P/Invoke，C++ Interop
和 COM Interop

附光盘

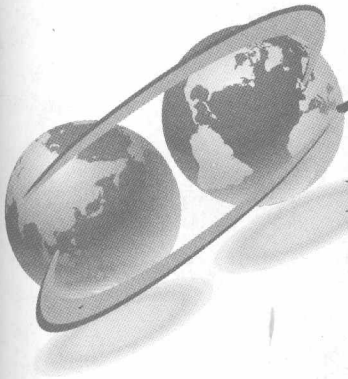
精通.NET互操作

P/Invoke，C++ Interop 和 COM Interop

- 使用P/Invoke调用C库函数及Windows API
- 使用C++ Interop与C++类库及核心算法库进行交互
- 使用COM Interop实现托管代码与COM之间的交互

黄际洲 崔晓源 编著

人民邮电出版社
POSTS & TELECOM PRESS



精通.NET互操作

P/Invoke，C++ Interop 和 COM Interop

黄际洲 崔晓源 编著

人民邮电出版社
北京

图书在版编目（CIP）数据

精通.NET互操作：P/Invoke，C++ Interop和COM Interop / 黄际洲，崔晓源编著. —北京：人民邮电出版社，2009.5
ISBN 978-7-115-20434-9

Ⅰ. 精… Ⅱ. ①黄…②崔… Ⅲ. 计算机网络－程序设计 Ⅳ. TP393

中国版本图书馆CIP数据核字（2009）第026781号

内 容 提 要

本书介绍 Windows 平台上的托管代码与非托管代码之间进行互操作的各种技术，包括由.NET 提供的各种互操作方法、属性以及各种工具的用法及其工作原理。本书包括 3 部分，平台调用——主要用于解决在托管代码中调用非托管程序设计语言编写的 flat API（如 Win32 API、C/C++风格的 API 等）的问题；C++ Interop——技术专门用于解决托管代码与 C++编写的非托管代码之间的互操作问题；COM Interop——介绍了使用 COM Interop 解决在托管代码中调用 COM 组件，以及在 COM 中调用托管类型的问题。

本书适合所有在开发过程中需要涉及到托管代码与非托管代码进行交互操作的.NET 开发人员阅读使用。不论是开始学习.NET 编程的开发人员，还是刚刚接触互操作的资深.NET 开发人员，都能从本书中获益。

精通.NET 互操作：P/Invoke，C++ Interop 和 COM Interop

◆ 编　　著　黄际洲　崔晓源
责任编辑　刘　浩

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京顺义振华印刷厂印刷

◆ 开本：787×1092　1/16
印张：27.25
字数：628 千字　　2009 年 5 月第 1 版
印数：1－3 500 册　　2009 年 5 月北京第 1 次印刷

ISBN 978-7-115-20434-9/TP

定价：68.00 元（附光盘）

读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154

对本书的赞誉

托管代码与非托管代码之间的交互是许多程序员在.NET 开发平台上不得不面对的任务。《精通.NET 互操作：P/Invoke,C++ Interop 和 COM Interop》这本书深入而透彻地解析了.NET 支持的三种与原生代码互操作的技术，作者以自己的经验讲述了原生代码与托管代码之间互操作所涉及到的编程要点，以及背后的一些实现原理。书中提供的实例有助于程序员快速领会并掌握.NET 与原生代码互操作技术的用法。我建议在.NET 平台上工作的程序员读一读这本书。

——潘爱民（著名技术作家，著有《COM 原理与应用》等多部畅销书，并翻译了多部经典名作，如《深入解析 Windows 操作系统》（第四版）、《C++ Primer 中文版》（第三版）、《COM 本质论》、《计算机网络》（第四版）等。他现在在微软亚洲研究院从事系统与安全方向的研究工作）

与非托管代码进行互操作是.NET 编程领域里面一个比较难的问题，因为它要求程序员对托管和非托管两个世界都有精深的了解。然而，互操作技术也是.NET 框架最根本的基石之一，包括文件操作、网络通信、GUI 界面等大量的.NET 类库都由平台调用而实现。事实上，掌握了与非托管代码互操作的技术之后，.NET 程序员能够编写的程序类型可以在一夜之间倍增很多，而不只是仅限于 ASP.NET 或 WinForm 程序的编写了。

在我做.NET 培训的时候，很多听众或者学员都对 P/Invoke 和 COM Interop 技术表现出了很大的兴趣，却又苦于没有 C/C++的开发背景，在编写与非托管代码进行互操作的代码时可谓举步维艰。其实当年我在自己摸索其中的规律时，也耗费了大量的时间和精力。在各种.NET 技术社区，关于非托管代码互操作的问题不绝于耳。该书的及时出现为以后的学习者铺平了道路。

这本书上的很多内容不是 MSDN 或者网上可以找到的，互操作代码的编写永远都是个复杂的任务。程序员只有掌握了其中的规律，了解了托管和非托管两个世界中程序的运行原理，才可能编写出高质量的互操作代码。本书能够极大地帮助程序员掌握这门技术。

——夏梺（2005 年微软最有价值专家（MVP），CSDN 技术论坛.NET 版大版主（网名 Sunmast（速马））。《.NET 企业服务框架——应用.NET 企业服务开发分布式业务解决方案》的译者之一）

.NET 平台是个托管的世界，提供了与以往完全不同的编程模型。毋庸置疑，.NET 是当前及未来 Windows 平台下开发技术的主流。但转到.NET 平台并非意味着使用.NET 将现有的 C/C++/COM 代码进行重写。.NET 提供的丰富互操作技术使开发人员能够通过这些技术将.NET 代码与非托管代码进行集成，以重用现有非托管代码。可惜的是，市场上大部分.NET 书籍对互操作技术少有涉足，中文资源尤其缺乏。本书深入且详尽地介绍了如何在托管代码与非托管代码之间进行互操作，涉及到了互操作技术的各个方面，是本非常有价值的参考书，高度推荐！

——金雪根（2002-2005 年微软最有价值专家（MVP），CSDN 技术论坛.NET 版大版主（网名 Saucer(思归)）。《.NET 企业服务框架——应用.NET 企业服务开发分布式业务解决方案》的译者之一）

从 2000 年.NET 平台问世以来，.NET Framework 已经从当初的 1.0 发展到目前的 4.0。.NET 为软件开发过程提供了一种新颖、高效的编程模型，因此广受软件企业及程序员们的青睐。但很多时候，程序员还需要在.NET 中重用已有的、经过严格测试的非托管代码。虽然.NET 平台提供了重用非托管代码的互操作技术，但是由于托管与非托管编程模型之间存在的巨大差异，因此掌握这些互操作技术并非易事。我曾经就为托管代码与非托管代码的交互操作问题而挠首不已，如果那时就有这么一本具有针对性的专业书籍，我就能省去很多的摸索时间和寻觅解决方案的痛苦过程。本书对.NET 平台提供的各种互操作技术的方法和原理进行了深入且全面的介绍，并通过实际问题介绍了各种情况下的最佳实践。据我所知，本书是国内第一本专门介绍.NET 互操作性的书。对于.NET 开发人员来说，本书无疑是一本难得的好书，非常值得一读。

——陈缘（2005-2008 年微软最有价值专家（MVP），CSDN 技术论坛 VB 版版主（网名 supergreenbean（超级绿豆）），《.NET 2.0 应用程序调试》译者）

在微软的技术格局中，风头正劲的现代软件开发平台.NET 所代表的托管代码世界，与已经早已取得实质性市场认同的非托管软件开发世界（比如 C++、COM 等）相比，编程模型完全不同。以微软尽量保持向下兼容的一贯设计风格，.NET 的设计中自然引入了各种互操作技术，以使这两个世界的往复沟通成为可能。然而多年的实践告诉我，这一领域就像是朵月中花——看上去很美，可真要“得道”却并非易事。.NET 提供的互操作技术远比想象中要复杂和晦涩的多，这本国内外少有的专门全方面讲解.NET 互操作技术的著作的出现实属难能可贵、雪中送炭。本书的两位作者把在该领域内多年摸索与实践的宝贵经验系统化地汇集在本书中，相信一定能够帮助读者深入理解和掌握.NET 互操作技术。

——金戈（2004-2008 年微软最有价值专家（MVP），《代码大全（第 2 版）》首席译者）

微软发布了.NET Framework 框架，这并不意味着要抛弃一切旧的技术，那些长期积累下来的非托管代码编写的东西，经过实践的检验，曾经带来过巨大的价值。将这些原有的代码转换成托管代码是不现实的，如何在托管代码与非托管代码之间进行互操作，继续使用原有的代码，本书给出了详细的解答。本书阐述了托管代码与非托管代码进行交互操作的主要技术，既有理论，又有详细的例子。相信读者通过阅读本书，能够全面掌握.NET 互操作这一技术。

——孟宪会（2002-2008 年微软最有价值专家（MVP），微软中文技术论坛版主，CSDN 技术论坛.NET 及 Web 开发版大版主（网名 net_lover（孟子 E 章）），《ASP.NET 2.0 应用开发技术》作者，《Eric Meyer 谈 CSS（卷 1）（卷 2）》译者）

.NET 提供了与非托管代码进行交互操作的支持。但是由于互操作代码经常很复杂，即使资深的开发人员也经常需要互操作方面的帮助。本书对托管代码与非托管代码进行互操作的技术做了详尽的介绍。书中每个部分都有配套的示例，指导读者处理各种互操作问题。本书对于需要经常进行互操作的开发人员来说是一本很好的参考书。

——蒋晟（2004-2008 年微软最有价值专家（MVP），CSDN 技术论坛 VC/MFC 版主，微软 MSDN 中文论坛 Visual C++版主）

前　言

自从2000年微软公司.NET平台问世以来，全球已经有超过400万开发人员使用.NET平台进行软件开发。对于.NET来说，这无疑是一个巨大的成功。这不仅仅体现在商业上的成功，其核心价值在于.NET为基于微软Windows平台的软件开发过程提供了一种新颖、高效的编程模型。在该模型下，开发人员能够更容易地将精力集中在其特定的开发情景中，而不用过多地关注消息循环、窗口过程等操作系统底层的处理。目前，基于.NET平台的技术和开发环境正处于飞速发展的时期。在本书即将出版之际，微软公司已经正式发布了.NET Framework 4.0所支持的新特性以及预览版。

另一方面，由于历史的原因，在.NET出现之前，开发人员已经编写了大量经过严格测试且可复用的非托管代码。它们以C库函数、C++类库以及COM组件的形式存在于诸多应用程序和框架之中，并承担着非常重要的角色。但由于在托管和非托管对象模型之间，数据类型、方法签名和错误处理机制都存在很大差异，从而使两种编程模型之间的代码互用和移植更加复杂。因此，在很长一段时期内，开发人员必须面对.NET与久经考验的“遗留代码（legacy code）”长期并存的局面。当然，开发人员可以选择.NET平台，使用托管代码重写这些已有的非托管代码。但这个重写的过程势必会枯燥无味，而且项目经理也不会在项目进度中安排大量的时间以进行重写代码的工作。更让开发人员感到尴尬的是，很多时候，即使花了很大代价对非托管代码进行了重写，但还是不能保证重写后的托管代码像那些久经考验的非托管代码一样正常或高效地工作。因此在很多情况下，重用已有非托管代码就成了最经济、可行的解决方案。以下是这些情况中的一些典型案例。

- 开发人员所在的部门一直使用第三方提供的COM组件为产品的核心功能提供支持。而新业务要求使用.NET平台。这就出现了一个问题。一方面公司已经为这些COM组件投入了大量的资金，不会轻易放弃这些组件。另一方面开发部门使用.NET平台进行开发，无法直接使用这些COM组件。因此，有效地在.NET平台中重用这些COM组件就成为产品成功的关键要素。
- 虽然.NET平台为开发人员提供了强大的框架类库（Framework Class Library，FCL），但它并没有包含基于Windows开发过程所涉及的全部Windows API。因此如果开发人员希望使用那些.NET平台尚未支持的编程接口，则需要找到在.NET中调用Win32 API的方法。
- 在产品中可能会存在一些核心算法模块，它们对产品的质量有很大的影响。因此测试人员对这类模块的要求有时会十分严酷。开发人员不但要保证模块能够在规定时间内完成计算任务，还要保证它在规定的内存空间内得以实现。在这种情况下，使用C++等非托管语言编写的模块与使用.NET编写的程序集相比，在性能上会有较大的提升空间。

可见在不同的开发情境中，开发人员会主动或被动地在开发过程中引入托管代码与非托管代码共存的情况。幸运的是，公共语言运行库（Common Language Runtime，CLR）

提供了一系列能够使托管代码与非托管代码进行交互操作的解决方案。其中主要包含 3 类互操作技术，如图 1 所示。

- 平台调用技术（P/Invoke）：主要用于处理在托管代码中调用 C 库函数及 Win32 API 函数等非托管函数的情形。
- C++ Interop：适用于在托管代码与 C++类库、核心算法库之间进行高效、灵活的互操作过程。一方面托管代码可以通过包装类机制使用 C++类库，另一方面非托管代码可以通过包装模板机制使用托管对象。
- COM Interop：该技术用于处理托管代码与 COM 之间的交互过程。托管代码通过运行库可调用包装（RCW）使用非托管 COM 组件。反过来，非托管 COM 客户端可以通过 COM 可调用包装（CCW）使用托管程序集。

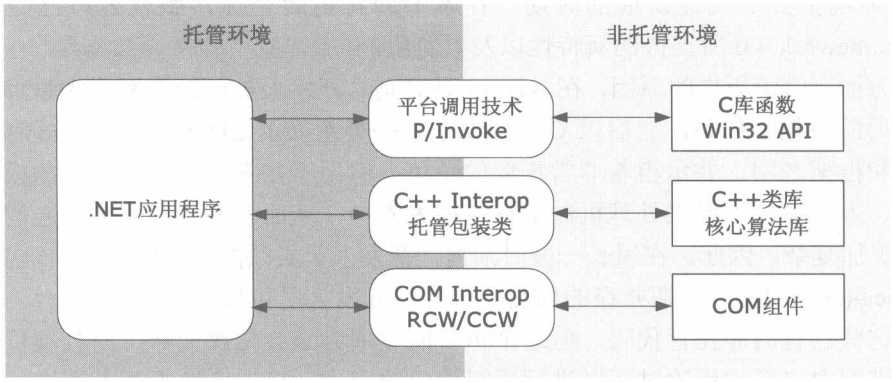


图 1　.NET 提供的 3 类主要互操作技术

由于不同的非托管对象，其设计和运行机制等存在很多差异。因此，托管代码与这些非托管对象进行交互操作时，在数据类型处理、错误处理机制、创建和销毁对象的规则以及互操作方法上，都需要根据不同的情况，分别进行不同的处理，从而导致互操作技术复杂多变且不易掌握。本书凝聚了作者多年使用互操作技术的经验，以严禁的态度和独特的内容组织结构与行文思路，系统且全面地介绍了在面对上述 3 种典型的开发情境时，如何在数据封送、性能优化和代码调试等环节均采用最佳实践。本书不但能够帮助读者快速找到解决问题的方法，还能帮助读者深入理解互操作技术的工作原理，并从更高的角度领悟互操作技术的本质和精髓。

本书的内容及组织结构

在托管代码与非托管代码之间进行互操作，主要有 3 种技术。因此本书将分为 3 个部分，共 6 个章节逐一对各项互操作技术进行详细介绍。

- 第 1 部分平台调用

本部分包含第 1、2、3 章，重点介绍平台调用（Platform Invocation Services，简称 P/Invoke 或 PInvoke）技术。平台调用技术主要用于解决在托管代码中调用非托管程序设计语言编

写的 flat API（如 Win32 API、C/C++风格的 API 等）的问题，其基本原理是在托管代码中重新声明一个与使用非托管语言编写的函数相等价的函数，并设置平台调用必需的一些属性及其字段，从而实现在托管代码中调用非托管函数。

- 第 2 部分 C++ Interop

本部分即第 4 章，主要介绍 C++ Interop（在早期的.NET 版本中，又被称为 IJW，即 It Just Works）技术。C++ Interop 技术专门用于解决托管代码与 C++编写的非托管代码之间的互操作问题。使用 C++ Interop，无需在托管代码中进行任何特殊声明或设置平台调用所需的属性，即可在托管代码中直接使用 flat API 及 COM API。开发人员甚至可以将托管代码和非托管代码定义在同一源文件中。因此，与 COM Interop 相比，C++ Interop 的功能显得更加强大，但使用 C++ Interop 要求开发人员对托管代码和非托管代码都具备深入的理解，才能避免诸如因数据封送错误、内存泄漏问题等引发的程序错误或异常。

- 第 3 部分 COM Interop

本部分包含第 5、6 两章。分别介绍了使用 COM Interop 解决在托管代码中调用 COM 组件，以及在 COM 中调用托管类型的问题。与平台调用技术非常相似，CLR 为 COM Interop 提供了完备的支持。本书不但会用代码示例解释位于 System.Runtime.InteropServices 命名空间下的各种 API 和属性，还将介绍.NET 平台为 COM Interop 专门提供的实用工具及其用法。

请不要忘记查阅本书的配套光盘，它包含本书所有示例的源代码。每个示例都是为介绍互操作的各种技术细节而精心设计的。此外，光盘还包含了许多实用的工具和资源。具体内容可参考本书的附录部分。

本书宗旨

本书旨在介绍 Windows 平台上的托管代码与非托管代码之间进行互操作的各种技术，这意味着本书将介绍由.NET 提供的各种互操作方法、属性以及各种工具的用法及其工作原理。虽然开发人员可以在微软的技术文档中找到相关的介绍，但其中很大一部分都没有对应的示例代码。因此开发人员很难清晰地理解一些方法和工具的用途及用法。此外，在有些情况下，.NET 为某些互操作问题提供了不止一种解决方案。因此，介绍如何选择最优的解决方案以及采用最佳的实践方式也是本书的目标之一。

本书适用于所有在开发过程中需要涉及托管代码与非托管代码进行交互操作的.NET 开发人员。不论是开始学习.NET 编程的开发人员，还是刚刚接触互操作的资深.NET 开发人员，都能从本书中获益。这是由于本书的构成区别于以往基于知识点的成文思路。本书在内容的设计上着眼于开发人员在开发过程中可能会遇到的各种问题，并以实际问题为背景将各种技术细节的介绍融汇于最佳实践之中。因此读者在阅读本书时，既可以根据自身的实际情况在书中直接找到解决问题的方法及示例代码，还可以根据本书的结构系统且全面地学习互操作技术的各个方面，从更高的角度理解互操作技术的本质和精髓。从而使本书不但有着“传道”的作用，还具有“解惑”的功能。

示例代码和开发环境

本书示例代码存放在配套光盘中。为了编译和运行示例代码，需要读者配置自己的开发环境。书中的示例代码是在 Visual Studio 2005（Service Pack 1）环境下开发的。.NET 平台是基于.NET Framework 2.0（Service Pack 2）x86 版本的。示例中的所有非托管代码都使用 C++编写，因此需要 Microsoft Visual C++ 2005 开发环境。而示例中的所有托管代码都使用 C#编写，因此需要 Microsoft Visual C# 2005 开发环境的支持。本书第 6 章使用了 VB6 编写 COM 客户端的示例，读者需要安装 Visual Basic 6.0 才能编译并运行该示例代码。

在配套光盘中还为读者提供了经过编译后生成的示例可执行程序。读者在安装了.NET Framework 2.0 的机器上即可运行示例程序查看结果。

读者还可以在支持 x64 平台的开发环境中修改示例代码的项目属性，从而观察示例在 x64 平台上的运行结果。但有一点小遗憾，目前 Visual Studio 的调试器（debugger）暂不支持 x64 平台下的托管代码与非托管代码的混合调试模式。

关于作者

黄际洲　2004-2007 年连续四年被微软授予微软最有价值专家（MVP）称号。感兴趣的研究方向主要包括自然语言处理、聊天机器人等。他曾翻译了三本游戏编程方面的书籍：《Direct 3D 中的 2D 编程》、《游戏编程 All in One》及《DirectX 角色扮演游戏编程》。

崔晓源　就任于微软亚洲研究院创新工程组，负责前沿研究成果的转化和新产品孵化。曾参与过 Live Search 拼写检查模块的研发工作。他最喜欢的项目是在线“电脑对联”第二版（http://duilian.msra.cn）。目前他正致力于下一代企业搜索技术和社会网络计算相关的研发工作。

致谢

在本书编写期间，我们得到了许多师友的帮助和支持。他们不但对本书的结构设计、成文思路提出了很多建设性的想法，还无私地分享了自己的宝贵经验。在此，作者谨向以下人士表示特别感谢。

感谢我的良师益友邹欣先生。他的支持推进了本书的成功出版。作为畅销书《移山之道》和《编程之美》的作者，他对本书的选题和定位都给出了中肯的建议，在此表示感谢！

感谢著名技术作家潘爱民老师对本书题材的认可以及对我们的鼓励，他还特别审阅了本书中涉及 COM 互操作技术的相关章节，并提出了不少宝贵的意见。

感谢殷丽丽和周萍女士，她们从写书之初一直到书稿完毕，都给予了我们极大的鼓励和支持。她们的大力支持为本书的如期出版提供了坚实的保障。

感谢重庆大学出版社副总编辑陈晓阳、编辑袁江（时任）及王斌，是他们的无私帮助和指引才让作者进入了书籍翻译和写作的领域。

感谢师弟陈远和邵毅，他们利用业余时间为本书搭建了专业的交流平台 www.interop123.com，使得大家得以充分利用社区优势学习和分享互操作领域的知识。感谢殷丽丽和周萍女士，她们为本书的排版付出了辛苦的努力。感谢张萌和穆怡雯为本书制作了令人赏心悦目的封面和扉页。

感谢在学校求学时的导师们，导师宽容的鼓励和中肯的批评给了我们无尽的教益和启迪。

感谢家人对我们的关爱和支持，他们的关爱、鼓励和鞭策着我们努力前行。

本书能够尽快写完付梓，也是那些一直寄予我们关心而在此无法一一述及的所有朋友的信念激励的结果。对于他们的帮助和支持，再次诚挚地致以谢意！

技术支持

在写作过程中，我们尽可能做到对每个知识点都严格求证并通过实例验证，但由于时间和水平所限，书中难免有遗漏之处。如果您在阅读过程中发现了书中的错误或者有任何建议和指教，请到本书的交流平台（www.interop123.com）或通过邮件（interop123@gmail.com）报告错误或提出您的宝贵建议。您也可以在该网站上获得有关本书的最新勘误信息。您也可以通过出版社的刘浩编辑（电子邮件：liuhao@ptpress.com.cn）与本书的作者取得联系。我们将非常感谢您的宝贵意见。

编者

2009.01

目　录

第一部分　P/Invoke

第二部分　C++ Interop

第三部分　COM Interop

第1章
使用C/C++类型的非托管函数

公共语言运行库（Common Language Runtime，简称CLR）提供了一系列能够使托管代码与非托管代码进行交互操作的解决方案。如果非托管函数采用C/C++编写，并且采用动态链接库（DLL）的形式将其导出，那么就能使用平台调用（Platform Invocation Services，简称P/Invoke或PInvoke）技术在托管代码中调用这些非托管DLL导出的函数。

使用平台调用技术能够调用采用C/C++编写的flat API（如Win32 API、C/C++风格的API等）函数。从基本概念上看，平台调用显得非常简单，只需要在托管代码中为非托管函数编写一个托管定义，并设置平台调用所必需的一些属性及其字段，就能在托管代码中调用非托管函数了。但真要是实际使用起来，平台调用却远不像看上去得那么简单。由于非托管代码和托管代码在本质上有着很大的差别，因此增加了平台调用的难度。比如，数据类型在内存中表示形式的差异增加了为非托管类型选择封送（marshal）类型的难度。

本章重点介绍和平台调用有关的基础知识，以及进行平台调用和提升平台调用性能的一些技巧。首先，介绍平台调用的作用及基本概念，并采用Hello World!作为示例，让读者快速地建立对平台调用的基本印象。接着讲述如何获得非托管函数声明、如何为非托管函数编写托管定义等基础知识，让读者能全面理解和掌握平台调用技术。随后讲述如何处理平台调用中出现的错误和异常、如何释放非托管内存以及进行动态平台调用的方法，让读者能够掌握平台调用的一些高级知识。最后介绍如何优化平台调用的性能，让读者理解和掌握如何根据平台调用的性能瓶颈进行有针对性的优化。

1.1　平台调用简介

平台调用（P/Invoke）可以消除.NET 托管代码（managed code）和非托管代码（unmanaged code）之间存在的鸿沟，它允许托管代码调用在 DLL 中实现的非托管函数。

托管代码必须在公共语言运行库（Common Language Runtime）环境下才能运行，而非托管代码一般都是本机代码（native code），它是完全脱离于公共语言库运行的。二者运行机制的差异导致无法直接在托管代码中使用非托管代码，必须以平台调用作为它们交互的桥梁。

一般来说，在.NET 平台下，使用 Visual Basic .NET 和 C#编写生成的代码都是托管代码，而使用 Visual C++ .NET 编写代码时，则可以根据自己的需要使用托管代码或非托管代码。另外，使用 Visual Basic 6 或 Visual C++ 6 等创建的代码、Win32 DLL 及由 C/C++创建的 DLL 都属于非托管代码的范畴。

1. 需要平台调用的场合

（1）很多时候，由于项目开发前期采用 C/C++编写了很多 DLL，而且这些 DLL 经过了严格测试，能够很好地工作，但是后续版本却需要使用托管代码进行开发。在这种情况下，如果使用托管代码重写这些 DLL，不仅浪费人力、物力，而且还需要经过不断测试才能正确工作。为了节省人力和时间成本，需要在托管代码中重用这些非托管代码，利用平台调用可以很好地解决这个问题。

（2）使用平台调用，可以通过调用 Win32 API 实现公共语言运行库当前尚没有提供的一些功能。因为当初在设计和构建.NET Framework 时，设计小组评估了为使.NET Framework 支持 Win32 API 所实现的功能而需要完成的工作，结果发现 Win32 API 经过 10 多年的发展和不断完善，规模实在太过于庞大，这就导致开发小组根本没有足够的资源和精力为所有 Win32 API 函数编写托管实现、测试并编写文档。因此只能采取折中的方式，只优先处理最重要和最常用的 Win32 API 部分，而对于许多尚未实现对应托管版本的 Win32 API 函数，就只能交由平台调用进行拯救和处理了。

（3）托管代码在实现某些功能上的效率不及非托管代码，为了提高效率，可以使用 C/C++编写核心算法，然后在托管代码中对其进行调用，从而使整体效率提升。

2. 从托管代码中调用非托管函数的步骤

（1）获得要调用的非托管函数的信息。

（2）在托管代码中声明该非托管函数，然后设置 P/Invoke 所必需的一些属性。

（3）在托管代码中直接调用上一步声明的托管函数。

平台调用依赖于元数据在运行时查找导出的函数并封送其参数，图 1.1 显示了这一过程。

3. 非托管函数的平台调用过程

（1）查找包含该函数的 DLL。