



VB.NET Developer's Guide



开发人员专业技术丛书

Visual Basic .NET

开发人员指南

跨人 .NET新纪元



(美) Cameron Wakefield 等著
Henk-Evert Sonder

智慧东方工作室 译



机械工业出版社
China Machine Press

SYNGRESS

824

7P3/3.4
v47

开发人员专业技术丛书

Visual Basic.NET 开发人员指南

(美) Cameron Wakefield 等著
Henk-Evert Sonder
智慧东方工作室 译

本书附盘可从本馆主页 <http://lib.szu.edu.cn/>
上由“馆藏检索”该书详细信息后下载,
也可到视听部复制



机械工业出版社
China Machine Press

本书是一本 VB.NET 应用程序的开发指南。它循序渐进地教您安装和配置 Visual Basic .NET 和 Visual Studio .NET, 该书全面讲述了新的集成开发环境 (IDE)、高级 VB.NET 编程概念、ADO.NET 结构和 XML Schema 定义工具, 并教您创建 Windows 窗体。本书还提供了数以百计的开发、部署及调试内容、安全警告和 VB.NET IAQ。本书讲解生动, 图文并茂, 附送包含实例代码的光盘。

本书适合于 Visual Basic 编程人员阅读。

Cameron Wakefield and Henk-Evert Sonder, et al: VB.NET Developer's Guide.

Original English language edition published by Syngress Publishing, Inc.

Copyright © 2001 by Syngress Publishing, Inc.

All rights reserved.

本书中文简体字版由美国 Syngress 公司授权机械工业出版社独家出版。未经出版者书面许可, 不得以任何方式复制或抄袭本书内容。

版权所有, 侵权必究。

本书版权登记号: 图字: 01-2001-4412

图书在版编目 (CIP) 数据

VB.NET 开发人员指南 / (美) 维克费德 (Wakefield, C.) 等著; 智慧东方工作室译. - 北京: 机械工业出版社, 2002.3

(开发人员专业技术丛书)

书名原文: VB.NET Developer's Guide

ISBN 7-111-08785-2

I . V... II . ①维 ... ②智 ... III . VB. NET 语言 - 程序设计 IV . TP312

中国版本图书馆 CIP 数据核字 (2002) 第 008288 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: 周 睿

北京昌平奔腾印刷厂印刷 · 新华书店北京发行所发行

2002 年 3 月第 1 版第 1 次印刷

787mm × 1092mm 1/16 · 31.5 印张

印数: 0 001 - 5 000 册

定价: 58.00 元

凡购本书, 如有倒页、脱页、缺页, 由本社发行部调换

目 录

译者序		1.9.3 以前版本的 VB 文件管理	15
前言		1.9.4 文件管理	15
配套光盘简介		1.10 自 VB 6.0 的改变	16
第 1 章 Visual Basic .NET 新特性	1	1.10.1 变体	16
1.1 概述	1	1.10.2 变低了的下界	16
1.2 新的 IDE	2	1.10.3 固定长度的字符串	16
1.2.1 界面增强	2	1.10.4 NULL 传播	16
1.2.2 开发加速	3	1.10.5 删除的其他项目	17
1.3 .NET 框架	4	1.10.6 属性和变量	17
1.3.1 一段非常简单的历史	4	1.11 小结	19
1.3.2 .NET 基本结构	4	1.12 本章要点	19
1.3.3 ASP.NET	5	1.13 常见问题解答	21
1.3.4 Framework 类	5	第 2 章 Microsoft .NET 框架	22
1.3.5 .NET 服务器	6	2.1 概述	22
1.4 公共语言运行环境	6	2.2 何谓 .NET 框架	23
1.4.1 历史	6	2.3 公共语言运行环境入门	23
1.4.2 集中性	6	2.4 使用与 .NET 兼容的编程语言	24
1.5 面向对象的语言	7	2.5 创建程序集	25
1.5.1 面向对象概念	7	2.5.1 使用表单	28
1.5.2 面向对象设计的优点	8	2.5.2 程序集缓存	29
1.5.3 面向对象和 VB 的历史	9	2.5.3 定位一个程序集	30
1.5.4 名称空间	9	2.5.4 私有程序集文件	34
1.6 Web 应用程序	9	2.5.5 共享程序集文件	34
1.6.1 Web 应用程序概述	9	2.6 理解元数据	34
1.6.2 Web 窗体	10	2.6.1 元数据的优点	34
1.6.3 Web 服务	10	2.6.2 根据元数据类型标识一个程序集	35
1.7 安全性	12	2.6.3 程序集依赖性	36
1.8 类型安全	12	2.6.4 反射	37
1.8.1 类型转换	12	2.6.5 结束 DLL Hell	38
1.8.2 数据转换	13	2.7 使用系统服务	39
1.8.3 按位运算	13	2.7.1 异常处理	40
1.9 新的编译器	14	2.7.2 垃圾回收	41
1.9.1 编译成可执行程序	14	2.7.3 控制台 I/O	41
1.9.2 基本结构	14	2.8 Microsoft 中间语言	42

2.9 用名称空间系统来组织类	42	4.2 组件结构	96
2.10 公共类型系统	43	4.3 被管代码与未被管代码的比较	98
2.11 依赖自动资源管理	46	4.4 系统名称空间	100
2.11.1 被管堆	46	4.4.1 文件 I/O	101
2.11.2 垃圾回收和被管堆	48	4.4.2 绘图	102
2.12 安全服务	52	4.4.3 打印	103
2.13 小结	56	4.5 公共类型系统	104
2.14 本章要点	56	4.6 垃圾回收	107
2.15 常见问题解答	57	4.6.1 对象的分配与回收	108
第3章 安装和配置 VB.NET	59	4.6.2 Close 与 Dispose	109
3.1 概述	59	4.7 小结	109
3.2 版本	59	4.8 本章要点	109
3.3 安装 Visual Studio .NET	60	4.9 常见问题解答	110
3.4 新的 IDE	65	第5章 .NET 编程原理	112
3.4.1 集成的开发环境自动化模型	65	5.1 概述	112
3.4.2 加载项	67	5.2 变量	113
3.4.3 向导	71	5.3 常量	115
3.4.4 宏	72	5.4 结构	115
3.4.5 主页	72	5.5 程序流程控制	117
3.4.6 项目选项	74	5.5.1 If...Then...Else	117
3.4.7 工具箱	76	5.5.2 Select Case	121
3.4.8 子窗口	78	5.5.3 While 循环	122
3.4.9 任务列表	81	5.5.4 For 循环	124
3.4.10 解决方案资源管理器	84	5.6 数组	125
3.4.11 属性窗口	85	5.6.1 声明一个数组	125
3.4.12 窗体布局工具栏	86	5.6.2 多维数组	126
3.4.13 隐藏/显示代码元素	87	5.6.3 动态数组	128
3.4.14 Web 窗体	88	5.7 函数	129
3.4.15 智能感知	88	5.8 面向对象编程	132
3.5 自定义 IDE	89	5.8.1 继承	133
3.5.1 自定义代码编辑器	89	5.8.2 多态性	133
3.5.2 自定义快捷键	89	5.8.3 封装	133
3.5.3 自定义工具栏	90	5.8.4 类	134
3.5.4 自定义内建命令	90	5.8.5 多载	137
3.5.5 自定义首页	91	5.8.6 覆盖	138
3.6 小结	93	5.8.7 共享成员	140
3.7 本章要点	94	5.9 字符串控制	141
3.8 常见问题解答	94	5.10 错误控制	143
第4章 公共语言运行环境	95	5.11 小结	145
4.1 概述	95	5.12 本章要点	146

5.13 常见问题解答	147	7.6.1 在窗体上锚定控件	203
第 6 章 高级编程概念	149	7.6.2 在窗体上停靠控件	204
6.1 概述	149	7.6.3 窗体上的对象分层	205
6.2 使用模块	150	7.6.4 在窗体上定位控件	205
6.3 利用名称空间	151	7.7 对话框	205
6.4 理解 Imports 关键字	155	7.7.1 显示消息框	206
6.5 实现接口	156	7.7.2 通用对话框	206
6.6 代表和事件	159	7.7.3 创建对话框	218
6.6.1 简单代表	162	7.8 创建和操作菜单	218
6.6.2 多址代表	162	7.8.1 在窗体上添加菜单	218
6.6.3 事件编程	162	7.8.2 动态创建菜单	220
6.7 语言互用	163	7.9 在窗体上添加状态栏	221
6.8 文件操作	164	7.10 在窗体上添加工具栏	222
6.8.1 目录列表	165	7.11 数据绑定	224
6.8.2 数据文件	166	7.11.1 简单数据绑定	224
6.8.3 文本文件	168	7.11.2 复杂数据绑定	224
6.8.4 追加到文件	170	7.11.3 用于数据绑定的数据源	225
6.9 集合	171	7.11.4 使用 Data Form Wizard	226
6.10 Drawing 名称空间	172	7.12 使用 Windows 窗体类查看器	228
6.10.1 绘图	174	7.13 使用 Windows 窗体 ActiveX 控件导入器	229
6.10.2 打印	176	7.14 小结	230
6.11 理解自由线程	179	7.15 本章要点	230
6.12 小结	181	7.16 常见问题解答	232
6.13 本章要点	181	第 8 章 Windows 窗体组件和控件	234
6.14 常见问题解答	182	8.1 概述	234
第 7 章 创建 Windows 窗体	183	8.2 内建控件	234
7.1 概述	183	8.2.1 Label 控件	236
7.2 应用程序模型	184	8.2.2 LinkLabel 控件	238
7.3 操纵 Windows 窗体	186	8.2.3 TextBox 控件	240
7.3.1 Windows 窗体的属性	186	8.2.4 Button 控件	243
7.3.2 Windows 窗体的方法	187	8.2.5 CheckBox 控件	245
7.3.3 创建 Windows 窗体	194	8.2.6 RadioButton 控件	246
7.3.4 改变窗体边框	195	8.2.7 RichTextBox 控件	247
7.3.5 改变窗体大小	197	8.2.8 TreeView 控件	249
7.3.6 设置窗体位置	197	8.2.9 ListBox 控件	250
7.4 窗体事件	198	8.2.10 ComboBox 控件	257
7.5 创建多文档界面应用程序	200	8.2.11 PictureBox 控件	261
7.5.1 创建 MDI 父窗体	200	8.2.12 TrackBar 控件	263
7.5.2 创建 MDI 子窗体	201	8.2.13 DateTimePicker 控件	264
7.6 在窗体上添加控件	202	8.2.14 Panel 控件	266

8.2.15	GroupBox 控件	267	区别	316
8.2.16	TabControl 控件	268	10.2.3	Web 窗体较传统 ASP 的优势
8.3	创建自定义 Windows 组件	269	10.3	在 Web 窗体里添加控件
8.4	创建自定义 Windows 控件	272	10.3.1	Web 窗体控件同 Windows 窗体控件的 区别
8.5	小结	274	10.3.2	ASP.NET 服务器控件
8.6	本章要点	274	10.4	创建自定义 Web 窗体控件
8.7	常见问题解答	275	10.5	Web 服务
第 9 章	使用 ADO.NET	276	10.5.1	Web 服务是如何工作的
9.1	概述	276	10.5.2	开发 Web 服务
9.2	XML 概述	277	10.5.3	Web 服务工具
9.2.1	XML 文档	277	10.5.4	在 Web 窗体中使用 Web 服务
9.2.2	XSL	277	10.6	在分布式应用程序中使用 Windows 窗体
9.2.3	XDR	277	10.7	小结
9.2.4	XPath	278	10.8	本章要点
9.3	理解 ADO.NET 结构	278	10.9	常见问题解答
9.3.1	ADO 和 ADO.NET 的差异	279	第 11 章	代码优化、调试与测试
9.3.2	XML 支持	279	11.1	概述
9.3.3	维持状态	280	11.2	调试概念
9.4	理解 XML Schema 定义工具	280	11.2.1	调试菜单
9.5	连接的层	282	11.2.2	监视窗口
9.5.1	数据提供者	282	11.2.3	断点
9.5.2	连接字符串	283	11.2.4	异常窗口
9.5.3	命令对象	284	11.2.5	命令窗口
9.5.4	DateReader	288	11.2.6	条件编译
9.5.5	DataSet	288	11.2.7	跟踪
9.6	未连接的层	290	11.2.8	断言
9.7	使用 SQL Server 数据提供者	296	11.3	代码优化
9.8	远程处理	299	11.3.1	终止
9.9	数据控件	299	11.3.2	转换
9.9.1	DataGrid	299	11.3.3	参数传递方法
9.9.2	DataList	304	11.3.4	字符串
9.9.3	Repeater	307	11.3.5	垃圾回收器
9.10	小结	310	11.3.6	编译器选项
9.11	本章要点	310	11.4	测试阶段及策略
9.12	常见问题解答	311	11.4.1	单元测试
第 10 章	开发 Web 应用程序	313	11.4.2	集成测试
10.1	概述	313	11.4.3	β 测试
10.2	Web 窗体	314	11.4.4	回归测试
10.2.1	一个简单的 Web 窗体	314		
10.2.2	Web 窗体同 Windows 窗体的 区别			

11.4.5 压力测试	379	13.5 部署控件	441
11.5 小结	380	13.6 小结	442
11.6 本章要点	381	13.7 本章要点	442
11.7 常见问题解答	381	13.8 常见问题解答	443
第 12 章 安全性	383	第 14 章 升级 VB 程序至 .NET	446
12.1 概述	383	14.1 概述	446
12.2 安全性概念	384	14.2 升级前的注意事项	446
12.2.1 权限	384	14.2.1 变量早期绑定	447
12.2.2 主体	385	14.2.2 避免空值传播	448
12.2.3 身份验证	385	14.2.3 使用 ADO	448
12.2.4 授权	385	14.2.4 使用 Date 数据类型	449
12.2.5 安全性策略	386	14.2.5 使用常量	450
12.2.6 类型安全	386	14.3 移植前考虑结构的问题	450
12.3 代码访问安全性	386	14.3.1 Intranet/Internet 应用程序	450
12.4 基于角色的安全性	400	14.3.2 客户机/服务器和多层应用程序	451
12.4.1 主体	401	14.3.3 单层应用程序	452
12.4.2 基于角色的安全性检查	404	14.3.4 数据访问应用程序	452
12.5 安全性策略	406	14.4 数据类型	453
12.5.1 创建新的权限集合	408	14.4.1 Variant 数据类型	453
12.5.2 修改代码组结构	411	14.4.2 整型	453
12.5.3 远程安全性	416	14.4.3 日期	454
12.6 密码	417	14.4.4 布尔型	454
12.7 安全性工具	419	14.4.5 数组	454
12.8 小结	420	14.4.6 固定长度的字符串	455
12.9 本章要点	421	14.4.7 Windows API 数据类型	456
12.10 常见问题解答	423	14.5 将 VB 窗体转换为 Windows 窗体	457
第 13 章 应用程序的部署	425	14.6 关键字的变化	459
13.1 概述	425	14.6.1 Goto	459
13.2 代码打包	425	14.6.2 GoSub	459
13.3 配置 .NET 框架	429	14.6.3 Option Base	459
13.3.1 创建配置文件	430	14.6.4 AND/OR	459
13.3.2 机器/管理员配置文件	430	14.6.5 Lset	460
13.3.3 应用程序配置文件	431	14.6.6 VarPtr	470
13.3.4 安全性配置文件	433	14.6.7 StrPtr	460
13.4 部署应用程序	435	14.6.8 Def	460
13.4.1 公共语言运行环境	435	14.7 程序设计的差异	460
13.4.2 Windows Installer	436	14.7.1 方法的实现	461
13.4.3 CAB 文件	436	14.7.2 对未被管库的引用	467
13.4.4 Internet Explorer 5.5	437	14.8 属性	471
13.4.5 资源文件	438	14.8.1 用属性过程工作	471

14.8.2 控件属性名的变化	472	14.10.4 非连接数据访问	479
14.8.3 默认属性	473	14.10.5 数据导航	479
14.8.4 Null 的用法	475	14.10.6 加锁	479
14.9 理解错误处理	476	14.11 升级接口	479
14.10 Visual Basic .NET 中数据访问的 变化	478	14.12 使用升级工具	485
14.10.1 数据集和记录集	478	14.13 小结	489
14.10.2 程序的互操作性	478	14.14 本章要点	489
14.10.3 光标位置	479	14.15 常见问题解答	491

第 1 章 Visual Basic.NET 新特性

本章内容包括：

- 新的 IDE。
- .NET 框架。
- 公共语言运行环境。
- 面向对象的语言。
- Web 应用程序。
- 安全性。
- 类型安全。
- 新的编译器。
- 自 VB 6.0 的改变。

1.1 概述

在开始详细探索 Visual Basic.NET 之前，我们先大致了解一下这次新版本所做的更改和增添的所有新特性。这一新版本相对于上一版本已发生了重大更改。你可能需要花费一些精力来接受它，但我相信你一定会认为这些新特性确有学习价值。Visual Basic.NET 已不仅仅是 Visual Basic 6.0 的升级版，如同你所期望的那样，它添加了可增强集成开发环境（IDE）功能的新特性。现在，所有的 Visual Studio 开发工具都可共享同一个环境。例如，在 Visual Basic 和 Visual C++ 间切换时，你不再需要学习其他 IDE。此外，本版本还添加了一些很好的特性，这些特性大大地简化了开发过程，而这正是我们一直期盼着的。

Visual Studio.NET 构建在 .NET 框架上。这是与 Visual Basic 6.0 的关键不同之处。基于 .NET 框架的应用程序开发将把 Internet 视为新的操作系统，应用程序不再受硬件束缚。这是 Windows DNA 模型的革命之举。这种新的框架建立在开放的 Internet 协议上，用于将平台和编程语言之间的互操作性标准化。.NET 框架还允许创建新类型的应用程序。应用程序现在使用公共语言运行环境（CLR）来运行。所有的 .NET 应用程序都使用这个运行时环境，该环境使 Visual Basic 应用程序与其他编程语言在同等的基础上运行。通过使用 CLR，Visual Basic 具备了继承性并提供自由的线程技术，而这两点曾经是 Visual Basic 应用程序的致命缺陷。Visual Basic.NET 是面向对象的。任何东西现在都是对象，每个对象都从标准的基类中继承而来。CLR 的另一个优点是公共类型系统，这意味着所有编程语言都共享同一种类型，而这将极大地提高语言间的互操作性。

Internet 的发展已进入了一个新的阶段。最初，它只用于显示静态的 Web 页面。商业机构很快就发现这对它们没有多大帮助。接着，Internet 开始展现动态内容并充当电子商务平台。

而下一个发展方向应该是支持完全在 Internet 上运行的应用程序。Visual Basic .NET 倡导这种新的 Web 应用。通过 Web 服务，我们可以将对象存放在 Internet 上的任意位置，然后通过 Internet 上的任意应用程序进行调用（不再需要配置 DCOM）。当然，将应用程序扩展到 Internet 上会增加安全隐患。 .NET 框架内建有许多安全功能可保护你的应用程序。

类型安全在本版本中已得到增强。此安全机制可防止代码访问它无权访问的内存位置。你可通过类型安全来定义访问对象的方式。在代码运行前，此机制将对它进行验证，以确认它是类型安全的。如果不是，则只有在你设置的安全策略允许时才能运行。

本版本的 Visual Basic 有许多新变化。本章将概述总体性的更改。这将帮助你在详细探索以后各章节内容之前获得一个全景式认识。

1.2 新的 IDE

不论开发人员还是管理人员，你更关心的可能是转换到此新环境的难易程度，而不是它所具有每个新特性。微软了解这一点。在探索 VB .NET 用途的过程中，你会发现它将旧版 VB 和来自其他语言的特性智能地融合在一起。最明显的莫过于 IDE。微软在本版本添加了一些重要的新功能，这些功能提高了开发人员的工作效率，且无需他们改变原有的工作习惯。

如果你见过以前版本的 Visual Basic，则 VB .NET 的 IDE 与它非常相似。而且，如果你过去还用过 InterDev，则会发现其实许多新增的界面也很熟悉。这是因为用于 VB .NET 的新型 IDE 集成了以上两种环境的最大优点，从而可以提供更有效的工作方式。

当然，得到任何好处都是要付出一定代价的。本章后面及随后的各章将讨论与升级到 VB .NET 有关的部分问题，选择开发工具时必须权衡所有这些利弊。首先，让我们看一下 IDE 的部分新特性及其优势。

1.2.1 界面增强

尽管 IDE 已发生了很多变化，但你首先注意到的可能是已有功能的外观变化。以前版本的 Visual Basic 曾试图在不影响屏幕内容显示的情况下提供快捷方式以访问尽可能多的功能。表 1-1 中列出了 VB .NET 解决这些问题的部分方法。

表 1-1 界面增强

特 性	说 明	优 点
多显示器支持	开发人员可同时使用多个显示器进行显示	开发人员可在一个窗口中执行代码，同时在另一个窗口中进行调试，从而真切体验终端用户的感受
制表键控制的窗体	在开发环境中用选项卡来显示子 MDI 窗体。代码窗口、帮助屏幕、窗体设计窗口以及主页可相互叠放在同一个窗格中，并通过拖拽显示在最上方	尽管无法同时看到很多信息，但节省了屏幕空间

(续)

特 性	说 明	优 点
工具箱	控件以垂直方式显示而不是显示在网格中, 每个控件旁都带有说明	在以前版本的 Visual Basic 中, 你必须将鼠标放在控件上才能显示出它的名称 (当你设置了自定义的控件, 而控件在很多情况下都使用同一个默认的图标时, 这一点尤其令人懊恼)
可扩展的代码	使用与 Microsoft Word 的大纲视图类似的界面, 你现在可将代码分为几段, 然后通过单击鼠标将它们隐藏或展开	开发人员现在可看到代码的更高级的视图, 从而允许他们更有效地将代码在应用程序间迁移
帮助	无需按 F1 键, .NET IDE 现在可识别你正在做的工作并在单独的窗口中显示出上下文相关的帮助信息	开发人员可实时地不断获取精确的指导信息

1.2.2 开发加速

当然, 并非所有新的 IDE 特性都只是界面上的变化。VB.NET 的开发工具还提供了新的界面以便更有效地使用已有的功能。表 1-2 中讨论的特性在 VB 6.0 都可找到, 但它们现在允许开发人员以更快的速度生成应用程序。

表 1-2 开发加速

特 性	说 明	优 点
菜单编辑器	使用分工明确的菜单编辑器, 你可在相关的窗体上直接编辑菜单	以前, 你必须从 Tools (工具) 菜单中选择 Menu Editor (菜单编辑器)。此更改将加快开发速度并减少误用窗体带来的错误
解决方案资源管理器	与以前版本提供的项目资源管理器不同, 解决方案资源管理器提供了一个资料档案库供你查看并维护各种开发资源	你现在可管理不是用 VB 开发的组件 (使 VB 更好地与其他语言协同使用是推出 .NET 的驱动力之一)
服务器资源管理器	现在你可看到客户机/服务器中可用的服务器或 Internet 应用程序, 并可直接将它们的资源集成到你的代码中	以前需要手工执行的操作现在可通过拖放鼠标来完成。例如, 如果在 SQL 服务器上有一个存储过程, 则可直接浏览该过程并可直接在页面上进行更新

(续)

特 性	说 明	优 点
主页	启动 VB 时出现的屏幕现在是在使用 DHTML 编写的	你现在能够可视化地完成更多编程任务，同时减少出错的机会。例如，如果在 SQL 服务器上有一个存储过程，现在可直接浏览该过程并可将其拖到需要的面板上。VB 将会自动完成余下的编程操作

1.3 .NET 框架

了解 .NET 优势的最好方法就是对比以前版本的限制。在本节中，我们将简略介绍 Microsoft 组件交互的历史，然后再简介其基本结构。

1.3.1 一段非常简单的历史

在 Windows 3.0 问世时，最初用来在应用程序间通信的方法是动态数据交换 (Dynamic Data Exchange) 或 DDE。DDE 要占用大量资源、灵活性差且易导致系统崩溃。然而，它在单个计算机上的使用仍被人们广泛接受，在相当长的时间内，有许多应用程序继续使用此方案在应用程序间发送信息。

随着时间的推移，微软逐渐不再支持使用 DDE，而是提倡使用公共对象模型 (Common Object Model, COM) 和分布式 COM (DCOM)。COM 用于在单个计算机上的 Microsoft 应用程序间进行通信，而 DCOM 用于与远程主机进行通信。

在此期间，提供商联盟 (包括 IBM、Sun 和 Apple) 提出了一种在主机间进行通信的替代方案，称为 CORBA。与 COM 不同，CORBA 的强项是在不同操作系统间传递信息。不幸的是，此协议需要占用大量资源，且不易编程，最终结果是，盛名之下，其实难负。

与此同时，微软也在不断改进自己的技术，相继推出了 COM+、Microsoft Transaction Server (MTS) 以及分布式网络结构 (Distributed Network Architecture, DNA)。通过使用这些技术，组件间可进行更复杂的交互，如对象池、事件和事务等。但是，这些技术要求每个应用程序都非常了解其他应用程序，因此当操作平台为异构型 (例如，Windows 应用程序与 Linux 进行通信) 时，结果就差强人意了。

现在，让我们回到 2001 年推出的 .NET，它兼有 COM 的功能和 CORBA 的灵活性。尽管此技术主要是微软开发的，但它的灵活性和可伸缩性从理论上意味着，在不久的将来可在其他平台上使用。(虽然 .NET 框架可在包括 Windows 95 在内的所有更高版本的 Windows 操作系统上运行，以后还会推出可在 Windows CE 上运行的 .NET Compact Framework。)

1.3.2 .NET 基本结构

.NET 框架由三部分组成：公共语言运行时 (Common Language Runtime)、Framework 类

和 ASP.NET, 这些将在以下各节中进行探讨。由于 .NET 的组件易于引起混淆, 图 1-1 中列出了 .NET 的基本结构。

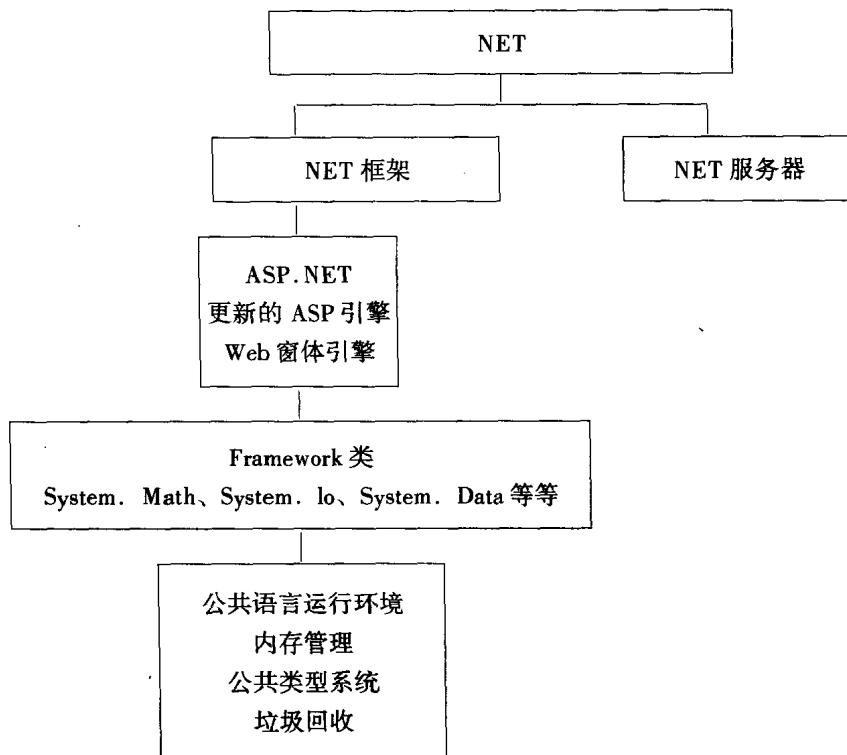


图 1-1 .NET 基本结构

1.3.3 ASP.NET

过去, 令 Visual Basic 开发人员头疼的主要问题是, 他们必须把经过编译的 VB 应用程序和在称为 VBScript 的简化版 VB 中编译的应用程序之间的不同调整成一致。更糟的是, 当活动服务器页面 (Active Server Pages) 推出时, 它支持的服务器端脚本语言是 VBScript, 而不是 VB。(从技术上来讲, 其他语言也可用作服务器端脚本语言, 但 VBScript 是最常用的。)

现在, 随着 ASP.NET 的出现, 开发人员有了新的选择。为保持向后兼容, .NET 现在还支持具有 ASP 扩展名的文件, 但同时也推出了 ASPX 型文件。ASPX 文件在第一次运行时进行编译, 它们使用的句法与独立的 VB.NET 应用程序相同。以前, 编写一个可执行编译后代码的简单的 ASP 页面, 许多开发人员得执行额外的操作, 现在可直接从活动服务器页面运行编译过的代码。

1.3.4 Framework 类

具有对比意义的是, VB.NET 的功能之所以更强大的一个原因恰恰是因为它所作的更少。

一直到 VB 6.0 为止，Visual Basic 编译器需要做的工作相对于 C++ 等同类型语言的编译器要多得多。这是因为 VB 内建的许多功能都是通过外部类以 C++ 来实现的。这易于更新语言和在其中添加新特性，同时也提高了共享同一个库的应用程序之间的兼容性。

现在，VB.NET 的编译器也采用此模型。以前属于 Visual Basic 的许多功能现在可直接通过 Framework 类来实现。例如，如果要计算一个平方根，将使用 System.Math 类中的一个方法来代替 VB 运算符。此方案不仅简化了 VB 语言还增强了它的伸缩性。

1.3.5 .NET 服务器

我们在此处提及这一点只是为了将 .NET 服务器与 .NET 框架区分开来。这些服务器支持 Web 通信，但它们自身并不一定是用 .NET 框架编写的。

1.4 公共语言运行环境

CLR 在程序代码和操作系统间构建了一个界面，以提供内存管理、公共类型系统和垃圾回收等功能。它的出现反映了微软致力于为所有基于 Microsoft 程序的代码提供一个统一和安全的框架，而不用考虑创建这些代码所用的语言。本章简介 CLR 的功能及其工作方式，详细信息将在第 4 章中进行介绍。

1.4.1 历史

许多年来，Visual Basic 的设计都注重在功能强大和使用简单之间寻找最佳结合点。为了使中间开发人员不涉及到 API 编程的复杂和危险，VB 开发人员受到许多限制。编译过的 VB 代码无法直接与 Windows API（通常以 C++ 编写）进行交互，而是通过运行时模块与之进行通信，这些模块用来处理数据收集和间接访问等繁琐任务。

由于上述原因，VB 和 C++ 的编程人员逐渐形成两个泾渭分明的阵营。实际上，许多 C++ 编程人员瞧不起 VB，认为它只适用于快速应用程序开发（Rapid Application Development），而不是严肃的企业开发项目的理想工具。他们还常常感到厌烦，因为不得不编写封装器以便 VB 开发人员访问 Windows API。所有这一切在 VB.NET 中全部改观。现在，由 Visual Basic 和 C++ 开发人员创建的代码都以同样的方式即 CLR 与 Windows 进行交互。（在这一点上，其他新的语言也一样，如 C# 或 JavaScript.NET。）

1.4.2 集中性

VB.NET 的一个优点就是现在可以使用 VB 来开发以前需要更低级别的语言来开发的应用程序，同时又不失 VB 开发的传统优势。不论你是开发人员还是管理人员，你都需要分析可用的不同工具以找到最佳组合，为更好地说明 VB 和 C 这两类平台的集中性，表 1-3 对比了它们处理四种关键问题的方式，其中包括它们以往不同时期的各个版本和 .NET 环境中的版本。

是否需要运行时间？从 VB 5.0 开始，Microsoft 即宣称 Visual Basic 程序可编译为真正的可执行程序，但更精确的说法可能是，运行时间模块只是变得更小并对用户更透明。相反，

C++ 从来无需运行时间模块。

表 1-3 VB 和 C 对比

问 题	VB1.0-4.0	VB 5.0-6.0	VB.NET	C++	C#
是否需要运行时间?	是	是	否	否	否
界面模型	COM	COM	CLR	COM	CLR
内存泄漏?	很少	很少	非常少	许多	非常少
是否支持继承性?	是	否	否	是	是

界面模型 使用 CLR，经过编译的代码不再与执行的代码完全相同，而是在客户机上进行转换（此方案的一些优势将在“新的编译器”一节中详细阐述）。在以前版本的 VB 和 C++ 中，代码经过编译以使用 COM，但在 VB.NET 和 C# 中，编译过的代码使用 CLR。

内存泄漏 VB 的一个传统优势就是内存由经过编译的可执行程序负责管理，而 VB.NET 中保留了这一优点，尽管这一工作现在 CLR 中进行。（相反，编写的不够好的 C++ 代码经常会出现此类错误，因为它在使用内存后不进行回收。）

是否支持继承性 这可能是 VB.NET 的最重要的改进，这些将在下节进行说明。（从 5.0 开始，VB 即开始支持部分继承性，这也在下节中进行详述。）

1.5 面向对象的语言

VB.NET 中最有价值的新特性可能就是真正的面向对象。尽管以前版本的 Visual Basic 也几乎是面向对象的，但只有 VB.NET 使开发人员真正享受到代码继承的益处，此功能使业务逻辑可更容易和更可靠地在组织内传播。在本节中，我们将简要介绍面向对象的设计的部分原理并说明它可为 VB 开发人员提供的帮助。

1.5.1 面向对象概念

有关面向对象设计的讨论完全可以写成一本书（实际上，许多人都这样做了），但我们在此处只作简单介绍。与以前的面向过程的程序设计方法相比，面向对象（OO）的语言的主要优势在于，你不仅可以将数据封装在基本结构中，而且还可封装行为。举例来讲，一辆汽车不仅能够描述成一堆螺栓、金属片和轮胎（属性），还可以描述为可加速和减速的对象（方法）。

OO 设计所需要的前期工作通常比其他环境要多，且设计开始时通常是先列出用于描述必须操作的对象的说明性语句。例如，如果你要使用面向对象的原理建造一辆汽车，可能会这样描述需求：

- 此汽车必须快速行驶。
- 此汽车属于交通工具类型。
- 此汽车为红色。

现在，我们已得到足够多的信息，可以开始定义我们需要的对象。总的来说，这些语句中

的名词描述了所要求的对象（在此例中为汽车）；其中的动词描述了对象必须执行的方法，而形容词描述了对象包含的属性。完成这些定义后，即可开发代码以支持这些需求。表 1-4 中总结了这些细目。

表 1-4 面向对象的术语

高级概念	词 性	例 子
对象	名词	汽车
方法	动词	快速行驶
属性	形容词	颜色 = 红色

1.5.2 面向对象设计的优点

面向对象设计的真正优点体现在当你要在对象间传播行为时。例如，如果你要建造一辆轿车和一辆双门小汽车，除车门数不同外（一个为四，一个为二），这两辆车的设计可能没有其他不同之处。

此时，即可使用继承性。如果你已设计了一辆轿车，则只需要覆盖车门数，然后继承该轿车的所有行为即可建造一辆双门小汽车。请看下列 VB 伪代码：

```
Public Class Coupe
    Inherits Sedan
    Overrides Sub BuildDoors()
        Doors = Doors + 2
    End Sub
End Class
```

现在，如果你要为轿车添加新特性（如侧面气囊），则这些特性将自动传播给双门小汽车而无需添加更多代码。

相反，如果你希望某个对象的方法根据传递给它的参数的不同而执行不同的行为，则需要使用多载功能。VB 会根据参数列表自动确定要运行的模块。表 1-5 中总结了覆盖和多载操作的区别。

表 1-5 覆盖与多载

类 型	覆 盖	多 载
方法名称	相同	相同
自变量列表	相同	不同
行 为	替换现有方法	补充现有方法