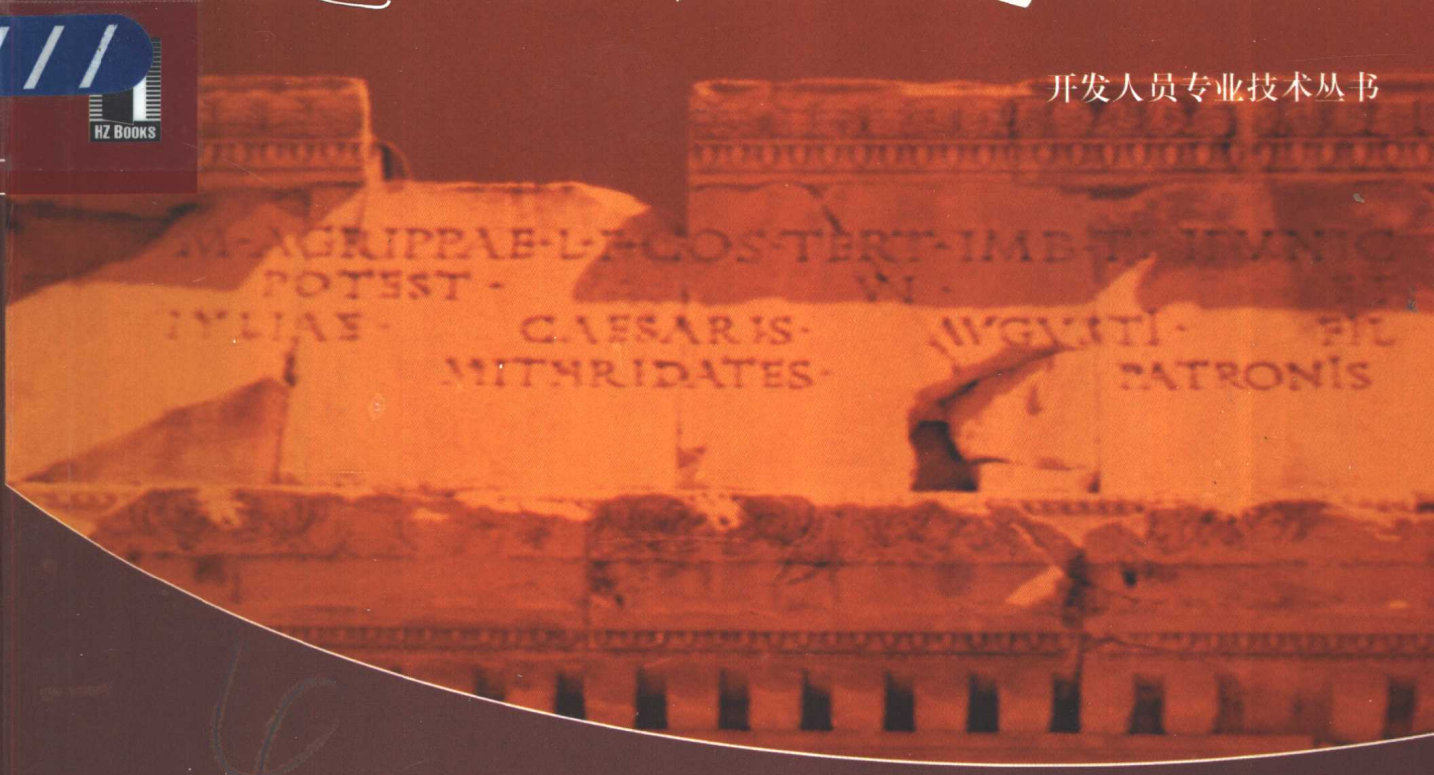
HZ BOOKS

开发人员专业技术丛书



Visual Basic .NET 面向对象编程指南

谈亮 蒋丹丹 等编著　刘艺 主审

从现在开始OOP不再是Java或C++程序员的专利

附赠

机械工业出版社
China Machine Press

开发人员专业技术丛书

Visual Basic .NET
面向对象编程指南

谈亮 蒋丹丹 等编著
刘艺 主审

机械工业出版社
China Machine Press

本书是一本全面实用的Visual Basic .NET面向对象程序设计指南。本书以精通Visual Basic .NET面向对象程序设计为目的，由浅入深地讲解了Visual Basic .NET面向对象程序设计的原则、方法和步骤，其中包括了许多宝贵的开发经验和有益的使用技巧。为帮助读者理解，书中还配有许多示例程序，这些示例程序全部基于Visual Studio.NET正式版本。

本书适合有Visual Basic开发经验并且希望掌握OOP方法以进一步提升层次的程序开发人员阅读，也可供大专院校相关专业的师生参考。

图书在版编目（CIP）数据

Visual Basic .NET面向对象编程指南/谈亮等编著. -北京：机械工业出版社，2003.1
（开发人员专业技术丛书）
ISBN 7-111-11180-X

Ⅰ.V… Ⅱ.谈… Ⅲ.BASIC语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2002）第087749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：武恩玉
北京第二外国语学院印刷厂印刷 · 新华书店北京发行所发行
2003年1月第1版第1次印刷
787mm × 1092mm 1/16 · 16.25印张
印数：0 001- 5 000册
定价：38.00元（附光盘）

前　言

微软已经将它的未来与.NET Framework紧密联系在一起，Visual Basic .NET很可能会成为未来几年中应用最广泛的开发工具，微软在Visual Basic .NET中添加了许多新功能，使之成为完全面向对象的编程语言，这是微软.NET战略的重要一环。较其前一版本相比，它具有众多引人注目的新特性。

我们写此书的目的是为使有经验的读者能够较为全面地掌握和使用Visual Basic .NET面向对象编程（OOP）技术。这是一本真正的从面向对象编程的角度讲述Visual Basic .NET的书，即使是熟练的Visual Basic 开发人员也将会从本书中受益。

需要声明的是本书完全根据Visual Studio.NET的正式发布版本撰写，所有程序也是在Visual Basic .NET的正式发布版本上调试运行通过。读者在阅读其他根据Visual Basic .NET测试版撰写的相关参考书籍时，可能会发现有些地方与本书提法不一致。

本书简介

第1章主要从Visual Basic开发人员的角度介绍.NET Framework，详细论述它的开发情况以及Visual Studio.NET IDE中的一些特性。

第2章讲述Visual Basic .NET的新特性，重点强调了命名空间的使用与完全继承。

第3章开始介绍面向对象程序设计的基本原理、术语及其发展。

第4章的内容包括对象及其公共接口、数据隐藏的概念及其重要性，以及一个面向对象设计的例子。

第5章对Visual Basic .NET的类和对象进行编程。我们将对类的公有和私有部分进行深入的研究。

第6章介绍Visual Basic .NET包含的一些对象工具和向导：引用对话框、对象浏览器和安装向导。

第7章进一步学习Visual Basic .NET的类。学习对象的生命周期以及一个类怎样成为一个对象。还将学习怎样创建自初始化的类以及完成使用后怎样销毁对象。

第8章使用多个实例介绍Visual Basic .NET一些比较重要的特性，包括：创建中间层组件、自由线程、交互操作。

第9章介绍封装带来的另一个重要概念——封装接口。包括访问Windows API和创建一个注册表/INI对象。

第10章是本书的结尾部分，讨论框架以及基于框架的开发。只有在深入领会了该章的内容之后，你才可能成为一个真正的面向对象高级开发人员。

本书读者

本书不是一本Visual Basic语言的学习教程，而是讲述在Visual Basic .NET中如何进行面向对

象编程的书，因此若想充分利用本书，熟悉Visual Basic语言是非常必要的。

本书主要面向有Visual Basic 6 经验的开发人员，面向想了解未来的.NET平台，并希望掌握流行的OOP技术，成为.NET平台上的高级编程人员。从现在开始，OOP不再是Java或C++程序员的专利。

.NET Framework提供了一种强大的、适合于Windows和Internet的新编程方法，这种方法会将使用其他语言和平台的有经验的开发人员吸引到.NET平台和Visual Basic .NET语言上。由于本书给出了许多OOP编程实例和技巧，因此读者能够迅速地从其他平台和语言过渡到Visual Basic .NET。即使你不打算进行过渡，本书的面向对象程序设计的原则与技巧对你依然是有帮助的。

使用本书的要求

运行所附光盘上全部样例代码所需的软硬件，大致地说，至少应具备以下条件：

- Pentium II，450MHz CPU，128MB的RAM和3GB的硬盘空间
- Windows 2000
- Visual Studio.NET正式版
- Internet Explorer 5.5
- IIS

致谢与反馈

徐勤建、胡金华、屈晓旭、高宇宁参加了本书部分章节的撰写工作。此外，为本书付出辛勤劳动的还有洪蕾、吴英。他们为本书的顺利出版付出了心血，在此一并表示感谢。

书中难免存在疏漏之处，恳请读者不吝赐教。如果读者有问题，或是在看书过程中遇到困难，可以使用以下电子邮件地址与作者联系：ltan@jlonline.com。

刘　艺

二〇〇二年八月于南京

目　录

第1章 .NET Framework入门

2000年7月，在佛罗里达的奥兰多，微软率先在Professional Developers Conference（PDC）上推出了它的.NET产品时，大多数人都觉得这是微软平台的Windows软件世界中的一个转折点。我们第一次见到了.NET Framework，这是一个崭新的开发和应用平台，一个为Internet设计的平台。

微软的.NET产品基础雄厚，不同凡响，它包括.NET Framework（包含了语言和操作平台）以及提供了丰富内置函数的扩展类库。除了核心的.NET Framework外，.NET产品还包括许多协议（例如Simple Object Access Protocol，通常简称为SOAP），提供了通过Internet集成软件的新环境，基于服务器的系列产品。.NET Enterprise Servers是微软BackOffice的下一代产品。

本章将从Visual Basic开发人员的角度讲述.NET Framework。由于Visual Basic.NET中的多数变化都以.NET Framework为基础，因此首先介绍.NET Framework，这是掌握后续章节所讨论的改进的前提。

对于每个使用微软技术的人来说，.NET Framework都意味着变化，而Visual Basic开发人员会比其他人受益更多。微软开发.NET产品的一个目的是把所有语言平台的精华集成到一起。从广义上讲，这意味着其他语言可以获得许多Visual Basic的易用性特征，如简单地拖放就可以创建窗体，同时Visual Basic也开发（或引进）了新的动态功能，如完全的面向对象的功能，它可以使Visual Basic开发人员摆脱以前的许多限制。

1.1 一个面向未来的开发平台

把.NET Framework称为“平台”并不是说明它应用范围如何广泛，功能如何强大。它包含一个可以省去开发过程中大量Windows API的虚拟机。它还包含一个拥有更多功能的类库，提供了一个跨越多种语言的开发环境，以及一个使多种语言集成更简单、更直接的体系结构。

初看上去，.NET的某些方面与以前的语言结构类似，如UCSD Pascal和Java。毫无疑问，.NET的某些思想受到以前技术成果的启发，但也有许多独特的新构思。总之，它是一个本质上全新的软件开发方法。这是首次基于Internet设计的开发平台。从前，Internet功能只是简单地连在Internet出现前的操作系统上，如UNIX和Windows。这要求Internet软件开发人员懂得大量技术和集成问题。而.NET Framework是为分布式软件而设计的，可以让系统中包含的Internet功能和交互性比以前更简单、更透明。

> .NET Framework是为分布式软件而设计的，利用XML使运行在一个组织内部的或世界各地的不同计算机上的功能单元集成到一个应用程序上。

这个微软.NET是全局的分布式系统，利用XML使运行在一个组织内部的或世界各地的不同

计算机上的功能单元集成到一个应用程序上。在这个版本中，从服务器到无线掌中宝或其他产品的各种系统都可以共享同一个平台，而且各种版本的.NET对它们都适用，彼此也可以进行透明的集成。

这并没有把已有的经典应用程序排除在外。相反，微软.NET也努力使传统的商务应用程序更容易开发和安装。.NET Framework的一些技术，如Windows Forms，表明微软并没有忘记传统的商务开发人员。

1.2 .NET Enterprise Servers的作用

微软已经发布了几种产品，它们都是.NET Enterprise Servers系列产品。.NET Enterprise Servers中的系列产品包括：

- SQL Server 2000
- Commerce Server 2000
- BizTalk Server
- Exchange 2000
- Host Integration Server（SNA Server的替代产品）
- Internet Security and Administration（ISA）Server（Proxy Server的替代产品）

这些产品的某些市场宣传强调它们是微软.NET战略的组成部分。但重要的是用户要理解这些产品与作为Visual Basic.NET基础的.NET Framework之间的区别。.NET Enterprise Servers不是基于.NET Framework的。其中大多数是以前基于服务器产品的替代品，且仍使用与以前产品一样的COM / COM+技术。

这些.NET Enterprise Servers仍旧在未来的软件开发工程中扮演重要的角色。当开发实际的.NET Framework工程时，大部分功能还要依靠.NET Enterprise Servers中的技术去实现，如数据存储和传递。第一个真正基于.NET Framework的产品是Visual Studio.NET（它包含 Visual Basic.NET）。

1.3 现有工具存在的问题

从1995年后期开始，微软向Internet做出重大转移。公司集中力量将Windows平台与Internet结合起来，并取得了一定的成功：将Windows打造成为重要的Internet平台，以及使用Windows DNA（Distributed Network Architecture）编程模型开发的面向所有商业软件的平台。

但是，微软也得作出某些重要的妥协，快速开发基于Internet的工具和技术。特别是 ASP（Active Server Pages）看起来已经有些笨拙。毕竟，编写大量解释脚本是结构化和面向对象开发方式的一种倒退。设计、调用和维护这种非结构化的代码也是很令人头痛的。

其他语言，如Visual Basic已经用在微软平台的Internet应用程序中，但大多数还像组件一样通过Active Server Pages工作。现在，对于理想的Web开发工作来说，微软工具还没有实现集成化和易用化。有人试图在传统的语言中设置Web接口，如Visual Basic中的Web Classes，但是它们都没有为大多数人接受。

> 微软在Visual Studio .NET中用ASP .NET取代了ASP，优点很多，逻辑运行已不再是解释过程，而是编译过程了。ASP+是ASP.NET以前的名称。

微软曾经试图用Windows DNA应用程序的一些概念使这种混乱状态变得有序。DNA为基于COM的三层标准开发描绘了一个广阔的前景，在这种三层开发方法中，Active Server Pages（和Win32客户机）在显示层，事务对象在中间层，关系数据存储和引擎在最底层。

1.3.1 显示层

在Windows DNA中，有两种主要的用户界面选项：Win32客户机和基于浏览器的客户机。

Win32客户机是最容易创建的，它提供一个更丰富的用户界面，它通常由可视化的开发工具（如Visual Basic）创建。不足之处是这种客户端软件很难安装和维护，它需要在每台客户机上安装，需要升级时，必须对每台客户机进行更新。除了为客户机安装软件以及维护/更新时很麻烦外，还有另一个很难解决的问题。由于安装在客户机上的操作系统和其他软件的版本不同，客户机上的DLL冲突很频繁。这些冲突很难诊断和解决，被称为“DLL Hell”[⊖]。

基于浏览器的客户机较难创建，它提供了一个比较有限的用户界面，只包含几个控件，而且对屏幕布局和屏幕事件处理的控制也很少。但它们很容易安装。所有客户机只需要一个兼容的浏览器和一个Internet或Intranet连接。

基于浏览器的客户机可以使用一些技术，如客户端脚本或者Java小程序，使用户界面更丰富，功能更强。这些方法都适用于当前大多数浏览器。但使用这些技术会增加额外的开发时间，也不能解决所有的用户界面问题。

还有一些“内部”的方法。如果客户机受到某种浏览器的限制，可以使用Dynamic HTML（DHTML）为界面增加更多的功能。如果客户机只能使用Internet Explorer，也可以用ActiveX控件使界面更接近Win32客户机上的界面。但是ActiveX控件本身会增加布署问题。ActiveX控件可以在Visual Basic中使用，但是布署这些控件要求在客户机上有许多支持Visual Basic的DLL。ActiveX控件一般是在C++中编写的，安装时很麻烦。并且这会增加开发时间，而且要求开发人员有很高的专家级水平。

> 在Web界面上使用ActiveX控件这一技术始终没有流行起来，只是在局域网内部它才被谨慎地使用。随着Visual Studio .NET的出现，Web Forms技术将取代ActiveX控件。

在DNA模型中常被忽略的一个重要因素是，需要使用基于Win32和Internet的用户界面。或者说需要有不同层次的用户界面，比如面向偶然的用户或新用户的用户界面，以及面向高级用户的用户界面。

1.3.2 中间层

在DNA应用程序中，中间层应该尽可能多地封装事务处理过程。除了客户机上检验数据所

⊖ 在本章“.NET Framework的核心要素”一节中对“DLL Hell”有更详细的解释。

需的规则外，大部分事务规则都应该在这个层里。

中间层经常分为几个子层。一个层处理客户的接口，一个层处理事务规则，另一个层处理数据仓库的接口。

Visual Basic是编写中间层组件最常用的语言。这是一种比基于窗体的典型Visual Basic程序更为高级的开发方式，要求开发人员非常精通COM和面向对象的编程概念。同时理解如何创建伸缩自如的组件也很重要，这通常意味着应使用Windows NT上的Microsoft Transaction Server或Windows 2000上的COM+ Services开发可执行的组件。这些组件向来使用非状态化设计，与在基于客户机的组件中常用的状态化设计完全不同。

构造中间层时理解COM的细微差别非常重要，因为该层中的组件必须一起工作。使所有的组件拥有适当的版本以致它们能了解相互间的接口是一件不容易的事。

中间层的组件可以与不同的协议和组件对话，将数据传达到数据层。

1.3.3 数据层

大多数商务应用程序必须存储信息以备长期使用。存储机制的本质随安装的不同而不同。通常要求有一个关系型数据库系统（RDBS），最常用的是Microsoft SQL Server和Oracle，如果信息大都由文档和消息构成，则需要一个消息数据库，如Exchange。许多安装仍取决于以前的主机系统。

除了保存数据外，数据层还有处理数据、检索数据和验证数据的逻辑功能。由SQL语言的某些变体编写的存储过程可以在RDBS数据库中使用，实现上述功能。

1.3.4 有关DNA模型的讨论

DNA所隐含的概念是非常合理的，但实际应用中会遇到许多挑战。例如，在一个DNA应用程序中，有许多场合都需要编程逻辑。如：

- 在窗体中编写Visual Basic代码。
- 用Visual Basic编写客户机上的组件。
- 用Visual Basic编写服务器上的组件。
- 用Visual Basic Script或JavaScript在Active Server Pages中编写服务器端脚本。
- 用Visual Basic Script或JavaScript编写客户端脚本。
- HTML、DHTML、CSS（Cascading Style Sheets，层叠样式单）。
- XML、XSL。
- 用C++编写ActiveX组件。
- 存储过程（用SQL Server中的Transact-SQL或者Oracle中的PL-SQL创建）。

有这么多选项，没有经验的开发人员很可能做出不正确的选择，例如，将属于服务器上的逻辑用在客户机上，或者创建Visual Basic脚本进行格式化，却不知道用CSS（层叠样式单）会更好。设计和构造一个基于DNA的复杂应用程序要求开发人员对许多不同的技术有很高的专业水平。

DNA应用程序通常需要服务器上有COM组件，有时也需要客户机上有COM组件。开发

COM组件需要具有一定的开发专业知识，并且这些知识要花很多时间才能掌握，当然，有一些语言，如Visual Basic更易于开发COM组件。

DNA应用程序的另一个大问题是布署。使COM组件的一个复杂的中间层正确工作，并使其与显示层和数据层正常连接，但达到此程度是非常困难的。所使用组件的版本和组件安装都会引发许多问题，但又必须使用这些组件来解决这些问题。

微软认为，虽然用基于Windows的技术可以编写出优秀的Internet应用程序，但还是特别希望能有更快速开发应用程序并且更容易地布署它们的方法。其他平台（如UNIX）和开发环境（如WebSphere）在开发Internet应用程序时仍存在障碍，因此，微软致力于解决DNA编程模型的局限性就显得很有必要。

用Visual Basic开发DNA应用程序的局限性

Visual Basic是开发具有DNA模式的应用程序最流行的语言。如前所述，它主要用于两方面——基于窗体的Visual Basic客户和COM组件（客户机、服务机上均可）。

当然也可以选择其他语言，包括C++、J++和各种第三方语言，如Delphi和Perl。但是用Visual Basic开发的人数远远超过了使用其他语言的人数的总和。

这并不意味着Visual Basic在这个环境中没有限制。最重要的限制包括：

- 没有多线程功能。
- 缺少继承性和其他面向对象的特性。
- 较差的错误处理能力。
- 与其他语言的集成性较差，如C++。
- 没有基于Internet应用程序的有效的用户界面。

例如，缺少多线程功能表明Visual Basic不能用于编写NT类型的服务程序，同时，也有这样的情况：由Visual Basic创建的组件使用单独线程时将影响性能。

Visual Basic面向对象功能的局限性使它不适合开发基于对象的构架结构，在这方面C++或Java开发人员就比Visual Basic开发人员有优势。

Visual Basic的错误处理技术在多层环境中非常令人懊恼。在Visual Basic中很难通过组件接口的堆栈跟踪和传递错误信息。

在一个COM应用程序中实现多语言集成也存在难度。Visual Basic的COM工具虽然很容易使用，但会出现这种集成问题。Visual Basic中的类参数是variant compliant，迫使要与Visual Basic集成的C++开发人员将参数转换成更适于使用的类型。在将Visual Basic中的组件集成到多语言工程中前，必须解决不同的数据结构和接口约定问题。除了需要附加的代码外，这些转变也可能产生性能问题。

当开发人员转向Internet时，使用Visual Basic的缺点就更加明显了。对于一台Win32客户机来说，Visual Basic窗体是很不错的，但对于有浏览器接口的应用程序来说，由于面临上述转换困难，如Web Classes和基于浏览器的ActiveX组件等问题，因此Visual Basic大多数时候会在组件中降级使用。

微软试图用Web Classes和DHTML页在Visual Basic 6中解决这个问题，但都没有成功。Web Classes提供一种模糊编程模式，以及可见布局上约束的控件。Visual Basic 6中的DHTML 页必

须向客户传递DLL，因此需要有一个宽带链接，这在很大程度上限制了互联网应用程序对它们的使用。

以上这些问题都需要解决，但微软决定跳出Visual Basic的固定圈子，在一个更广的层面上解决这些问题。所有这些限制都通过使用.NET Frameworks中的技术在Visual Basic .NET中得到解决。

1.4 .NET Framework简介

首先，.NET是一个构架，它包含了在操作系统上进行软件开发的所有层。它使微软或其他平台上的显示技术、组件技术和数据技术实现高度集成。第二，创建的整个结构更易于开发Internet应用程序，就像开发桌面程序一样简单。

> 如果你愿意，也可以将.NET Framework看作一个操作系统。实际上，你很难将其与Windows彻底区分开。

.NET Framework实际上“封装”了操作系统，将在.NET环境下开发的软件与操作系统的具体工作（如文件处理和内存分配）相脱离。这就使.NET环境下开发的软件可以在各种硬件和操作系统之间迁移。所有开发工作的公共底层微软.NET Framework 的主要组件如图1-1所示。

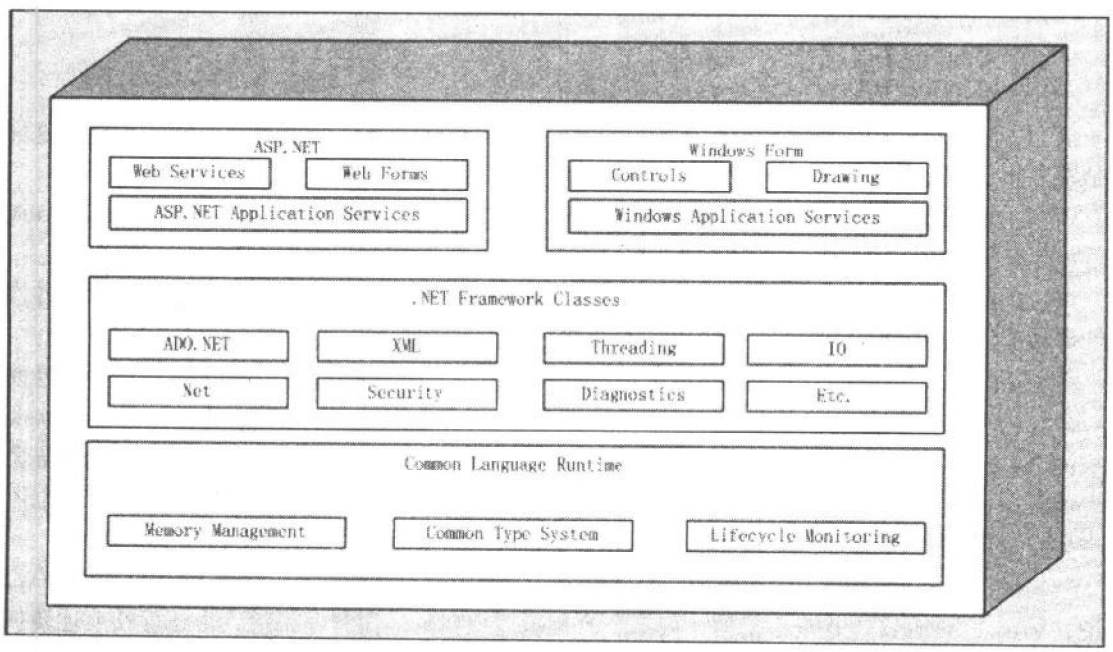


图1-1 微软.NET Framework 的主要组件

该结构从最下边的内存管理和组件加载层开始，向上到达显示用户和程序接口的多种方式。在它们中间的一些层可以提供开发人员需要的任何系统级功能。

最底层的Common Language Runtime（公用语言运行环境），经常缩写为CLR。这是.NET Framework的核心：它是驱动关键功能的引擎。例如，它包括一个数据类型的公共系统。这些公共类型加上标准接口协议就可以实现跨语言的继承。除了分配和管理内存外，CLR也可以进行对象跟踪引用和处理无用存储单元收集。

中间层包括下一代标准系统服务，如ADO.NET和XML。这些服务都在Framework的控制下，使它们能够在全世界范围内使用，并保持不同语言间用法的一致性。

顶层包括用户和程序接口。Windows Forms（通常称为WinForms）是一种制作标准Win32窗口的更高级的新方法。Web Forms提供了一个基于Web的UI。最具革新性的应是Web Services，它为程序通过Internet（使用SOAP）进行交流提供了一种机制。Web Services提供了一种类似COM和DCOM但基于Internet技术的工具，用于对象的调度连接，因此也可以与非微软平台集成。Web Forms和Web Services主要由.NET的Internet接口工具组成，可以通过.NET Framework中的ASP.NET执行。

有一个控制台接口允许创建基于字符的应用程序。用以前版本的Visual Basic很难创建这种应用程序。

以上所有功能都适用于以.NET平台为基础的任何语言，当然包括Visual Basic .NET。

1.5 公共语言运行时环境

我们都很熟悉运行环境技术：它们比DOS语言出现得还要早。但CLR比传统的运行环境技术更高级，它们之间就好比机枪与步枪相比。

1.5.1 主要的设计意图

CLR设计主要基于以下目的：

- 使开发过程更简单、快捷。
- 通道的自动处理，如内存管理和过程通信。
- 优秀的工具支持。
- 更为简捷、安全的布置。
- 可伸缩性。

下面详细介绍以上各点的具体内容。

1. 使开发过程更简单、快捷

广博、一致的结构可以使开发人员编写的代码较少，重复使用的代码较多。编写较少的代码是可以实现的，因为系统提供了一套丰富的底层功能。.NET中的程序可以用标准、一致的方式访问这些功能，与这些功能交互时，手工编写的代码和定制的逻辑都比较少。

由于有了标准化数据类型和接口约定，在.NET中编程也变得容易了。下面将要讲述.NET使掌握COM中错综复杂的技术不再那么重要。

2. 取消通道

许多编程的基础结构都由CLR自动处理或者完全无需处理。也就是说，一些结构是隐藏的，一些结构根本就不再存在。

内存管理就是隐藏基础结构的一个例子。Visual Basic 开发人员很久以前就不再担心内存问题。现在其他.NET语言的使用者也获得了同样的好处。

隐藏这些基础的结构简化了许多编程任务。下一节（.NET内存管理的内容）在讨论.NET中的内存管理时会提到，这种简化是非常有价值的。

许多通道都由元数据来替代，元数据是可以用一致性方式访问的组件、接口和过程的标准化信息。下一节将详细介绍元数据。

3. 工具支持

虽然CLR的用途大多与操作系统的功能类似，但它主要用于支持开发语言。它包含大量可用于各种工具的对象模型，如设计器、向导、调试器和配置文件。由于对象模型可以在运行期间使用，可以安排这些工具在所有使用CLR的语言之间工作。希望第三方软件也能创造出这种工具。

还要注意，微软并不对微软语言限制使用CLR。第三方语言的开发商也可以用 CLR重新建构他们的语言，这会使他们从中获益匪浅。除了要充分利用CLR的所有功能外 （也就是无需编写或支持它），用CLR还能进行各语言之间的集成，这在以前是从未出现过的。在下面“多语言集成和支持”一节会详细介绍此技术。

CLR与多种语言透明工作的能力会使开发人员获益更大。调试器就是一个最好的例子。CLR可以编写一个源调试器，它能同等地对待所有的语言，按需要从一种语言转向另一种语言。Visual Basic 开发人员可以通过访问这些更强大的工具而获益。

4. 更为简捷、安全的部署

熟悉Windows组件开发人员都知道，没有注册表、GUID和其他类似的工具，将很难明白系统是如何工作的，但CLR却无需使用上述工具。在.NET构架中创建的应用程序只需一个简单的XCOPY就可以安装，即将文件复制到磁盘上并运行此应用程序。自从DOS出现以来就没有见过这种操作。

能实现上述方法的原因是，.NET构架中的编译器将标识符（其格式是元数据，见下一节）嵌入到编译好的模块中，CLR会自动管理这些标识符。标识符提供了加载和运行模块，以及定位相关模块所需的所有信息。

作为一种优秀的附属产品，CLR可以管理同一组件的多种版本（共享组件也可以），让它们同时运行。标识符会告诉CLR某个编译好的模块需要哪种版本，因为编译时可以捕获这些信息。为了使用编译时可用的确切的组件版本、使用最新的兼容版本或指定确切版本。

初看起来，这里有一些不太明显的暗示。例如，如果程序需要直接从光盘上运行（先不运行安装程序），这在Visual Basic 3以后的版本中是不可行的，但在Visual Basic .NET中就可行。

> 在.NET中创建的应用程序不需要安装一个运行期间的模块，可以直接运行。

.NET中另一个重大部署优点是应用程序只需安装自己的核心逻辑。在.NET中创建的应用程序不需要安装一个运行期间的模块，例如用于ADO或XML的模块。这种基本功能是.NET Framework的一部分，它可以单独安装且每个系统只装一次，最终会包含在操作系统中，也可能包含在各种应用程序中（例如Internet Explorer）。那种一个Visual Basic “Hello,World”程序需

要安装4张软盘的日子已经成为过去。

实现程序自动运行还需要一个复杂的安全基础结构。.NET Framework可以捕获一条代码的出处，并用一个公共密钥标识模块的发布者。底线是系统设置为不运行不可信的软件，它提供阻挡病毒的机制，如臭名昭著的I LOVE YOU病毒。组件的方法无论在对象模型中有多强的功能，都必须在身份验证后才能运行。

5. 可伸缩性

因为大多数系统级的可执行函数都集中在CLR，所以经过优化和建构后，可以使微软.NET结构创建的应用程序具有更大的可伸缩性。与CLR的其他多数优点一样，可伸缩性可以毫不费力地用于所有的应用程序。

内存管理和过程管理可以内置可伸缩性。CLR中的内存管理是自我配置和自动调节的。无用存储单元收集（即回收不再使用的内存）进行了高度优化，CLR支持许多MTS/COM+的组件管理功能（如对象缓冲池）。使用无用存储单元收集功能可以加快组件速度，支持更多的用户。

这样做会产生一些有趣的副作用。例如，性能和可伸缩性在不同语言间的差别越来越小。所有的语言都编译成一个标准字节的代码Microsoft Intermediate Language （MSIL），经常简称为IL，下面会介绍CLR如何执行IL。把所有的语言都编译成相似的字节代码，在多数情况下，当出现性能问题时，就没有必要考虑其他语言。.NET语言间性能的差别很小——例如 Visual Basic具有与其他.NET语言相同的性能。

CLR早期规划的是能在各种设备上使用。最终这个构想成为让.NET在各级设备上运行，从小巧的掌上设备到Web群。也就是说相同的开发工具应该可以在整个范围内工作——有消息称这个构想受到了那些想用Windows CE开发软件包的人的拥护。当然，这是一个野心勃勃的计划，可能会有变化或被撤销。

1.5.2 元数据

.NET Framework需要应用程序的许多信息来执行非常多的自动功能。.NET的设计需要应用程序在其内部提供这些信息，即应用程序是自我描述性的。这种描述应用程序收集来的信息就称为元数据。

元数据的概念并不新鲜。COM组件就使用了它的一种形式，称为类库，类库包含了描述组件所使用的类的元数据，并用于执行OLE Automation。使用COM+的功能也需要提供更多的元数据来执行某些操作，例如，指定组件是否支持事务处理。

COM和COM+中元数据的一个缺点是元数据要存储在组件外部不同的地方。组件的类库存储在一个单独的文件中，而该组件的注册GUID（这是与组件的标识相关的元数据）存储在Windows注册表中。

然而，.NET中的元数据存储在一个地方——即在元数据描述的组件内部。.NET中的元数据也包含组件的更多信息，且更易于组织。

在.NET中，元数据是编译器生成的，自动存储在一个EXE或DLL文件中。它们都是二进制数据，但构架提供了一个API，可以从XML模式或COM类库中导出元数据。从XML模式导出比较有效，例如，可以从组件仓库中提取版本和编译信息。下面是为.NET Framework定义的元数