微软.NET程序员系列

Microsoft Press

- 欧美读者评价: ★★★★
- C++权威专家精心编著
- Visual Studio .NET产品组鼎力推荐
- 理论与范例相结合
- 全面介绍C++托管扩展

Microsoft® Visual C++ .NET 托管扩展编程

Programming with Managed Extensions for Microsoft Visual C++ .NET

[英] *Richard Grimes* 著

梁超 钱伟 吴连祥 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 C++ .NET 托管扩展编程

[英] Richard Grimes 著

梁 超 钱 伟 吴连祥 译

Visual Studio .NET 产品组 审校

清 华 大 学 出 版 社

北 京

内 容 简 介

对 Microsoft Visual C++语言进行扩展可以生成.NET 代码，这些扩展称为 C++ .NET 托管扩展。C++是惟一一种可以在同一个源文件中混合.NET 代码和非托管代码的语言，是真正的.NET 系统语言。

本书内容根据开发过程进行组织。全书分 7 章，首先描述语言的基本特性，然后探讨.NET 的特性，如 Interop、委托和 GUI 应用程序，最后介绍 Visual Studio .NET 的项目管理和调试功能。

本书适合准备使用 C++ .NET 托管扩展进行开发的中、高级读者阅读。

北京市版权局著作权合同登记号：图字 01-2003-2189 号

图书在版编目(CIP)数据

Visual C++.NET 托管扩展编程/(英)格瑞默著；梁超，钱伟，吴连祥译.—北京：清华大学出版社，2003
(微软.NET 程序员系列)
书名原文：Programming with Managed Extensions for Microsoft Visual C++
ISBN 7-302-06646-9

I.V… II.①格…②梁…③钱…④吴… III.C 语言—程序设计 IV.TP312

中国版本图书馆 CIP 数据核字（2003）第 037146 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
http://www.tup.tsinghua.edu.cn
责任编辑：马云艳
印 刷 者：北京牛山世兴印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×960 1/16　**印张**：28.25　彩插页：5　**字数**：670 千字
版　　次：2003 年 6 月第 1 版　2003 年 6 月第 1 次印刷
书　　号：ISBN 7-302-06646-9/TP・4974
印　　数：0001～4000
定　　价：49.00 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 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

.NET 最显著的特性是运行库，Microsoft 称它为公共语言运行库(Common Language Runtime，CLR)。运行库的概念在 Microsoft 技术中并不是刚刚出现的——Visual Basic 应用程序总是带有 Visual Basic 运行库的包袱。Microsoft 对 Java 领域的入侵带来了 Microsoft Java 虚拟机(JVM)。但是与 Visual Basic 运行库和 JVM 不同，.NET 运行库不局限于特定的语言。Microsoft 和第三方公司的几种语言都可以编写在.NET 运行库中运行的代码。其中有些语言比较新，如 C#，而另一些语言使用现有语言的语法。Microsoft Visual C++ .NET 是一种现有的语言，通过对它进行扩展可以产生.NET 代码，这些扩展被称为 C++托管扩展。

托管扩展使 C++类可以利用.NET 垃圾回收和内存保护机制。更重要的是，它们使 C++代码可以访问.NET 框架类库和用其他支持.NET 的语言编写的库，而其他语言可以使用 C++编写的托管库。C++开发人员要创建功能全面的应用程序不再需要使用 COM、DLL 导出函数和模板库等所有技术来访问它们所需要的库，所有必需的库代码都可以通过.NET 框架类库中的.NET 类获得。

从本质上讲，托管扩展定义了 C++语言的一个子集——它看起来像 C++，感觉像 C++，但是实际上是.NET。您可能会问："既然.NET 让笔者可以选择多种语言，为什么还要用 C++编写.NET 代码？"C++一直是一种系统语言，它为您提供了创建具有创造性的解决方案所需的能力和灵活性。这种特性也延续到了托管扩展中，在其中不仅有.NET 运行库和类库的全部功能，也有非托管语言的全部功能。确实，C++是惟一一种可以在同一个源文件中混合.NET 代码和非托管代码的语言。编译器还让您可以无缝地访问所有非托管库：静态链接库、模板库、COM 对象和 DLL。这种访问的方便性意味着可以重用所有现有的代码，并且当.NET 框架类库没有合适的类时，可以使用现有的非托管库。同样，没有其他语言可以有这种能力，所以其他语言不会被视为.NET 系统语言。

本书内容

本书将引导您从头到尾经历开发过程。首先描述语言的基本特性，然后探讨.NET 的特性，如 Interop、委托和 GUI 应用程序。本书的最后两章分别介绍 Visual Studio .NET 的项目管理和调试功能。用 C++开发.NET 代码不一定要用 Visual Studio .NET，但是正如您在第 6 章和第 7 章可以看到的，如果使用它，开发工作将容易得多。下面是每一章内容的更详细说明。

第 1 章

第 1 章讨论了托管扩展的基本特性。首先解释如何开发托管类型以及它们与非托管类型的区别，包

括它们的声明和用法。本章介绍了如何使用托管数组、接口和异常。以继承和强制类型转换(cast)来说，用托管扩展编写的 C++遵循.NET 模型而不是 C++模型，所以这一章的最后描述了.NET 在这些方面与非托管 C++的不同。

第 2 章

使用 C++的理由之一是它允许在.NET 项目中使用现有的非托管代码。托管扩展编译器有一种称为 It Just Works! (IJW)的技术。这种技术使您可以在托管项目中使用非托管库，并且混合托管和非托管类。这一章会介绍如何使用 IJW，并揭示 IJW 的工作原理。

.NET 还有一种名为**平台调用**(platform invoke)的基于属性的技术，它可以让所有支持.NET 的语言访问从 DLL 导出的代码。本章解释了如何使用平台调用，并描述了如何自定义它所执行的封送处理。平台调用的一个变种是 COM Interop，这是这一章的最后一个主题。COM Interop 让托管代码可以使用 COM 对象，就像它们是.NET 对象一样；也使非托管代码可以使用.NET 对象，就像它们是 COM 对象一样。本章将讲述 COM Interop 是如何工作的，以及如何注册 COM Interop 所需要的类并生成 COM Interop 所需要的属性。

第 3 章

在非托管项目中，函数指针很有用，因为通过它们可以在运行时而不是在编译时进行函数绑定。C++虚函数和 COM 接口是基于函数指针的，函数指针使您可以定义通知系统。.NET 有自己的函数指针——**委托**(delegate)——它是类型安全的，因为它消除了非托管函数指针的一大缺点：即在函数指针类型之间进行强制类型转换。

这一章将展示如何在 C++中使用委托、这种方式与非托管函数指针的区别，以及如何对非托管代码使用委托。本章还将解释如何通过委托进行异步调用(使用系统提供的线程)，并且介绍如何用.NET 编写多线程代码。最后，将讲述.NET 如何用委托实现名为**.NET 事件**(event)的正式通知机制。

第 4 章

.NET 框架用一种称为 GDI+的图形库扩展了 Windows。这是一个非托管库，但是.NET 框架带有.NET 包装类。在.NET 中的窗口技术称为 Windows 窗体。可以用 GDI+在窗体上绘图，可以将窗体作为控件的容器。这一章将解释如何在 C++中用 Windows 窗体创建 GUI 应用程序，并描述如何用 Win32 窗口实现这种应用程序。本章还展示如何通过.NET 事件处理 Windows 消息，以及如何绕过这种机制，从而取得对窗口行为的最大控制权。

本章还将讨论如何在托管类中有效地使用托管资源和标准资源，使得应用程序不再需要它们时可以释放这些资源。最后，本章定义了什么是托管资源，并解释了如何向应用程序中添加托管资源，还讨论了如何本地化资源。

第 5 章

这一章描绘.NET 代码是如何存储在可执行文件中的。首先解释.NET 程序集的格式，并且描述它们是如何实现为 Win32 可移植可执行(PE)文件的。然后讨论如何用.NET 框架提供的 COM 对象得到有关.NET 元数据以及程序集中的代码的信息。.NET 运行库是用非托管代码实现的，Microsoft 将运行库设计为非托管代码可以通过 COM 对象访问运行库。这一章将解释如何用这些对象从非托管代码中访问并配置运行库，以及如何指示运行库运行托管代码。

托管应用程序可以通过与该应用程序相关联的 XML 文件进行配置。当应用程序启动时运行库读取这个配置文件，这样它就可以得到有关应用程序所需要的资源的信息。运行库的一大好处是它只会加载生成应用程序时指定要使用的库。您可以配置这些规则——运行库通过配置文件查找这些库时要用到这些规则。代码也可以访问配置文件中的信息，这一章将展示如何做到这一点以及如何扩展配置文件和 API 以读取它们。

最后，本章描述代码访问安全，并展示如何在代码中应用这种安全机制。本章还展示用 C++托管扩展编写的.NET 代码所需要的默认权限。

第 6 章

Visual Studio .NET 本身是一个集成了各种应用程序开发工具的应用程序，是托管和非托管应用程序的混合体。这一章将介绍如何利用这个环境开发项目。本章讨论了编译器的功能以及用来管理项目的工具。本章还介绍了有关 Visual Studio .NET 项目向导和可开发的 C++项目的类型。最后介绍了一个可开发托管项目类型的例子，并描述了如何定制项目向导所提供的代码。

第 7 章

开发周期的最后一个阶段通常是测试阶段：您需要测试项目以保证它按照预期的方式运行，否则，需要调试代码以确定问题所在。尽管测试阶段通常是在开发周期的最后，但是可以预先编写提供诊断信息的代码，使测试工作量大大减少。这一章描述了.NET 框架提供的诊断代码中的问题的工具，并解释如何收集这些诊断信息。

Visual Studio .NET 集成了托管和标准代码调试器，所以发现问题后，可以单步执行代码直到找到问题的根源。本章介绍如何使用调试器及其各种工具，以及在调试多线程代码和由多个进程组成的应用程序时需要考虑的特殊问题。最后，本章展示了如何分析代码。Visual Studio .NET 没有提供代码分析器，但是.NET 框架可通过用户提供的 COM 对象提供分析信息。本章给出了这种分析对象的一个例子。

附录 A

.NET 框架类库非常全面，其中的代码能够完成以前用 C 运行时库(CRT)或者标准 C++库可以完成的任何任务。这个附录用一系列的表展示了与最有用的 CRT 函数和标准库类相对应的.NET 代码。附录 A 是您使用.NET 时的起点。

附录 B

这个附录列出了进一步学习所需的资源。这个列表并不完全，肯定也不是有关.NET 资源的最好列表。不过，这里提供的资源曾经对笔者特别有用，希望您也可以从中受益。

系 统 要 求

前 5 章只需要 C++编译器(版本 13)。.NET 框架 SDK 可以从 Microsoft 公司的网站(*msdn.microsoft.com/netframework*)免费下载。作为.NET 框架 SDK 一部分提供的 C++编译器不能产生优化的代码，它也不能提供像非托管 ATL 属性提供程序这样的扩展，但是它是功能全面的 C++编译器，可以用于进行托管和非托管 C++开发。如果想要学习.NET 框架，那么可以从 C++编译器开始。

最后两章使用了 Visual Studio .NET 的功能。Visual Studio .NET 包括全面优化的 C++编译器，它还带有非托管库：完整的 CRT 库、标准 C++库和合并的 ATL (ActiveX 模板库)以及 MFC (Microsoft 基础类)库，可以从.NET 代码中访问所有这些库(*msdn.microsoft.com/vstudio*)。Visual Studio .NET 还提供了可以创建应用程序初始文件的代码向导、管理项目的工具、功能全面的编辑器和集成的调试器。如果要开发包含 10 个以上的类的项目，那么就应该使用 Visual Studio .NET。

支 持 信 息

如果您对本书有任何建议、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

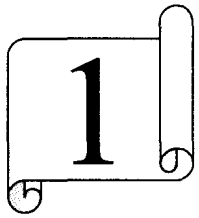
读者服务部

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



托管类型

C++托管扩展是对 C++编译器和链接器的扩展，以使它们可以创建.NET 代码。托管扩展使用 C++关键字和语法，但是它们的类型和功能遵循.NET 规则。所以，事实上，它成了语言中的语言。在有些情况下，.NET 具有一些在标准 C++中不存在的概念，在另一些情况中，它的一些项具有与 C++中的项类似的名称，而行为迥异。为了针对这些新功能对语言进行扩展并区别.NET 和标准 C++项，在语言中加入了一些新关键字。这些新关键字，加上一些新语法、两个新的杂注(pragma)和一个编译器开关构成了托管扩展，通俗地讲，就是托管 C++。

托管扩展是**扩展**——即，可以继续使用标准 C++，并且 C++的标准规则仍然适用于该代码。确实，所有现有代码都可以与为.NET 编译的代码(标准 C++、静态库和模板库)一同使用。

1.1 Visual C++ .NET 中的新关键字

为了区分为.NET 运行库编写的代码和不由运行库托管的代码，Microsoft 用表 1.1 所示的关键字引入了对 C++语言的扩展。

此外，编译器和链接器还具有用来编译.NET 代码的新开关，在第 6 章将对它们作更详细的介绍。最重要的新编译器开关是/clr。该开关告诉编译器将所有代码都编译为 Microsoft 中间语言(Microsoft Intermediate Language，MSIL)，不管代码是否由.NET 垃圾回收器托管。

表 1.1 C++中支持托管代码的新关键字

关 键 字	说 明
__abstract	指明类是抽象的，即没有实现所有的方法，而且要使用这个类，必须从它派生
__box	用于装箱值类型。装箱操作产生了携带被装箱值类型值的对象(有关装箱的更多细节参见本章1.2.2.2小节)
__delegate	声明委托类型。委托本质上是类型安全的函数指针

续表

关 键 字	说 明
__event	声明事件，这是作为类的一部分的通知机制。该关键字指明类可以生成事件，并且它添加代码以存储在事件被激发时调用的委托
__gc	指明类由.NET垃圾回收器托管或者指针指向托管对象
__identifier	当类型或者成员的名称是C++中的关键字时使用，并指示编译器忽略该词的C++含义
__interface	与__gc关键字共同使用时，__interface关键字可以用于声明托管接口
__nogc	用于指明该类型不被.NET垃圾回收器托管或者指明指针指向非托管对象
__pin	用于指针，以固定它所指向的对象。这种固定意味着在指针的作用域内，对象将不会在垃圾回收时从托管堆中移走
__property	指明方法是某个属性的get或者set方法
__sealed	指明该类是完整的，不能通过类派生来扩展
__try_cast	这个强制类型转换操作符执行运行时类型检查，并在强制类型转换失败时引发托管异常
__typeof	这个操作符用于为特定类型获取Type对象
__value	指明类型是轻型的，是在堆栈上而不是托管堆上创建的

在使用/clr 编译器开关时，在项目中的某个地方应该有#using 语句：

```
#using <mscorlib.dll>
```

该语句有两种功能。首先，它可以访问指定程序集中的元数据，这意味着可以使用在程序集中定义的公共类型。其次，#using 指示链接器在输出程序集中生成元数据，以标识输出程序集使用的程序集。每一个程序集都必须使用 mscorlib 中的类型，这就是为什么必须包括前面的#using <mscorlib.dll> 语句。注意，在#using 语句中给出的名称是包含元数据的**文件**的名称，而不是程序集的名称。

程序集的完整名称包含区域性(culture)、版本和公钥令牌，以及短程序集名。必须将您的程序集所依赖的程序集的这些信息(如果有的话)添加到该程序集的清单(manifest)中。如果想要使用安装在全局程序集缓存(GAC)——共享程序集的容器——中的程序集，那么将该程序集的正确的、完整的名称放到您所创建的程序集中是很重要的。GAC 中的程序集可以有几个版本，所以.NET Fusion 技术使用从属程序集中的元数据来确定加载哪个版本(Fusion 技术用于处理程序集定位和绑定)。#using 语句不在 GAC 中

查看元数据容器，所以必须提供该程序集在 GAC 外面的一个副本的名称(可能是完整的路径)。系统程序集(mscorlib、System、System.Windows.Forms 等)安装在 GAC 中，但是在%SYSTEMROOT%文件夹中的.NET Framework 文件夹中有它们的 DLL 的副本。#using 语句自动检查该文件夹。

#using 语句取用元数据容器的名称，用尖括号或者引号括住；使用哪一种形式并不重要。如果指定了路径，那么编译器将用该信息查找元数据。mscorlib 程序集是个例外。如果提供了到 mscorlib.dll 的路径，那么编译器会忽略路径信息。

搜索顺序为：

1. #using 中指定的完整路径
2. 当前工作文件夹
3. %SYSTEMROOT% 中的 .NET Framework 文件夹
4. 使用带有编译器开关 /AI 的命令行中提到的文件夹
5. 在环境变量 LIBPATH 中提到的文件夹

.NET Fusion 技术使用特定的规则(探测)在运行时查找程序集(第 5 章将对此做介绍)。#using 搜索顺序与 Fusion 的探测规则不同。

注意，前面提到的只是#using 将使用包含元数据的文件的名称，而不是说您必须提供程序集。可以在程序集、模块和.obj 文件中找到元数据，用#using 指定其中的任何文件。如果提供了程序集的名称，那么程序集的细节就会添加到所创建的程序集的清单中，这样 Fusion 将可以在运行时探测和绑定到程序集。如果要在程序集中使用导入模块中的类型并希望模块成为程序集的一部分，就要提供模块的名称。这样，所创建的程序集的清单将含有该模块的元数据。最后，可以在#using 指令中使用.obj 文件。只要编译源文件，.obj 文件就将具有.obj 文件中的类型的元数据。如果希望使用.obj 文件中的类型，那么文件还必须链接到所创建的程序集。在这一方面，#using 类似于标准 C++的#include。用#using 指定库程序集时，将只能够访问该程序集中定义的公共类型(在本章后面将介绍如何声明公共类型)。如果对.obj 文件使用#using，那么将可以同时访问公共和私有类型。

在.NET 中，可执行文件是程序集，并可以导出类型。不过，不要因此对可执行文件使用#using。尽管 C++编译器可以编译这种代码，但是在代码运行时，.NET 运行库将会产生报怨，因为在加载程序集时，运行库将发现程序集不是库程序集(DLL)，所以引发 BadImageFormatException 异常。

要将 C++编译为 MSIL，必须用/clr 开关调用编译器。#using <mscorlib.dll>语句和/clr 开关同时使用——如果使用其中一个，则必须使用另一个。该开关告诉编译器代码应该编译为 MSIL。所有托管代码都要编译为 MSIL，但是大多数标准(非托管的)代码也会编译为 MSIL。这意味着如果有将要在非托管 C++堆上创建的类，那么代码将仍然是 MSIL，并将由.NET 运行库运行。虽然有例外的情况(本章将在后面描述)，但是实质上所有代码都将编译为 MSIL。