微软.NET程序员系列

Microsoft Press

- 欧美读者评价 ★★★★★
- 权威专家精心编著
- 典型示例 贯穿全书
- 可重用组件开发全攻略
- 中高级程序员必备参考

MICROSOFT®

Visual Basic .NET 编程标准

Practical Standards for Microsoft Visual Basic .NET

[美] James Foxall 著

附昭伟 译

Visual Studio .NET产品组 审校

Microsoft .net

清华大学出版社

微软.NET 程序员系列

Visual Basic .NET 编程标准

[美] James Foxall 著
附昭伟 译
Visual Studio .NET 产品组 审校

清 华 大 学 出 版 社
北 京

内 容 简 介

本书介绍如何在 Visual Basic .NET 中进行规范化编程，提供对不严谨的编程风格进行改进的方法。本书分为 6 大部分，分别介绍了设计、规则、编码结构、高级编程、用户界面以及团队项目这几个方面的内容。这些内容说明使用 Visual Basic .NET 进行编程时将代码规范化的实际方法，以便若干年后人们仍能容易地理解它。本书的各章都提供了典型的示例以说明问题。

本书适用于 Visual Basic .NET 编程人员阅读。不论读者只编写过很少的代码还是在团队中开发企业级应用程序，都能从本书中学到如何利用规范化标准为进程开发更优秀更可靠的代码。

图书在版编目(CIP)数据

Visual Basic .NET 编程标准/(美)福克斯奥著；附昭伟译.—北京：清华大学出版社，2003
(微软.NET 程序员系列)
书名原文：Practical Standards for Microsoft Visual Basic .NET
ISBN 7-302-06608-6

I.V… II.①福…②附… III.BASIC 语言－程序设计 IV.TP312

中国版本图书馆 CIP 数据核字(2003)第 032608 号

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.com.cn
责任编辑：赵飞虎
印 刷 者：清华大学印刷厂
发 行 者：新华书店总店北京发行所
开 本：787×960 1/16 印张：23.5 彩插页：5 字数：557 千字
版 次：2003 年 6 月第 1 版 2003 年 6 月第 1 次印刷
书 号：ISBN 7-302-06608-6/TP·4949
印 数：0001～4000
定 价：39.00 元

《微软.NET 程序员系列》序

自 2000 年 6 月微软宣布自己的.NET 战略以来，在不到两年的时间里，.NET 已经从战略变成现实。.NET 带来了全新的、快速而敏捷的企业计算能力，也给软件开发商和软件开发人员提供了支持未来计算的高效 Web 服务开发工具。作为微软.NET 战略的重要组成部分——Visual Studio .NET (中文版)已经于 2002 年 3 月 22 日正式在中国推出。

Visual Studio .NET 是一个功能强大、高效并且可扩展的编程环境。它充分展现了应用程序开发的潜能，并提供了生成应用程序所需的工具和技术。这些应用程序将给当今的企业、机构提供强大的支持，并推动下一代基于 XML Web 服务软件的发展。

有了 Visual Studio .NET，那些对全世界数百万的专业和业余程序员来说，曾一度极端复杂、费时费力，甚至让人望而生畏的编程任务现在已不再神秘。更重要的是，Visual Studio .NET 使开发人员能运用既有的技能和知识来迎接新的编程挑战。

在 10 年前，Visual Basic 1.0 成为数以百万计的开发人员的革命性的应用程序开发语言。现在，Visual Studio .NET 为未来的 10 年做好了准备。

微软出版社为了配合 Visual Studio .NET 的推广以及.NET 技术的普及，邀请 Visual Studio .NET 项目开发组的核心开发人员和计算机图书专业作家精心编写了英文版《微软.NET 程序员系列》丛书。该丛书自面市以来，在美国图书销量排行榜上一直高居前列，颇受好评，成为程序开发人员和网络开发人员了解.NET 技术的权威工具书。尤其是《Microsoft .NET Framework 程序设计》一书，长期占据美国及欧洲此类书籍的排行榜冠军位置，程序开发人员不可不读此书。

清华大学出版社为了满足中国广大程序开发人员、网络开发人员学习最新技术的渴望，在微软出版社的配合下，从《微软.NET 程序员系列》这套丛书中精选了 50 余本翻译成中文，以满足国内广大读者的需要。这套丛书阵容庞大(且在不断扩充之中)，几乎涵盖了.NET 技术及其应用的各个方面；也正因为如此，翻译和编辑加工的工作量也大得惊人。但为了保持国外优秀技术图书的魅力，同时使读者领会新技术的真谛，本丛书的翻译和编辑都是经过严格筛选的、具有很高的翻译水平或丰富编辑经验的技术人员；另外，我们还聘请微软公司 Visual Studio .NET 产品组的技术专家审读每一本书，确保在技术上准确无误。

相信这套丛书定会帮助程序开发人员、网络开发人员以及那些具有一定编程基础的中、高级读者，快速、全面地掌握.NET 技术，协助他们为技术生涯的下一个 10 年做好准备，为培养新一代软件人才，并推动中国软件产业的快速发展起到积极的作用！

这套丛书分为 3 个子系列：技术内幕系列、语言参考系列和程序员系列。目前，已出版和在编的共有 36 本，已从 2002 年 6 月份起陆续和读者见面。

- **技术内幕系列**

  目前共有 7 本：

  - 《Visual C++ .NET 技术内幕(第 6 版)》

    本书是 Visual C++和 MFC 开发的经典著作。它秉承了第 4 版和第 5 版的风格，已根据该编程语言的最新版本 Visual C++ .NET 进行了全面更新和补充，是.NET 时代的 C++程序员必读的教材。此外，本版仍由第 4 版译者潘爱民先生翻译。

  - 《Microsoft .NET Compact Framework 技术内幕》
  - 《Visual Basic .NET 技术内幕》
  - 《Visual C# .NET 技术内幕》
  - 《ADO.NET 技术内幕》
  - 《Microsoft .NET 程序设计技术内幕》
  - 《Visual J# .NET 技术内幕》

- **语言参考系列**

  目前共有 3 本：

  - 《Visual Basic .NET 语言参考手册》
  - 《Visual C# .NET 语言参考手册》
  - 《Visual C++ .NET 语言参考手册》

- **程序员系列**

  目前共有 27 本。详细书目，请参见本书彩插页。

  其中，Microsoft .NET Framework 1.1 类库(1～4 卷)是.NET System 命名空间大全，直接源自 Microsoft .NET Framework。这 4 卷书包含了.NET Framework Class Library System 命名空间的全部详细信息。帮助开发人员充分利用.NET Framework 来开发 Web 应用程序、客户端应用程序和 XML Web 服务。

随着技术的发展，我们将根据读者的需要，不断增加新的书目。

丛书版式特色

本丛书在风格上力求文字精炼。并采用小 5 号字编排，内容紧凑，版面清晰美观，易于阅读。此外，书中还安排了一些特色段落，提供正文之外的一些细节知识：

注意：提醒阅读和操作过程中应注意的事项，避免出现错误或问题。

提示：指点一些操作捷径或实用技巧，使您少走弯路，阅读和操作更为高效。

要点：总结关键知识点或操作细节，助您适时掌握要领。

注：提示首次出现的编程元素，以及书中涉及到该元素的其他位置以供参考。

尽管我们倾心相注，精心而为，总有疏忽纰漏，恳请广大读者不吝赐教与指正，我们定会全力改进，以期在后续工作中得以完善。

本丛书在创作过程中得到了微软(中国)公司的大力支持。本丛书能够顺利出版，更是倾注了无数幕后人员的汗水和心力。在此，对他们的辛勤劳动一并表示衷心感谢！

编者

2003 年 3 月

前　言

作为功能化社会的一员，人们每天都要接触很多标准。人们通常不会注意到这些标准，然而人们认可这些标准并且自然的运用它们。例如，当开车的人接近十字路口时会停下自己的汽车并观察交通状况。一旦十字路口，开车的人会确定谁有通过的权力(在右边首先到达停止标记的汽车)并且在适当的时候通过路口。这种标准的运用正如汽车通过每天可能要发生 50 次。人们可能在吃油炸圈饼、喝咖啡、打电话(现在在许多地方这会被罚款)或与乘客交谈时做这件事。最初学开车时，为熟悉这一规则会花费很多精力，然而一旦习惯了这些规则，这种过程就会成为一种条件反射。程序的标准也应该与此类似。

与人们的日常生活相关的标准之一是 110V 电压。当把一个新的电器接到墙上的插座时，不必去查看电器的说明书，也不必用电压表测量插座以确定电器和插座是否匹配。人们相信两个设备都符合严格的标准。如果设备使用不满 110V 的电压，可以用变压器之类的设备来降低功率，这样设备就能够连接到 110V 的系统上了。如果需要更高的功率，例如干衣器需要 220V 电压，它的插头的尺寸和形状很特殊，这样它就不能和 110V 的系统相匹配。显然，如果不符合标准会导致伤亡事故。这些标准如此重要以至于厂商会认证其产品而且确保产品符合必要的规范，并通过了必要的安全检查，这就是为什么几乎所有的电子产品会带有 UL(Underwriters Laboratories)标识。

使用接口和编程标准不会使某人免于遭电击或阻止两辆汽车在路口相撞，但是严格遵循一套已制订的标准会带来许多好处。遵照一套标准所编写的代码易于维护和升级，并且正确地使用公认的标准通常会改善代码，研究表明符合标准的代码错误通常较少。

标准应用于用户接口的优秀示例是 Office 套件。不管接口是否令人喜欢，它易于理解并且能体现可视化标准套件带给最终用户的好处。没有什么事情是完美的，虽然 Office 接口发生了一些可喜的变化，但当本书作者编写第一个兼容 Office 的商用程序时便认识到接口的益处，接口是应用标准的直接结果。由于过去使用工具时不能创建所需的确定功能，为了得到软件的发行证书而进行了很多工作，甚至需要弃权。此工具绝不是 Visual Basic，但却是 Microsoft 的产品。我与编程的同事们编写了一些优秀的代码而并不想编写说明书。我们在工具栏、菜单与加速键上花费了大量的时间。当用户开始使用我们的产品时我们获得很多赞誉。

我们的用户说为实现应用程序所进行的内部培训与支持远比预计的要低。用户能将注意力集中在程序的功能与特性上，而不是接口的机制上，例如导航。当用户需要打印时，他知道到哪里去找 Print 按钮。此按钮的位置与 Office 中一样，它会显示用户所熟悉的打印机图标。同样，用户熟悉 Office 的标准热键，知道可以通过按下 Ctrl+P 进行打印。最终，开发最初所作的投入让用户受益。对应用程序的界面来说，标准提供的一致性是关键。尽管看起来不明显，一致性与标准也是编写代码的关键。

本书的必要性

与其他开发语言相比 Visual Basic 有世界上最大的用户群体，并且还在不断增长。Visual Basic 的用户大概有 300 万。Visual Basic 过去曾经被作为计算机业余爱好者的语言，但后来迅速得到开发人员的关注，现在它已经被广泛应用于公司与其他商业环境。Visual Basic .NET 的功能更加强大，它与 Visual C#以及其他.NET 语言共享公共语言运行库。它们也共用相同的 IDE 和调试工具等。过去有很多关于使用 Visual Basic 建立企业解决方案的争论。许多争论围绕 Visual Basic 快速应用程序开发的特性与 C++强大功能的特性进行深入探讨。Visual Basic 的能力令人惊奇，但是为了使用 Window 的"底层"功能就必须使用 API 调用以及第 3 方组件。现在，竞争是在同一个层次上了。Visual Basic .NET 的强大灵活性以及其对向后兼容性的重视，使它成为很强大的语言。令人遗憾的是这些特性也使程序变得冗长，现在的快速开发环境仍然不能使之简化。

例如，Visual Basic .NET 确实可以创建 voodoo 变量。voodoo 变量仅仅因为其名称被引用而被凭空创建出来。如果没有将 Option Explicit 语句加入模块或在项目一级设置 Option Explicit，就可以通过引用未经声明或使用的变量名来创建 voodoo 变量。voodoo 变量通常因为错误而创建，并且非常难调试。Visual Basic .NET 不应该允许编程者能够关闭对变量显示声明的限制条件。这项功能具有向后兼容性，但它不适用于现代编程，而为娱乐编写程序或编写分布式程序时可以不受这个条件的限制。

许多开发人员这几年由于使用详尽但冗长的代码而获得好处，这些代码包括自己编写的代码也包括团队中其他人编写的代码。欠缺编程技术会带来很多不良影响。例如，您是否经历过对过程进行很小的修改但终因代码不容易修改(扩展)而必须将整个过程重写?当例程中的进程被同时修改时，如果它们能够被修改，则改善它们会显得尤其困难。从另一方面来看，编写出正确的代码会在需要修改代码时使工作更容易进行。

您过去是否发现修改别人的代码时需要几小时或几天甚至几个星期来理解特定的进程如何工作？您是否曾经在一段时间之后再查看自己的代码时却发现由于自己写的注释太少而忘记进程是如何实现的？当您被迫处理自己留下的麻烦时，标准化的益处便显而易见。按照严格标准编写的代码包括了足够多的注释，这样除了便于了解如何完成任务以外，还便于了解过程中发生的事。代码的外观，例如它们的缩进与空格的使用，和合适的变量名称也能使代码更易于理解。

许多开发人员仍不能对变量的数据类型给予足够的重视。这个问题包括无须使用 Object 类型或需要用 Double 数据类型时却用了 Single 类型等。在这些情况下，Visual Basic 可能会进行强制类型转换或进行无法预计的舍入处理，这样就会导致意料之外的结果出现。例如，如果将一个小数放入 Integer 变量，小数部分会被舍入而不会出现错误。然而，如果将一个比变量数据类型允许值大的值放入变量，便会出现运行时错误。设计时，必须通过程序放入大量的测试数据才能对某些条件进行测试。必须从一开始就正确的定义变量，否则，直到客户的计算机出现问题时才会知道。Visual Basic .NET 包括一

个新的设置项 Option Strict，它强迫进行严格的变量声明与使用，应该尽可能将它打开。然而，使用 Option Strict 会增加必要的代码(还有精力)，然而，这样做是值得的。

开发人员一直优先考虑性能。因此，尽管计算能力在明显提高，但开发人员应该知道不是每个人都在使用和自己开发时所用计算机一样的主流配置计算机。按照一定标准编写代码通常比不按标准编写的代码运行的快，例如严格的数据类型以及对于给定任务使用合适的循环语句。

不遵循恰当的编码标准的另一个后果是针对开发人员的：由于不严格的编码他可能得不到工作甚至会失业。作为 Visual Basic 程序员的雇主，我发现大多数落选的申请者是因为他们的代码而丧失资格，而不是因为他们的简历。我查阅申请到我们公司工作的申请人的代码示例。最常见的问题是代码缺少足够的精确注释，而这是形成劣质代码的前兆。我所看到的情况真是令我感到震惊与同情。我所查阅的代码中没有一个变量是明确注释了的，voodoo 变量随处可见。我所查阅的代码中所有的变量都被声明为模块级变量。出于好奇心我随机对某些变量进行搜索。其中大多数变量只在一个过程中使用过。很多变量并没有被到处使用。当我质问代码的作者时，他告诉我，他觉得将来可能会使用这些变量，他的计划多么超前啊。如果这些例子对您来说听起来不像是缺乏编程经验，那么这本书对您来说将很有价值。

如果您认为我吹毛求疵，您可能是正确的。然而，我并不孤立。许多人认识到按照一致的编程标准写出优秀健全代码的人几乎能够理解任何进程的代码。如果您正在找工作，那么请将本书中的原则默记在心中，并且将代码修改之后再发送给未来的雇主审阅。

如前所述，Visual Basic 强大而且灵活并且具有内置的向后兼容性(Visual Basic .NET 确实比旧版本少的多)，使之产生了很多问题。如果很多 Visual Basic 程序员是自学而成，并且缺乏正式的开发训练(这种训练能使他们认识到低劣编程技术的缺陷)，这些问题就会更加突出。Visual Basic .NET 是一种新语言，它不同于以往所有版本的 Visual Basic，这并不是缺陷。您只是不可能再像旧版一样找到什么捷径。这些事实说明需要发布标准。

尽管使用标准化技术开发的应用程序包含的错误极少，并且易于维护与升级(这难以完整说明)，但一些程序员仍然反对标准化。遗憾的是，某些争论通常是由于自以为是、缺乏理解、反对变革或仅仅是因为懒惰造成的。例如，声称为变量添加 3 字符命名前缀以说明其数据类型的做法太费力，就像是说将变量命名为 TotalStudents 比将其命名为 X 更费事一样。当然，输入 X 比输入 TotalStudents 更容易，但是谁能同意与后面产生的混乱相比节省的时间是有价值的呢？

注意 看起来很少有主题能够像匈牙利表示法(命名前缀)一样使开发人员两极分化。Microsoft 建议不要再使用匈牙利表示法使这个辩论更加白热化。我调查了很多开发人员并查阅了很多新闻组，看起来很多开发人员计划继续使用匈牙利前辍，这是因为匈牙利前辍是他们开发工作的一个重要组成部分。好的消息是您仍能够在自己的项目中继续使用匈牙利表示法并同时能够与.NET 框架的设计指导方针保持一致。我是使用命名前缀的

支持者，因此我不仅在本书中使用它们而且推荐使用它们。第 3 章“命名约定”中将详细介绍匈牙利表示法的使用，其中包括尽我所能制作的一些示例。如果您是憎恨前缀的另一个阵营的开发人员，我会说“各有所好”并且请您浏览本书示例中前缀的用法。

当更多的开发人员加入项目时，标准化所带来的益处就会成倍增加，因而公司通常首先想要进行标准化。在使用某个级别标准化的开发组织中，标准通常是由高层的管理人员推行，这些开发人员必须审阅维护其他人员编写的代码。然而使用标准通常只是措施之一，而且标准通常由内部创建，这些标准由很多独特的特征构成，而不是能够统一使用的技术。被雇佣的开发人员可能必须学习与他们所熟悉的技术有很大不同的开发技术。这样当然会增加不必要的工作并增加出现错误的几率。

每个编写代码的人都应该使用某些通用的标准化技术，即使只编写几行代码也应如此。标准化的需求随着代码开发人员的数量与程序中的代码行数呈指数增长。不论您是业余程序开发人员还是商用程序开发人员，标准化技术应是您基本技术的一部分。

很久以前我们在自己的公司作过一项分析，结果表明我们使用的标准很混乱，每个开发人员都在做自己想做的事情。由于我自己已经经历过，所以我知道要在开发过程中采用标准还有一段很长的路要走。这样做不容易，但却是值得这样做的。

本书内容

本书会和前一版一样优秀，本书是帮助 Visual Basic .NET 程序员创建专业化代码的书籍，并且本书仅仅针对 Visual Basic .NET 程序员。如果您是 C#程序员，则本书中的界面一章值得一读，但对全书而言，其内容并不适合您。同样的，如果您正在使用 Visual Basic 的早期版本，则应该读本书的第一版。然而，如果您是 Visual Basic .NET 程序员，不论是学生还是专业人员，要是能学习这本书，哪怕只遵循这本书中的几个标准，则本书也是物有所值的。

本书的内容并不是介绍应用程序架构、需求分析、设计规格或程序测试，本书是关于开发程序的书。书中某些主题确实会含有其他软件设计类书籍的内容，例如设计模块和过程，但是这些内容不是作为理论，而是作为实践规则出现在这里的。本书不会介绍如何编程，而是介绍如何将程序编的更好。

本书针对标准的实践性方面，因此“实践”是本书的关键词。本书使用了大量的代码示例，展示了好坏两方面的编码技术。在必要的地方会指出其他的方法并进行讨论。作者花费了大量的时间来设置书中代码的格式，以便使这些代码看起来像是在 Visual Basic .NET 代码编辑器中。因此作者使用了黑体字，这样代码就易读了，这样也为看起来乏味的代码清单添加了特色。

为了让文章更加清晰，作者尽量使每章的内容模式保持一致。但由于某些章节的内容不适合严格的模式，因此在必要的地方会出现略微的不同。然而，在每章的开始部分通常都会有本章思想与主题

的简略介绍，其中包括标准所涉及的问题或主题。开始部分的目的在于让读者关注问题本身，而不是介绍解决问题的标准与方法。我尽量避免在此部分加入过多的代码，而是在每章的后面提供详尽的代码示例。

在简略介绍之后是每章中与特定编程主题有关的重点目标列表。如果您不想遵循各章介绍的标准，那也应该确保达到了每章目标列表中的要求。实现编程目标的最佳途径就是遵循每章中的标准，但您注意到这些标准可能使开发团队瓦解。如果您感到我所倡导的标准并不合适，那么您可以使用其他技术来达到目标，只要达到目标就可以。

每章中都有大量的编程原则，即指导读者的特有的标准化技术。每章的编程原则部分说明如何以及为何要遵循特有的标准，并且还会提供正确与错误的应用示例。

在编程原则部分有时会提供一些实际应用，还会提供一些关于编程原则的详细说明以便说明编程原则中特有的组件(应用是指技术的运用而不是程序)。几乎每个实际应用除了包括遵照代码标准编写的进程外，还包括不遵循相关标准的代码示例。当有多个能够解决问题的方案时，有时会给出其他的示例。这些示例不仅仅是 Debug.WriteLine("Hello, World")。大多数代码示例来自于实际的商用项目，许多是完整的过程。希望这些代码能够使给定技术所带来的好处更加明显并且使这些技术易于在您的项目中实现。

某些编程原则与实际应用可能最适合您的情况，这样很好。某些情况下会指出非常重要的标准与技术，这是必须要遵守的部分。即使是在这种情况下，您仍可以作出不同的选择。关键是最终要接受一套标准并且坚持应用这些标准。这样做的好处随手可得。读完本书后您将写出更优秀的代码。

致　　谢

感谢 Devon Musgrave、Danielle Bird、Dail Magee Jr.和其他优秀的有才能的人们，其中包括 Microsoft 出版社的 Elizabeth Hansford 和 Lisa Pawlewicz。另外，还要感谢 Kel Good 对这本书提出的意见和建议，特别是在命名约定部分。最后，要感谢家人、朋友和同事，他们使这本书能够写成。

支持信息

为了降低书的成本，减轻读者负担，对于因内容很少而不值得单独配盘的图书，我们将其范例代码或练习文件放在我们的网站上，供读者下载。敬请访问以下网址：http://www.wenyuan.com.cn，查找本书的有关链接。

如果您对本书或配书文件有任何建议、意见或想法，请通过以下电子邮件与清华大学出版社计算机应用编辑二室客户服务部取得联系：

service@wenyuan.com.cn

或致函：

北京 1000084-157 信箱

读者服务部

邮编：100084

亦可致电：010-62792098-220。

请注意，上述地址并不提供软件产品的支持。

目　录

前言 xiii

第 I 部分　设计

第 1 章　创建对象和项目模板 3

1.1　使用对象模板 3
1.2　使用项目模板 5
1.3　编程原则 7
1.3.1　绝不要在对象模板中对应用程序和组件特有的值进行硬编码 7
1.3.2　在对象模板中特别是需修改之处提供广泛的注释和任务 9

第 2 章　设计模块和过程 11

2.1　创建具有很强内聚性的模块 12
2.2　创建松耦合和高度特定化过程 13
2.2.1　使所有过程都执行特定的功能 13
2.2.2　尽量提高过程的独立性 14
2.2.3　最小化扇入和扇出 15
2.2.4　尽量按字母表顺序排列模块中的过程 16
2.3　编程原则 17
2.3.1　给过程和模块起一个表意性强的名称 17
2.3.2　给每个过程设定惟一的出口 18
2.3.3　给每个过程定义一个明确的范围 20
2.3.4　在过程之间用参数传递数据 21
2.3.5　使用统一和直观明了的方式调用过程 23
2.3.6　使用 Return 语句返回函数值 25
2.3.7　在复杂函数中使用暂存变量 26

第 II 部分　约定

第 3 章　命名约定 31

3.1　匈牙利表示法 32
3.2　什么情况下不用匈牙利表示法 33
3.3　指明变量的数据类型 34

3.4 指明变量的范围 35
3.5 其他对象前缀 36
第 4 章 使用常量和枚举 40
4.1 使用常量 40
4.1.1 幻数容易出现数据输入错误 40
4.1.2 幻数难以更新 41
4.1.3 常量使代码更容易阅读 41
4.2 使用枚举 41
4.2.1 创建自定义枚举 42
4.2.2 使用自定义枚举 43
4.3 编程原则 44
4.3.1 给所有常量加上前缀 c_和范围指示符 44
4.3.2 用常量代替幻数而不必考虑可见范围 46
4.3.3 尽可能使用枚举 47
4.3.4 参数接收数量有限的值时使用枚举 48
4.3.5 验证作为枚举类型传递的值 48
第 5 章 变量 52
5.1 编程原则 52
5.1.1 定义用途明确的变量 52
5.1.2 给变量起一个表意性强的名称 56
5.1.3 在变量名中混合使用大小写字母 59
5.1.4 只对常用变量名或长变量名使用缩写 59
5.1.5 使用统一限定词 60
5.1.6 使用肯定形式的 Boolean 变量 61
5.1.7 显式声明变量 62
5.1.8 用精选的数据类型声明变量 64
5.1.9 只有在绝对必要时才使用 Object 数据类型 68
5.1.10 使用 Option Strict 选项严格限制类型 70
5.1.11 最小化变量可见范围 72
5.1.12 尽可能使用初始值设定项 74
5.1.13 使用与号(&)连接字符串 74
5.1.14 用字符串长度属性判断字符串是否为空 75

第 III 部分 编码结构

第 6 章 格式化代码 79

6.1 编程原则 82

6.1.1 不要在同一行中放入多个语句 82

6.1.2 使用续行符 83

6.1.3 缩进连续行 87

6.1.4 使用缩进显示代码的组织结构 90

6.1.5 在模块声明部分缩进代码以显示从属关系 97

6.1.6 用空行把相关的语句分组 98

6.1.7 创建可折叠的代码区域以便管理 104

第 7 章 注释代码 107

7.1 编程原则 108

7.1.1 用文字说明代码的作用 108

7.1.2 解释为什么要违背良好的编程风格 109

7.1.3 在写代码前先写注释 109

7.1.4 单色字符注释行仅用于主要注释 110

7.1.5 不要创建注释框 112

7.1.6 用撇号表示注释 113

7.1.7 增强注释的可读性 114

7.1.8 缩进注释使之与随后的语句对齐 116

7.1.9 给每个过程写一个注释头 116

7.1.10 用内嵌注释说明代码进程 120

7.1.11 用行尾变量来说明变量声明 125

第 8 章 循环结构 126

8.1 编程原则 126

8.1.1 循环次数确定时使用 For...Next 循环 126

8.1.2 循环次数不确定时使用 Do...Loop 循环 134

8.1.3 用 Do...Loop 循环代替 While...End While 140

8.1.4 使用 For Each...Next 循环遍历集合中的成员 140

第 9 章 控制代码流 144

9.1 编程原则 145

9.1.1 判断基于一个条件的值是否为真时使用 If...Then...Else 分支结构 145

9.1.2 当把一个非 Boolean 表达式与多个值进行比较时使用 Select Case 语句 148

9.1.3 使用行尾注释增加嵌套判断结构的清晰度 152
9.1.4 格式化表达式以便准确求值和易于理解 154
9.1.5 仅在别无选择时才使用 GoTo 语句 156
第 10 章 异常处理 159
10.1 异常对象 160
10.2 异常处理程序的类型 160
10.3 使用 Try...Catch...Finally 语句编写异常处理程序 161
10.3.1 捕获异常 163
10.3.2 异常处理程序和调用堆栈 165
10.4 中央异常处理程序 167
10.4.1 把异常写入文本文件 170
10.5 编程原则 173
10.5.1 用 Try...Catch...Finally 处理不可预料的和可预料的异常 173
10.5.2 处理不可预料的异常时使用统一的格式 174
10.5.3 永远不要指责用户 175
第 IV 部分 高级编程
第 11 章 编程对象 179
11.1 什么是对象 179
11.2 垃圾回收 180
11.3 编程原则 181
11.3.1 要尽早绑定对象 181
11.3.2 要尽可能使用.NET 对象而不是调用 API 函数 183
11.3.3 公开公共属性而不是公开公共变量 184
11.3.4 无论打开什么资源都要关闭它们 186
11.3.5 使用 OverLoads 创建名称相同但参数列表不同的属性和方法 186
11.3.6 为所有对象创建 Dispose 方法 188
11.3.7 要尽可能为类创建构造函数 190
11.3.8 仅在必要时才在类中添加终结器 192
11.3.9 仅在必要时才强制执行垃圾回收 194
11.3.10 使用 With...End With 提高性能和代码的可读性 194
第 12 章 文件操作 196
12.1 System.IO 196
12.1.1 System.IO.File 和 System.IO.Directory 196

12.1.2 System.IO.Path 200
12.2 System.Environment 201
12.3 编程原则 202
12.3.1 把所有临时文件都保存在用户的 Temp 文件夹中 202
12.3.2 用系统指定的临时文件名保存临时文件 203
12.3.3 及时关闭不再需要访问的文件 204
12.3.4 决不要在程序中使用硬编码路径 206
12.3.5 使用 System.IO.Path 来操纵文件路径 206
12.3.6 文件的默认保存路径为用户的 My Documents 文件夹 207
12.3.7 将文件添加到用户最近用过的文档列表中 207
12.3.8 删除重要文件时请求确认 208
第 13 章 调试 210
13.1 从 Visual Basic 6 到 Visual Basic.NET 210
13.1.1 用数据提示查看表达式的值 210
13.1.2 用 Debug.Assert 定义断言 211
13.2 条件编译 214
13.2.1 编写条件编译代码 214
13.2.2 用项目【属性页】对话框设置编译器常量 216
13.2.3 用命令行设置编译器常量 216
13.3 断点 217
13.3.1 设置和删除断点 217
13.3.2 进入中断模式 217
13.3.3 使用【断点】窗口管理断点 219
13.3.4 用 Stop 语句进入中断模式 222
13.4 Visual Basic.NET 调试窗口 222
13.4.1 【自动】窗口 223
13.4.2 【局部变量】窗口 224
13.4.3 Me 窗口 224
13.4.4 【监视】窗口 225
13.4.5 【快速监视】窗口 226
13.4.6 【命令】窗口 226
13.4.7 【输出】窗口 228
13.4.8 【任务列表】窗口 229
13.4.9 【模块】窗口 231