HZ BOOKS

Designed for Microsoft Windows NT Windows 98

Practical Standards for Microsoft Visual Basic

微软公司核心技术书库

(美) James D. Foxall 著 王建华 等译

Visual Basic 编程标准

机械工业出版社 China Machine Press

Microsoft Press

微软公司核心技术书库

Visual Basic 编程标准

（美）James D. Foxall 著
王建华 等译

机械工业出版社
China Machine Press

本书是讲述Visual Basic编程标准的一本专著。全书分5个部分，共14章。分别讲述使用Visual Basic编程时总体设计、代码结构设计、用户界面设计和小组软件开发应遵循的标准。全书配有大量应用实例，便于读者学以致用。

本书适用于Visual Basic 编程人员，对于使用其他语言的编程人员，也有很高的参考价值。

本书版权登记号：图字：01-2000-0803

图书在版编目（CIP）数据

Visual Basic 编程标准/（美）福克斯尔（Foxall, J. D.）著；王建华等译. -北京：机械工业出版社，2000.6

（微软公司核心技术书库）

书名原文：Practical Standards for Microsoft Visual Basic

ISBN 7-111-08043-2

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

中国版本图书馆CIP数据核字（2000）第23833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：陈贤舜

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2000年6月第1版第1次印刷

787mm × 1092mm 1/16 · 14.75印张

印数：0 001- 6 000册

定价：38.00元 (附光盘)

译 者 序

在日常生活和工作中，我们必须遵守一定的规则，如行车不能闯红灯，走路必须靠右行等，以保证人身和车辆的安全。使用家用电器的电源也有严格的标准，才能避免触电事故。对于软件开发人员来说，代码的编写必须遵循一套标准，才能编写出更好的代码，以便于使用、改进和维护。我们使用的许多软件，比如Microsoft Office，它为我们提供了美观而统一的界面，给使用和操作带来了很大的方便。这是运用统一的编码标准而产生的结果。在编写代码时，一致性和标准化已经成为一个十分重要的问题。

Visual Basic是目前使用非常广泛的一种编程语言，它功能非常强大，具有很大的灵活性和向后兼容的特性，大量应用于企业和商用编程环境之中。正因为它所具备的良好特性，导致产生一些编程工作中的粗心大意和不符合标准的做法，使得一些代码难以阅读和维护。

本书是介绍Visual Basic编程标准的一本专著，重点讲述编程标准实际应用的各种技巧。全书分5个部分，共14章。第一部分介绍Visual Basic编程时总体设计中应该遵循的一些标准，即创建对象和项目模板以及创建模块和过程时使用的标准。第二部分讲述Visual Basic编程时使用的一些约定。比如使用常量和枚举值时应该遵循的标准，使用变量时采用的标准，以及处理代码中的错误时应该遵循的标准。第三部分介绍代码结构设计方面的标准，如代码格式化的标准，如何给代码加上注释，循环结构使用的标准，以及如何控制代码流等。第四部分介绍用户界面的标准，如界面设计应该遵循的标准，用户的输入和通知消息使用的标准。第五部分讲述小组软件开发时应该遵循的标准，包括版本控制和源代码控制时采用的标准。本书提供了大量的编程标准实际应用的举例，直观明了，便于学习。

本书由王建华、杨宝明、侯丽坤、董志敏、陈唤、蒋小英、王卫峰和张新芳等翻译。由于译者水平有限，不妥之处在所难免，敬请读者指正。

译 者

2000年3月

前　言

作为社会中的成员，必须遵守日常生活的一些准则。我们往往并不考虑为什么要遵守这些准则，而是将它们视为理所当然的事情，并且自然而然地遵守这些准则。例如，当你的汽车开到一个十字路口的时候，你会把汽车停下来，看一看信号灯。确定谁拥有通行权（首先到达停车线并且靠右边的这辆汽车），然后在适当的时候通过十字路口。当你刚刚开始学习驾驶汽车时，你要花费很大的精力才能掌握交通规则，但是，一旦熟悉了情况，遵守交通规则就会成为你的自觉行为。编程规则的情况也是一样。

110V的电气标准是影响我们日常生活的技术标准之一。当你将一台新的家用电器插入墙上的电源插座时，你不必查看这台设备的说明书，也不必使用电压表来测定电源插座的电压，以便确定家用电器能否插入电源插座。你完全可以放心，它们都是严格符合电气标准的。如果某个设备不需要110V的电压，它可以使用变压器之类的装置来降低电压，使它在与110V的系统连接时仍然能够正常工作。当需要更高的电压时，比如衣服烘干机需要220V的电压，那么这个设备的插头的大小和形状是完全不同的，这使它无法与110V的系统相兼容。显然，如果不遵守这些标准，就可能造成人身伤害或者死亡。这些标准是如此重要，以至于专门设立了一个机构负责设备的认证，并确保它们符合必要的标准并且通过某些安全检查，这就是为什么几乎所有电气设备都带有UL标志。

运用接口和编码标准是无法使人避免触电的，也不能避免两辆汽车在十字路口相撞，但是严格遵守公开发布的一套标准确实是有好处的。按照一组标准编写的代码更加容易维护和改进，正确使用人们已经认可的标准往往可以创建更好的代码。研究表明，按照标准编写的代码包含的错误通常比较少。

如果想要了解将标准应用在用户界面时带来的好处，你只需要看一看Microsoft的Office软件，它就是个很好的例子。不管是否喜欢这个界面，你都能够非常容易理解该软件的可视化标准给最终用户带来的好处。没有一种东西是完美无缺的，我能够提出Office软件的界面应该进行的一系列改进，但是，当我编写第一个与Microsoft Office相兼容的商用程序时，我就真正意识到这个界面的优点所在了，它是运用标准而产生的一个直接结果。你需要花费大量的心血才能得到了人们的承认，当我使用这个工具来创建某些必要的特性，而它却无能为力的时候，甚至想要放弃它。不，这不是Microsoft Visual Basic，不过它是个Microsoft的产品。我和我的编程员同事发现，如果不想得到他人的认可，我们也能够编写出非常好的代码。我们还在工具栏、菜单和快捷键上花费了大量的时间。但是，当我们的客户开始使用该产品时，我们得到大量的赞美和褒扬。

客户对我们说，实现我们的应用程序时所需要的内部培训和支持比预期的要少。用户可以将精力集中放在我们程序的功能和特性上，而不是放在导航之类的界面工具上。例如，当用户想要打印时，他知道何处可以找到Print按钮。该按钮的位置与Office中的位置是一样的，它显示的打印机图标是用户已经熟悉的。同样，熟悉Office的标准热键的用户只需要按下Ctrl+P键，就可以打印。总之，我们在软件开发中的初始投入就获得了客户的回报。就应用程序的界面来

说，标准产生的一致性是非常关键的。虽然它看起来并不明显，但是，当编写代码时，一致性和标准化也是个关键。

为什么必须阅读本书

在所有编程语言中，Visual Basic拥有的用户数量最大，而且这个数量还在不断增长。据估计，Visual Basic的用户数量大约为300万。在它被视为一种业余爱好者的编程语言后，它很快就赢得了专业软件开发人员的重视，而且现在已经广泛用于企业和其他商用编程环境中。Visual Basic的巨大灵活性与它的向后兼容性相结合，使它成为一种功能非常强大的编程语言。不过，这些特性也会导致编程中的粗心大意。

例如，Visual Basic使你能够创建我所谓的Voodoo（魔怪）变量。Voodoo变量是指在编程过程中仅仅按它被引用的名字来创建的一种变量。如果你的程序模块中没有包含“显式选项”语句，你就可以通过引用尚未说明或者尚未使用的变量的名字来创建一个Voodoo变量。Voodoo变量常常是误创建的变量，它们很难调试。Visual Basic甚至不应该允许你取消显式说明变量的要求。这个特性可以用于向后兼容，但是它在最新的编程操作中是无法使用的，在为娱乐而编写的应用程序中也不能使用，当然在分布式应用程序中也不能使用。更为困难的是，当你安装Visual Basic时，它的默认设置并不要求对变量进行显式说明。如果你不知道激活这个特性，那么将会在你的代码中埋下一颗地雷。

许多编程人员多年来经常遇到编写得粗心大意的代码，有的是他们自己编写的代码，有的是他们小组中其他人编写的代码。低劣的编程技术产生的后果是数不胜数的。例如，你可能想要对一个过程进行一些小的修改，结果却不得不重新编写整个过程，原因是整个代码修改起来并不容易。当你将一个例程中的各个进程组合起来时，会使例程的改进和修改变得困难起来，虽然这种修改并不是不可能。而编写正确的代码，在需要时进行修改通常要容易得多。

也许你曾经修改过别人编写的代码，结果发现，你需要花费几个小时、几天甚至几个星期，才能了解某些进程如何运行。也许你在经过一段时间后又重新查看自己编写的代码，结果却发现你的代码注释并不充分，你甚至记不得如何来实现这些进程。当你不得不处理你自己编程中的错误时，标准的好处就变得非常明显了。严格按照标准来编写的代码通常包含充分的注释，因此很容易了解过程中发生的一些情况，也能够很容易了解任务是如何完成的。代码的直观显示，比如代码的缩进、白空间的使用、变量的正确命名，等等，都能够使代码非常容易理解。

许多编程人员仍然不能充分考虑它们的变量的数据类型。比如在不必要的时候使用数据的变形，或者在需要使用Double（双精度）数据时却使用了Single（单精度）数据。在这些情况下，Visual Basic能够执行类型强制转换，也能进行意料之外的圆整，使你可以得到意想不到的结果。例如，如果你将一个带有小数值的数字放入一个Integer（整数）变量中，该值将被圆整，但是不会产生错误。然而，如果你试图将一个值放入一个变量，而这个值比它的数据类型允许的大小要大，那么就会产生一个运行期错误。在程序设计的时候，如果不让该程序运行大量的测试数据，几乎无法对执行情况进行测试。你必须从一开始就对变量进行正确的定义，否则，要等到你的客户在他的计算机上运行这些变量时，才会发现问题。

代码的性能始终都是软件开发人员首先考虑的问题。因此，虽然计算机的能力一直以极高的速度向上攀升，但是你应该记住，并不是人人都将计算机当作最先进的软件开发设备来使用的。如果你编写的代码符合某些标准，那么它们运行的速度常常比不符合标准的代码要快，这

些标准包括严格地进行数据的键入以及对规定的任务使用正确的循环等。

不遵守正确的编码标准的另一个后果是会给你这个软件开发人员带来影响，那就是你可能找不到工作，甚至可能丢掉你的饭碗，因为你编写的代码不严格。作为Visual Basic编程人员的雇主，我发现我没有聘用的大多数求职者是因为他们编写的代码不符合要求，而不是因为他们的履历有什么问题。我审阅了每个求职者编写的示例代码，最常见的问题是他们的代码没有准确的文字注释，就不高明的代码来说，这只是冰山的一角。我看到的情况真的令人吃惊，并且常常是令人遗憾的。我审阅的代码中没有一个显式说明的变量，到处都是Voodoo变量。我甚至看到过这样的代码，里面的所有变量都是在模块级上说明的变量。出于好奇，我对某些随机选择的变量进行了搜索。大多数变量只用于一个过程中。许多变量哪儿也没有使用！当我询问代码的作者为什么这样做时，他告诉我说，他感到将来可能需要使用这些变量，因此他只是为将来做准备的！如果你认为这些东西都不是什么不好的编程做法，那么本书对你来说应该是非常有用的。

如果你认为我是吹毛求疵，你可能是对的。但是我可以告诉你，象我这样的人并不是只有我一个。许多人都认识到，凡是遵守严格的编程标准，编写出非常好的代码的人，他就能够学会对任何进程进行编码。如果你想找一份工作，那么请将本书介绍的原则牢记在心，并且修改你的代码，然后再将你的代码送给雇主审阅。

正如我已经讲过的那样，Visual Basic是一种功能强大和非常灵活的编程语言，它具有内置的向后兼容的特性，这给它带来了许多的问题。由于大量的Visual Basic编程员是自学的，缺乏正规的编程技术培训，而这种培训有助于他们了解不好的编程技术带来的危害，因此上述问题就显得更加严重。这些情况使得标准化的工作更为必要。

虽然使用标准化技术编写的应用程序包含的错误比较少，比较容易维护，并且更加容易修改，这些优点不能说是够了，但是有些编程人员竟然反对标准化。持有这种观点的人往往强词夺理，知识浅薄，反对革新，有的纯粹是出于懒惰。例如，有的人说给变量加上3个字符的前缀来表示它的数据类型太费功夫，这好比是说，调用变量TotalStudents时键入整个变量太费事，不如键入X简单。当然，键入X比键入TotalStudents是要容易些，但是，与键入X给以后带来的麻烦相比，究竟哪个方法更加合算呢?

随着更多的编程人员参与一个程序项目的开发，标准化带来的好处就显得更加明显，因此常常是企业和商用产品开发机构首先试图实现标准化。在使用某种标准化技术的软件开发机构中，采用的标准通常来自一个高层经理，它必须审阅和维护其他人编写的代码。但是这些标准常常是应该使用的那些标准的很小一部分，并且它们常常是内部建立的一些标准，它们包含许多专用的特性，而不是通用的编程技巧。当编程人员被这样一个单位聘用时，他将不得不学习不同于他已经熟悉的那些编程技巧。当然这会增加一些不必要的工作，并且增加了出错的可能性。

编写代码的每个人都应该使用通用的标准化技巧，即使编写几行代码，也不能例外。随着参与代码编写的人员的增加，以及程序中的代码行数的增加，标准化的必要性也大大增加了。无论你是业余爱好者，还是企业或商用软件的开发人员，标准化技术应该是你使用的基本技巧的一部分。

我们在很早以前进行过一次分析，结果发现在我们公司中使用了一种混合标准，每个编程员基本上都在做他想要做的工作。在编程中要采用统一的标准必须经过很长时间的努力，这并不容易，但这是值得的。

本书内容简介

如果你曾经查找过一组特定的Visual Basic编程标准，可能会因为它缺乏可供使用的材料而感到不快。Steve McConell撰写的《Code Complete》一书（Microsoft出版社1993年出版）是介绍编程标准的一本好书，但是它主要针对C语言编程人员而写，你很少能够找到Visual Basic代码的例子。这本书介绍了许多其他语言的编程技巧，比如Pascal，而Visual Basic则成了二等公民。另外，《Code Complete》主要是以统计数字为基础的高级理论，而如果你是个Visual Basic编程员，通过这本书你可以学到你真正要学的东西。

就本书而言，我试图将它写成一本供Visual Basic编程人员使用的专业性编程著作，并且只供Visual Basic编程人员使用。如果你是一个C++编程员，你会发现关于界面的几章值得一读，但是阅读全书则并无必要。不过，如果你是个Visual Basic编程员，无论你是个学生，还是个专业编程员，那么只要你学习并且遵循书中介绍的几项标准，你都会得到应有的回报。

本书并不介绍应用程序的结构、需求分析、设计规范或测试等内容，它介绍的是软件的开发。有些问题，比如设计模块和过程，当然包含你可以在软件设计著作中能够找到的材料，但是这里是作为实际应用指南来介绍的，而不是纯粹的理论。这里你学不到如何进行编程的方法，但是你可以学到如何编写更好的程序。

本书的重点放在标准化的实际应用的各个方面，“实际应用”是关键字。书中大量使用了代码举例，既展示好的编程技巧，也展示不好的编程技巧。必要时，还展示并介绍了一些替代方法。我也花费了大量的时间对本书的代码进行了格式化，使它看起来就像你在Visual Basic的代码编辑器中使用的代码一样。我还根据情况使用了黑体字，希望这使代码更易于阅读。

为了清楚起见，我设法采用统一的格式来展示每一章中的材料。有些章节的内容本身不适合采用严格的格式，因此，必要时我进行了一些小的修改。但是，总的来说，每一章的开头是本章内容和问题的概述，包括编程标准要解决的问题和课题。开头一节的目的不一定告诉你用于解决问题的标准和方法，而是向你讲述问题的本身。我试图在这一节中不放入太多的代码，而是将详细的代码举例放在每一章的后面部分中。

总的介绍之后是一个带项目符号的列表，列出与本章介绍的特定编程问题相关的目的。如果你不想实现此章中介绍的标准，你仍然要确保你的操作符合本章列出的目的。解决某个编程问题的最好方法是按照该章介绍的标准来操作，但是，我相信你一定知道，许多编程问题很快就会将一个编程小组分成许多个部分。如果你坚持认为我主张使用的标准不适合，那么请改用能够达到目的的其他方法，只要能够达到目的就行。

在每一章中，你会发现许多编程原则，它们是一些特定的标准化编程技巧，用于指导你进行编程操作。每个编程原则都做了相应的编号。这种编号有两个作用，一是使你能够更容易交叉参考本书中的各个编程原则，另一个作用是使管理人员和教员能够编译特定的编程原则列表，以便用于他们的编程项目或说明。编程原则的每一节都包含一些文字，说明如何和为什么要实现一个特定的标准，并且通常还包含一些成功地使用和没有成功地使用编程原则的代码示例。

有时，在编程原则这一节中，我介绍了一个或多个实际应用举例，它们是与要遵循的编程原则相关的更加详细的原则，并且说明了改原则的某些特定组件（这里的“应用”是指技巧的应用，而不是程序的应用）。几乎每个实际应用举例都包含一个不符合相关标准的代码举例，也包含一个按照标准化规则编写的相同代码进程。有时，如果有多个可以接受的解决方案，那

么将提供一些替代例子。这些例子完全不是试验性的代码例子，大部分是来自实际商用程序项目的代码例子，并且许多例子是完整的过程。我希望这些代码例子会更加清楚地展示某种编程技巧带来的好处，并使该编程技巧能够更加容易地在你自己的程序项目中实现。

有些编程原则和实际应用举例也许与你的情况关系更加密切，这很好。在某些情况下，我设法指明一些非常重要的编程标准和技巧，也就是不应该出现偏差的那些情况。但是，即使在这些情况下，你的决定可能不同于我的决定。根本的关键是要接受一组正式的标准，并且始终一贯地使用这些标准。这样做的好处是显而易见的。读完本书后，你就能够编写更好的代码。

目　录

译者序

前言

第一部分　设　计

第1章　创建对象和工程模板……1

1.1　使用对象模板……1

1.2　使用项目模板……2

1.2.1　Visual Basic项目模板概述……2

1.2.2　创建自定义项目模板……4

1.3　自定义模板的行为特性……4

1.3.1　激活模板和取消模板的激活状态……4

1.3.2　设置模板文件夹……5

1.4　编程原则……5

1.4.1　不要将对象模板中的特定应用程序的值或特定组件的值进行硬编码……5

1.4.2　在对象模板中提供内容广泛的注释，尤其是在需要进行修改的地方要加上注释……7

第2章　设计模块和过程……9

2.1　创建具有很强内聚力的模块……9

2.2　创建松散连接和高度专用的过程……10

2.2.1　使所有过程都执行专门的任务……10

2.2.2　尽量使过程成为自成一体的独立过程……11

2.2.3　尽量减少扇入和扇出……12

2.2.4　设法按字母顺序对模块中的过程进行排序……12

2.3　编程原则……13

2.3.1　为过程和模块赋予表义性强的名字……13

2.3.2　为每个过程赋予单个退出点……15

2.3.3　为每个过程赋予明确定义的作用域……18

2.3.4　用参数在过程之间传递数据……18

2.3.5　使用统一和直观明了的方式来调用过程……21

第二部分　编程中使用的约定

第3章　命名约定……23

3.1　数据类型后缀……23

3.2　匈牙利标记法……23

3.2.1　表示变量数据类型的前缀……24

3.2.2　表示变量的作用域的前缀……25

3.2.3　其他前缀……26

第4章　使用常量和枚举值……28

4.1　使用常量……28

4.1.1　幻数很容易在数据输入时出错……28

4.1.2　幻数很难更新……28

4.1.3　常量使代码更容易阅读……29

4.2　使用枚举值……29

4.2.1　创建自定义的枚举值……30

4.2.2　使用自定义枚举值……30

4.3　编程原则……31

4.3.1　给所有常量加上前缀c_和作用域指示符……31

4.3.2　无论什么作用域，均用常量取代幻数……33

4.3.3　只要可能，均应使用枚举……33

4.3.4　引用控件数组的元素时请使用常量……34

4.3.5　将应用程序前缀或公司特定的前缀用于枚举成员……35

4.3.6　当枚举值不能使用时，使用常量……36

4.3.7　当参数接受有限数量的值时，请使用枚举……38

4.3.8　验证作为枚举类型传递的值……38

第5章　变量……41

5.1　编程原则……41

5.1.1　定义有焦点的变量……41

5.1.2　为变量赋予表义性强的名字……43

5.1.3　在变量名中混合使用大小写字母……46

5.1.4 只对常用变量名和长变量名进行缩写 ……46
5.1.5 使用统一的量词 ……47
5.1.6 使用肯定形式的布尔变量 ……47
5.1.7 显式说明变量 ……49
5.1.8 用精心选择的数据类型说明变量 ……51
5.1.9 只有在绝对必要时才使用Variant数据类型 ……54
5.1.10 尽量缩小变量的作用域 ……56
5.1.11 使用“和”字符对字符串进行并置操作 ……58
第6章 对错误的处理 ……60
6.1 Visual Basic的编译选项 ……60
6.2 Err对象 ……61
6.3 错误处理程序的类型 ……62
6.3.1 使用On Resume Next以忽略错误 ……62
6.3.2 使用On Error GoTo转移执行的代码流 ……64
6.3.3 错误处理程序与调用栈 ……68
6.3.4 使用On Error GoTo 0，在运行时取消错误处理程序的激活状态 ……69
6.3.5 用调试方式激活错误处理程序和取消其激活状态 ……70
6.4 中央错误处理程序 ……71
6.5 编程原则 ……76
6.5.1 使用On Error GoTo语句捕获意料之外的错误 ……76
6.5.2 使用On Error Resume Next语句捕获预料之中的错误 ……77
6.5.3 创建统一的错误处理程序块 ……79

第三部分 代码结构

第7章 代码的格式化 ……83
7.1 编程原则 ……86
7.1.1 不要将多个语句放在同一行上 ……86
7.1.2 使用行接续符 ……87
7.1.3 缩进后续行 ……90
7.1.4 运用语句缩进来显示代码的组织结构 ……92
7.1.5 对模块的Declarations部分中的代码进行缩进，显示其从属关系 ……98
7.1.6 使用白空间将相关语句组合在一起 ……99
第8章 代码的注释 ……106
8.1 编程原则 ……106
8.1.1 用文字说明代码的作用 ……106
8.1.2 如果你想违背好的编程原则，请说明为什么 ……107
8.1.3 用注释来说明何时可能出错和为什么出错 ……108
8.1.4 在编写代码前进行注释 ……109
8.1.5 纯色字符注释行只用于主要注释 ……110
8.1.6 避免形成注释框 ……112
8.1.7 使用撇号来指明注释 ……112
8.1.8 增强注释的可读性 ……114
8.1.9 对注释进行缩进，使之与后随的语句对齐 ……115
8.1.10 为每个过程赋予一个注释标头 ……115
8.1.11 使用内部注释来说明代码进程 ……118
8.1.12 用行尾注释来说明变量 ……122
第9章 循环结构 ……123
9.1 编码指导原则 ……123
9.1.1 使用For...Next，使代码循环运行规定的次数 ……123
9.1.2 使用Do...loop，使循环按照未定次数来运行 ……130
9.1.3 用Do...Loop取代While...Wend ……135
9.1.4 使用For Each...Next，循环运行一个集合的所有成员 ……135
第10章 控制代码流 ……140
10.1 编程原则 ……140
10.1.1 当根据一个条件是True还是False来作出判断时，使用If...Then...Else ……140
10.1.2 对非布尔表达式与各种可能的值进行比较时，使用Select Case语句 ……143
10.1.3 用行尾注释使嵌套式判断结构更加清楚 ……147

10.1.4 对表达式进行格式化，以便进行准确的计算和代码的理解 ……148
10.1.5 尽可能不要使用GoSub ……149
10.1.6 只有当没有其他替代方法或者当转移到一个错误处理程序或单个退出点时，才使用GoTo语句 ……150

第四部分 用户界面的操作

第11章 用户界面的设计 ……153
11.1 界面设计必须保持一致 ……153
11.2 编程原则 ……154
11.2.1 为窗体赋予统一的外观和行为特性 ……154
11.2.2 使控件具备标准外观 ……162
11.2.3 在规定情况下使用最佳界面组件 ……167
11.2.4 提供便于理解和使用的菜单 ……171
11.2.5 尽可能使用系统颜色 ……177
第12章 用户的输入和通知消息 ……179
12.1 用户的输入 ……179
12.2 通知消息 ……180
12.3 编程原则 ……180
12.3.1 确保完善的键盘导航和交互操作特性 ……180
12.3.2 提供统一和直观的鼠标交互操作特性 ……185
12.3.3 创建有创意和功能良好的消息框 …193

第五部分 小组操作的项目

第13章 版本控制 ……199
13.1 编程原则 ……199
13.1.1 每次对程序编译后应将版本号递增1 ……199
13.1.2 在About对话框中显示程序的版本号 ……200
13.1.3 保持ActiveX组件中的向后兼容性 ……201
13.1.4 在Readme文件中记下所做的修改 ……205
13.1.5 对文件做好备份 ……206
13.1.6 使用Microsoft Visual SourceSafe来维护源代码的版本 ……206
第14章 源代码控制 ……207
14.1 小组软件开发必须解决的问题 ……207
14.2 了解Visual SourceSafe ……208
14.3 安装Visual SourceSafe ……208
14.3.1 创建Visual SourceSafe数据库 ……209
14.3.2 打开Visual SourceSafe数据库 ……210
14.3.3 将用户添加给Visual SourceSafe数据库 ……211
14.3.4 将Visual Basic工程置于SourceSafe控件下 ……211
14.4 Visual Basic项目与Visual SourceSafe …213
14.4.1 指定工作文件夹 ……214
14.4.2 创建项目的工作拷贝 ……215
14.4.3 使用Visual SourceSafe Explorer借出文件 ……216
14.4.4 通过Visual Basic IDE借出和归还文件 ……218
14.4.5 将新文件添加给源代码控制下的项目 ……219
14.4.6 获取文件的最新版本 ……219
14.4.7 对不同版本进行比较 ……221

第一部分 设 计

第1章 创建对象和工程模板

如果要创建许多个项目，那么通过创建和使用对象和项目模板，就可以节省大量的开发时间，并增强应用程序的一致性。例如，咨询公司常常为不同客户创建不同的应用系统。当咨询公司通过代码和对象的重复使用来减少其工作量时，它们就能获得较高的收益，缩短开发周期。

如果要开发许多应用程序，可以创建非常相似的许多组件，虽然它们并不完全相同。可以使用常用的闪现屏幕或About框和版权窗体。也可以拥有一个常用的配电盘式的窗体类或自定义的浏览文件类。在这些情况下，可以对有关对象进行小的修改，而将它们作为模板添加到集成式开发环境（IDE）中。然后用它们启动新的开发项目，确保所有应用程序具有比较统一的外观。

1.1 使用对象模板

当选择将新对象添加给一个项目（project）时，Visual Basic显示一个对话框（例如Add Form（添加窗体）对话框）。在这个对话框中，可以选定一个现有对象，也可以创建新对象（既可是通用对象，也可以是基于模板的对象，见图1-1）。虽然Visual Basic配有许多预定义的对象模板（最主要的是许多窗体模板），但并不限于只能使用这些现成的模板。用Visual Basic创建和保存的任何标准对象（例如窗体或程序模块）都可以做成对象模板。

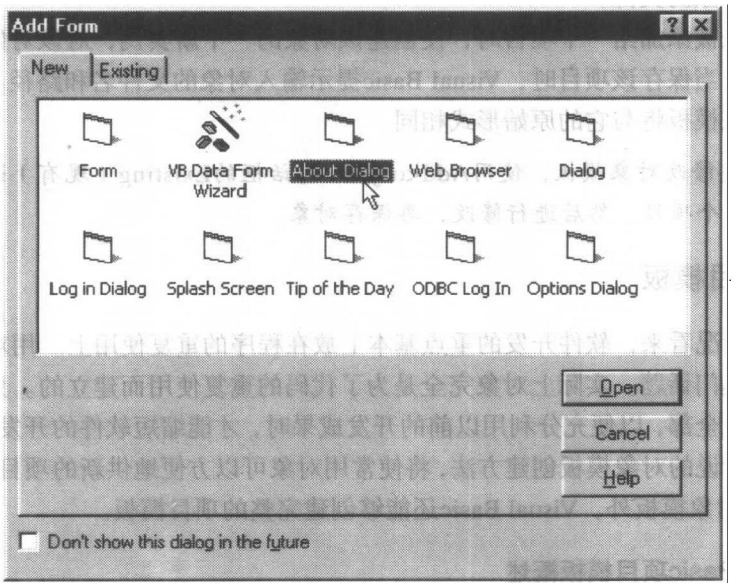

图1-1 Add Form对话框显示了Template\Forms文件夹中包含的窗体模板

当显示包含许多对象模板图标的Add<object>对话框（见图1-1）时，这些对象模板并不是来自一个静态列表，Visual Basic查看主文件夹中的Template文件夹，而Template文件夹中包含了许多子文件夹，可以拥有模板的每种对象都有一个子文件夹（见图1-2）。Visual Basic查看相应的子文件夹，并将该子文件夹中的每个对象作为模板图标添加给Add<object>对话框。如果子文件夹中的文件不属于正确的类型，则忽略该文件。

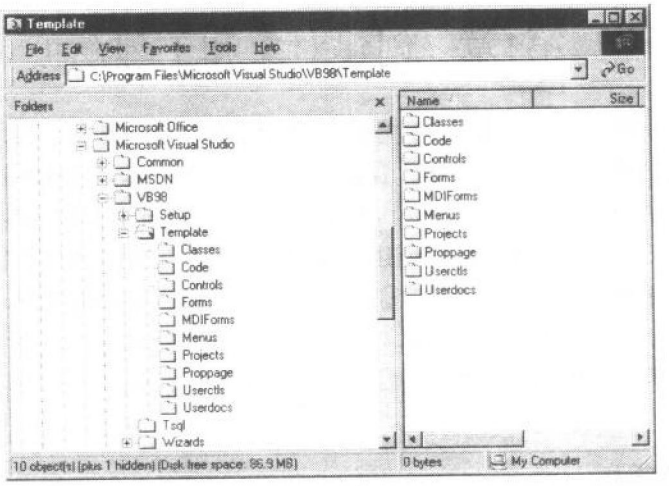

图1-2　Template文件夹包含可以拥有模板的不同对象的子文件夹

例如，假设创建了一个窗体对象，并希望在每次从Project菜单中选择Add Form（添加窗体）时使该窗体对象出现在Add Form对话框中，只需将窗体文件的拷贝放入Template文件夹的Forms子文件夹中。如果窗体拥有一个与其相关联的二进制文件（即.frx文件），该文件也必须放入Forms子文件夹中。若不希望对象出现在模板中，就从Template文件夹的相应子文件夹中删除或移走该对象的文件。

当模板对象被添加给一个项目时，便创建该对象的一个新实例，对该对象所做的修改并不传给模板本身。当保存该项目时，Visual Basic提示输入对象的文件名和路径。下次根据该模板创建对象时，该模板将与它的原始形式相同。

注意　若要修改对象模板，使用Add<object>对话框的Existing（现有）选项卡，将对象添加给一个项目。然后进行修改，再保存对象。

1.2　使用项目模板

从目前的情况看来，软件开发的重点基本上放在程序的重复使用上。用对象进行编程已经成为程序员的热门话题。实际上对象完全是为了代码的重复使用而建立的。只有当使用现有对象的某些部分或全部，以便充分利用以前的开发成果时，才能缩短软件的开发时间和测试时间。按照本章开头所说的对象模板创建方法，将使常用对象可以方便地供新的项目或现有项目使用。除了能够创建对象模板外，Visual Basic还能够创建完整的项目模板。

1.2.1　Visual Basic项目模板概述

项目模板是包含一些基本对象（如窗体或程序模块）的框架模板，也可以是包含多文档界

面（MDI）前端程序这类复杂代码的半完整应用程序。若要查看项目模板的例子，仅仅启动Visual Basic本身是不够的。如果没有取消图1-3所示的New Project（新项目）对话框的激活状态（参见本章后面的“激活模板和取消模板的激活状态”这一节的说明），那么每次启动Visual Basic时，就会显示New Project对话框。该对话框中显示的默认项目模板是随Visual Basic一道安装的。这些项目模板位于Visual Basic主文件夹下的Template\Projects文件夹中（见图1-4）。

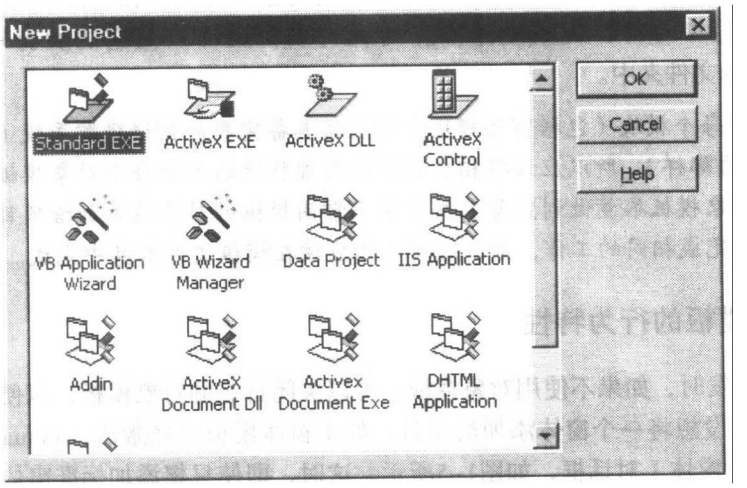

图1-3 Visual Basic用于新程序开发的项目只是一些默认模板

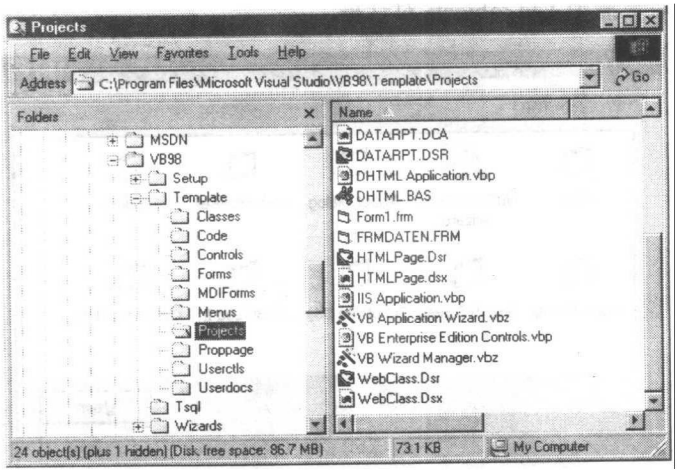

图1-4 Visual Basic项目模板位于Template\Project文件夹中

双击New Project对话框中的一个项目图标时，就为Template\Project文件夹中的对应项目文件创建一个临时拷贝。保存该项目时，要求给项目中的所有对象以及项目本身命名。这与将基于模板的对象添加给项目时的情况很相似，Visual Basic决不会改写原始模板文件。

可以修改Visual Basic的默认模板，但不能修改基本项目，如标准EXE、ActiveX EXE和ActiveX DLL。若要修改任何现有的项目模板，必须直接打开这些项目。直接打开项目模板而不是根据模板创建新项目时，Visual Basic就会像对待任何其他项目那样对待该模板。当保存所做的修改时，实际模板文件会被相应地改写。这是调整标准项目模板的好办法，以便这些模板能更加直接地用于开发环境。

1.2.2　创建自定义项目模板

修改现有项目模板是实现项目的一致性和迅速启动新项目的方法之一，但它不是唯一的办法。更好的办法是创建自己的项目模板。通过创建自定义的项目模板，就能消除需要重新安装Visual Basic时会改写模板的危险。若要创建新的项目模板，只需要创建任何类型的一个新项目，然后将它的所有属性设置为相应的值。接着，创建对象（如窗体和模板），就像让它们出现在基于模板的项目中那样。完成上述操作时，将项目文件以及它的所有对象文件保存在Template\Projects文件夹中。

注意　由于每个模块（包括窗体模块）都应该具备完整的错误跟踪手段（正如我在第6章中介绍的那样），所以应该将相应的错误处理特性添加给每个对象模板。它的目的是使每个对象模板尽量做到完整。每当基于常用模板的对象被添加给项目时，如果这些对象必须完成相同的工作，那么这项工作应该在模板文件本身中完成。

1.3　自定义模板的行为特性

进行软件开发时，如果不使用对象模板，可以关闭特定的对象模板，以便稍稍加快开发的进程。例如，假设想将一个窗体添加给项目，如果窗体模板已经激活，Visual Basic就会显示Add Form（添加窗体）对话框，如图1-5所示。这时，即使只想添加标准窗体，也必须选定一个特定类型的窗体。但是，可以取消窗体模板的激活状态，以便取消这个操作步骤。同样，也可以为其他类型的对象撤消Add<object>对话框。

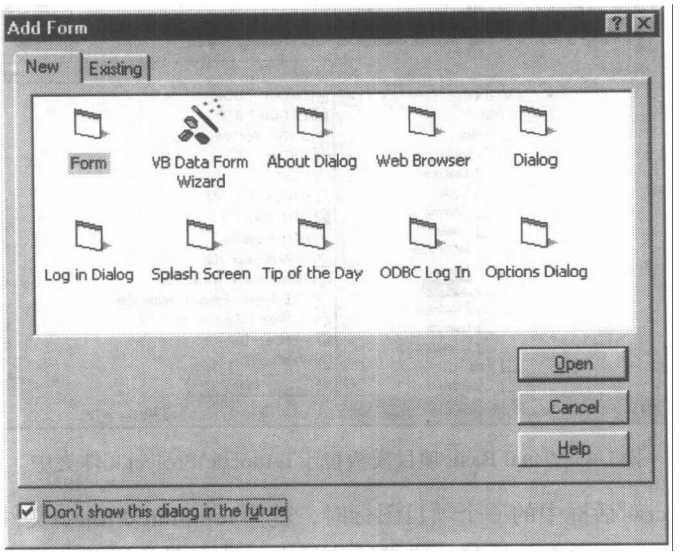

图1-5　如果不使用模板，Add<object>对话框会给开发进程增加一个不必要的操作步骤

1.3.1　激活模板和取消模板的激活状态

若要取消一个对象模板的激活状态，从Tools菜单中选择Options，然后单击Environment（环境）选项卡，打开Visual Basic的Options对话框（见图1-6）。如果不想使用某种类型的模板，就清除相应的对话框。当取消了某个对象类型的模板时，添加该类型的对象就会导致该类型的

新标准对象被添加给项目，但不出现Add<object>对话框的中间步骤。

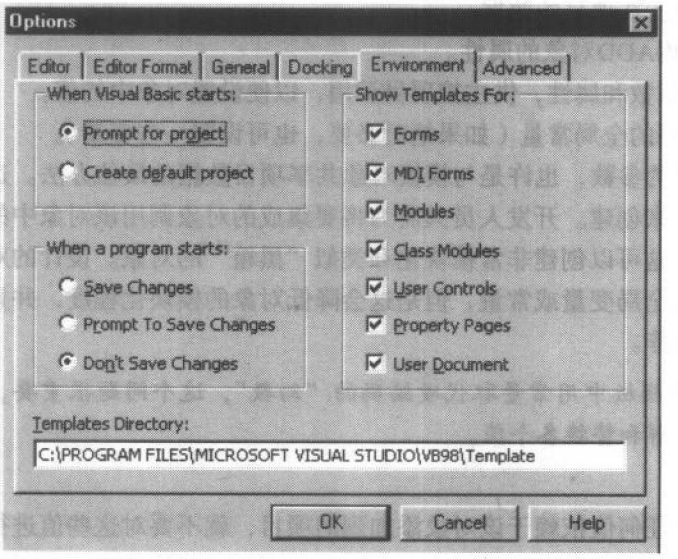

图1-6 可以选择要激活何种类型的模板或取消何种模板的激活状态

1.3.2 设置模板文件夹

通过设定自定义的模板文件夹，就能使开发环境增加另一层一致性。在Options对话框中Environment选项卡上的Templates Directory（模板目录）中设定的文件夹（见图1-6）是父文件夹，Visual Basic可从中查找对象模板。例如，激活模板并且从Project菜单中选择Add Form（添加窗体）时，Visual Basic就在规定文件夹中查找名叫Forms的子文件夹，然后将找到的所有窗体文件添加给对话框。

若要创建自定义模板文件夹，必须创建根文件夹（即在Templates Directory文本框中设定的文件夹）和相应的子文件夹。若要简化操作进程，只需将Visual Basic文件夹中的Template文件夹拷贝到一个共享驱动器中。删除不想使用的模板，并将Templates Directory文本框中的路径设置为已创建的新文件夹。然后将公司的所有对象模板放入这个自定义模板文件夹中，并让所有开发人员将新文件夹的路径输入他们的Template Directory文本框。每当一位开发人员将模板对象添加给项目时，他将使用与其他任何人相同的模板。

使用对象和项目模板的目的

使用对象和项目模板的主要目的是：

- 促进和鼓励代码的复用
- 缩短新项目和现有项目的开发时间

1.4 编程原则

1.4.1 不要将对象模板中的特定应用程序的值或特定组件的值进行硬编码

对象模板应该尽可能做到通用。由于很少能够预先知道对象终止运行时位于哪个项目之中，