托

12.1	何为事件驱动模式	244
12.2	如何响应事件	245
12.2.1	事件处理函数	246
12.2.2	代码的分配	248

12.2.3	事件侦听器使用	250
12.3	如何提供事件	255
12.3.1	何时应当提供事件	256
12.3.2	事件的命名	256
12.3.3	传递与事件相关的数据	257
12.3.4	用于事件的委托及其要遵守的约定	259
12.3.5	触发事件	260
12.4	使用委托	261
12.4.1	何时使用委托	261
12.4.2	何时使用匿名方法	261
12.4.3	基类型与派生类型	263

第 13 章 集合类型

13.1	系统内置集合类型	264
13.1.1	数组	264
13.1.2	列表	266
13.1.3	字典	266
13.1.4	其他类型	267
13.2	选用适当的集合类型要考虑的几个方面	267
13.2.1	容量	267
13.2.2	进出次序	268
13.2.3	定位的问题——索引/键访问	268
13.2.4	元素结构	269
13.2.5	排序	271
13.3	性能比较	272
13.4	提供自己的集合类型	274
13.4.1	何时应提供集合类型	274
13.4.2	集合类型的命名	276
13.4.3	提供与内置集合类型一致的行为	276
13.4.4	索引器及其应遵守的规则	278
13.4.5	迭代器	279

第 14 章 LINQ 查询

14.1	提高 LINQ 查询的效率	282
14.1.1	查询语法和方法语法的区别	282
14.1.2	LINQ 查询的创建、执行与性能	284
14.1.3	减少返回的数据量	285
14.2	LINQ 中的错误处理——采用防御式编程	286
14.3	LINQ 查询的相关机制	288
14.3.1	匿名类型	288
14.3.2	隐式类型的局部变量	289
14.3.3	Lambda 表达式与匿名函数	290

第 15 章 异常

15.1	处理异常时应遵守的规范	292
15.2	抛出异常	296
15.2.1	异常的语义	296
15.2.2	不应使用异常的位置	298
15.2.3	控制异常	298
15.2.4	异常的重新抛出——重新包装时要注意的	300
15.3	选用合适的异常类型	303
15.3.1	常见的异常类型	303
15.3.2	不应使用的异常类型	304
15.4	异常提示信息	305
15.5	设计自定义异常及应遵循的约定	305

第 16 章 全球化与本地化

16.1	分离与特定区域相关的信息	309
-------------	---------------------	-----