马子洋 编著

Delphi 2005

C# Builder

程序设计

机械工业出版社

C# Builder 程序设计

马子洋 编著

机械工业出版社

C# Builder 是 Borland ALM 宏伟战略中第一个针对.NET 平台的开发工具，也是 Borland 全力打造并寄予厚望的新一代开发工具产品。本书将介绍使用 C# Builder 开发.NET Windows Forms 应用程序、数据库应用程序、ASP.NET Web 应用程序和 Web Services 应用程序所必需的专业知识。

本书是一本实战性质的图书，它尽管时时考虑到如何引领 C# 和.NET 的新入门读者提高兴趣和水平，但其重点任务是深入指导使用 C# Builder 的.NET 开发人员掌握.NET 程序开发所必需的更深更实在的知识和实战技能。本书在概念说明时使用了大量的图、表；提供了近百个复杂而完整的实例代码，这些能帮助读者轻松地将所学的理论知识应用于实践。本书适合.NET 平台上的所有开发人员，特别是以 Delphi 2005 和 C# Builder 作为开发工具的.NET 应用系统开发人员以及广大在校学生阅读。

图书在版编目（CIP）数据

C# Builder 程序设计/马子洋编著. —北京：机械工业出版社，2005.3
ISBN 7-111-16254-4

Ⅰ. C… Ⅱ. 马… Ⅲ. C 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2005）第 018151 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：任正一 赵国平
版式设计：李元杰
北京牛山世兴印刷厂印刷 • 新华书店北京发行所发行
2005 年 3 月第 1 版第 1 次印刷
787mm×1092mm 1/16 • 52.5 印张 • 1025 千字
印数：0001-3000 册
定价：59.00 元
凡购本书，如有缺页、倒页、脱页，由本社发行部调换
本社购书热线电话（010）68327245，88379873

前 言

经过几年的发展，.NET 已经走进了绝大多数开发人员的视野，并成为许多开发人员真实生活中的一部分。在.NET 时代，过去以 Windows 为中心的应用程序将革新为以 Web 为中心的应用程序。就像当年从 DOS 转向 Windows 一样，它带来了 Windows 平台应用程序在性能和使用方式上的又一次飞跃。但说实话，微软推出.NET 之后，Windows 上的软件开发环境将发生革命性的变化。这对于开发人员和开发工具厂商来说，又是一场新的挑战。最受我们关注的开发工具厂商 Borland 面临的严峻考验是如何将 Win32 下原生（Navite）开发工具产品线延伸到.NET CLR 虚拟平台上并保持自己一贯的优势，在同微软的.NET 开发工具 Visual Studio.NET 的竞争中拥有一席之地。为此，Borland 主要采取了下面的步骤：首先在以 Win32 为基础的 Delphi 7 中支持.NET。Delphi 的这个版本中对 Object Pascal 语言做了很多修改以支持.NET。其次，Borland RAD 部门倾全力打造了支持标准 C#语言的全新的纯.NET 平台开发工具 C# Builder。之后，Borland 又推出了完全支持.NET Framework 的 Delphi 8（又称 Delphi For .NET），Delphi 8 中的语言在 Delphi 7 的基础又做了许多根本性的改变，被称为 Delphi 语言。最近，Borland 又发布了 Delphi 2005，Delphi 2005 将 C# Builder 和 Delphi For.NET 集成在一起，Delphi 2005 成为更加面向企业应用、支持建模、协作与集成的平台。

听到 Borland 在开发针对.NET Framework 的 C# 语言开发工具后，我一直固执地对这款产品看好，这或许是受以往 Borland 在开发工具领域杰出表现的影响，或许是作为使用 Borland 产品的一名老战士对 Borland 情深意笃的原因。尽管在.NET 平台上，任何一门编程语言提供的功能都只是.NET Framework 下一个子集的映射，具体的语言已经退居为一个语法表达的层次了，但 C#是微软专门为.NET 平台开发的全新编程语言。说其是专门为.NET 平台开发，主要是指 C#与.NET 平台在软件服务化、组件化、Internet 化、开发配置简易化、安全性可靠性最大化等方面具有天然的集成性。说其是全新的编程语言，主要是指 C#没有历史包袱，完全从零开始，吸收了 C（高性能）、C++（面向对象的结构）、Java（安全性）、Delphi 和 VB（快速开发、简单）等语言的众多经验，反映了当前软件开发从面向对象到面向组件、从桌面应用到 Internet 应用、从专有标准到开放标准（如 XML、SOAP）的最新趋势。所以，C# 是开发.NET 平台上以组件为基础的、多层分布式 Web 应用和 Windows 应用的最理想的语言。

.NET 平台反映了微软这位软件巨人对未来软件的思考，其几乎囊括了微软所有的最新技术成果。还清晰地记得.NET 发布之初微软对它的描述，当时真让人对其高远的架构蓝图以及相应的表现产生过

许多困惑，好在随着时间的推移这些都成为过去，现在如果您用俯视的眼光梳理.NET平台中的各项技术的内涵，就会发现在.NET平台中，.NET Framework始终占据着核心的地位，它是整个.NET平台的关键支撑。是为众多语言（如C#、Delphi、 Visual Basic.NET、Managed C++）和应用程序模型（如Windows Forms、Web Forms、Web Services 等）提供各种服务的重要基石；也是各种类型的.NET 应用程序在设计、开发、测试、部署、运行等诸多环节赖以依靠的基础。作为开发人员，它是必须坚持敲开的宝山。值得庆贺的是.NET Framework彻底改变了MFC时的表现，这或许是您在一大堆类、接口、API等内容中煎熬时能得到的惟一自慰。

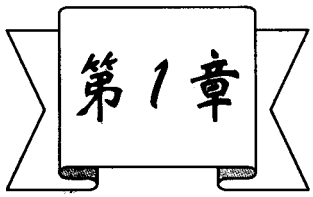
写程序的确是一份辛苦的工作，但如果有一天由于跟不上技术的进步让您不再写程序了，那未免也有些辛酸！笔者写作本书最初的目的是想把自己一个老程序员在学习及应用.NET时所经历的彷徨犹豫、挫折困顿以及日积月累后的渐悟或顿悟过程作一次回顾和反省，系统地记录下自己认为最容易迷惑和跌倒的地方，以便对后学有所帮助。笔者认为学习一门技术最好的方法还是理论和实践结合，所以在写作本书时笔者时刻努力希望使之落到实处，但这仅仅是笔者追求的理想境界，至于结果还请读者自己评判。我需要在此重点说的是这本拖了一年多的书稿要付印时自己依然忐忑不安，这主要原因是笔者深感自己水平不足，怕有负读者的期望，所以在此我要恳请读者能够将阅读过程中产生的任何疑问或对本书的任何看法告诉我，我在期待着与您的交流。

笔者还要在此感谢促成这本书的赵少平主任，为此书付出许多的张明主任、任正一老师、赵国平老师，还有微软SharePoint产品和技术咨询顾问马子耕先生及其他我无法一一致谢的朋友，这其中包括我的家人、我所有书籍的读者。

作 者

2005 年 3 月

目　录



理解 Microsoft .NET 技术体系

总是要跟着技术跑的。

只是在J2EE这个老兔子和.NET这个新兔子之间，必须要有个选择而已。我不知道发展会怎样，但我觉得我只能选择.NET这只兔子。

如果你不仅仅想追着兔子，还想打着兔子，那么我建议选择Delphi.NET。它是一把好猎枪，因为枪托上刻着：Made In Borland。

——周爱民

上面这段话摘自周爱民老先生在 2003 Borland Conference 上的演讲，此处引用它，绝对没有继续讨论这两大平台优劣之意，只是有感于追赶兔子的艰辛历程。确实是总要跟着兔子跑的，甚至是多只兔子!现在我们要追踪的是.NET，手里拿的是仍刻着 Made In Borland 的 C# Builder。

本章重点：

- 分布式应用系统及开发技术演进
- 传统 Windows 开发平台面临的挑战和.NET 思想
- .NET 平台
- .NET Framework（框架）及实现的设计目标
- CLR （Common Language Runtime） 环境
- .NET 编程——公用编程模型

1.1 分布式应用系统及开发技术演进

分布式计算并不是一个新概念。从计算机科学诞生的时候起，无论是通过哑终端访问远程大型主机，还是通过局域网（LAN）中的客户端访问后台服务器，或者是在 Internet 的浏览器上执行 Web 服务器端的程序操作后台数据，都能看到分布式计算的影子。但是，如何创建一个广域分布并全面协作有效的分布式环境，却是在过去的 20 年中 IT 界一直努力发展并取得突破的一项很重要的技术。建立一个有效的分布式环境并不容易，需要解决许多复杂的问题。如在分布式环境中，首先要求端点（即客户和服务器）上的通信基础设施是相对完善的；关于人、机器以及应用的信息如何才能成为可用的；在分布式环境中，关于人、机器以及应用的信息如何才能被合法用户访问到，而不让非法用户访问；分布式环境中的分布式软件系统的各个部分相互之间如何找到对方，它们之间如何通信；如何建立伸缩性强、容易部署的应用，如何建立支持多个并发用户的应用等等。

1.1.1 分布式系统体系结构的发展演进

分布式应用系统体系结构的发展过程可简单分为下面几个阶段：

- 终端/主机（Terminal/Host）

在 20 世纪五、六十年代（甚至 70 年代），一个经典的客户/服务器设置是一台大型超高速服务器（16 位 CPU，主频 6MHz）加上一些哑终端作为客户端。哑终端（Dumb Terminal）访问服务器上的程序和数据。哑终端本身没有太多的工作，它只能通过某种直接的连接，看到大型机上运行的进程；哑终端所做的事就是把字符显示到屏幕上，并把字符回传给服务器。产生这种应用结构的主要原因是当时计算机太少，价格也太昂贵了，有许多企业常常租用专门机构的大型机，为了让更多的员工都能共享一台租用的主机，就产生了多终端远程同时使用大型主机的这种方案。但当时这种多终端远程连接的通信技术非常简单，就是由一条电缆线从终端连入大型主机，所以终端不需要决定自己往哪里发送数据，因为它和主机是通过一条单一的直接链路通信的。这也是称其为哑终端的原因，这种体系结构如图 1-1 所示。

哑终端给人们提供了极大的方便，从某种意义上可以认为用户远程操作终端和直接操作主机是等效的，所以在当时受到了广泛的欢迎并赢得了大量的用户。但与此同时，用户也提出了更高的要求，

这样哑终端便很快发展到智能型终端（Smart Terminal），智能型终端开始包含了专门的电路、固件，还可能有通信协议和母机进行特定的合作。这些智能型终端已经能够承担极其微小的计算工作，可以说它已带来了分布式处理的第一道曙光。

随着 PC（Personal Computer）的出现，仿真终端（Emulation Terminal）也很快出现了，它是在 PC 机上使用简单的终端仿真软件，业务系统仍然完全运行在主机（或 UNIX 服务器）上，这样数百名用户都能远程登录到一台大型的主机（或 UNIX 服务器）上，每个终端用户远程控制业务处理。仿真终端和它们的主机之间的通信要复杂得多，仿真终端和主机之间使用特定的通信协议来完成交互。用户可以在配置终端时选择使用特定的通信协议。

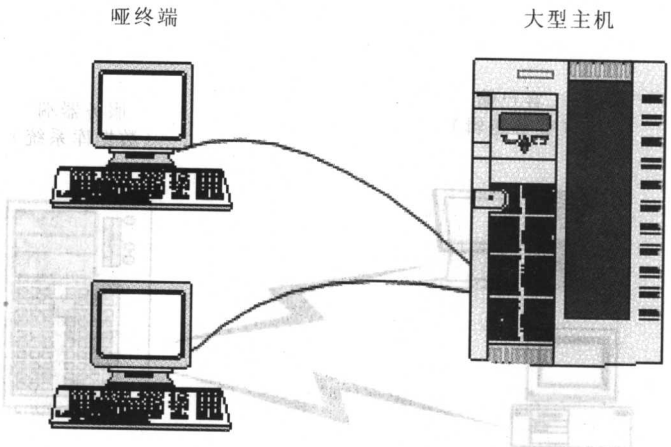


图 1-1　终端/主机结构

- 客户（Client）/服务器（Server）结构

仿真终端的出现已经使终端不再仅仅能处理简单的计算，还要能够处理和控制复杂的业务逻辑过程。这样就给主机系统带来了巨大的压力。另外，随着 PC 机功能越来越强大及其广泛流行，这种远程计算的方法根本没有发挥出 PC 功能的威力，所以，软件开发者意识到这种终端/主机结构的计算环境一方面造成了主机系统运算压力过大，另一方面造成了客户端的资源严重浪费。他们开始考虑把成百上千名用户在服务器上所做的一部分工作放在各自的 PC 终端上来完成。这就产生了具有划时代意义的客户/服务器架构的分布式计算环境。

客户/服务器结构计算环境现在被称为两层（Two Tier）结构的分布式计算环境，其客户和服务器之间使用多种通信协议完成通信；例如可使用 Named Pipe（命名管道）、NetBIOS、Socket（套接字）和 TCP/IP 等协议完成通信。客户端软件和服务器端软件交互则需要靠列集（Marshaling）软件协助来完成，因为程序进程只能理解可认知数据（例如一个有大小和类型的函数参数等等），它无法理解一个原始的网络流。所谓列集（Marshaling）软件就是将可认知数据打包送到原始的网络流，原始的网络流通过网络层传递，收到列集数据的一端需要将原始的网络流数据流解列（Unmarshal）成可认知的格式。

起初在客户/服务器结构的分布式系统中，客户端仍完成辅助处理功能，服务器端则要处理主要的业务逻辑和后台持久性的数据。但随着客户/服务器结构技术的演进，客户端变为“胖”客户端，即客户端可完全处理业务逻辑和大部分计算，服务器端仅处理持久性的数据。这种结构如图 1-2 所示。

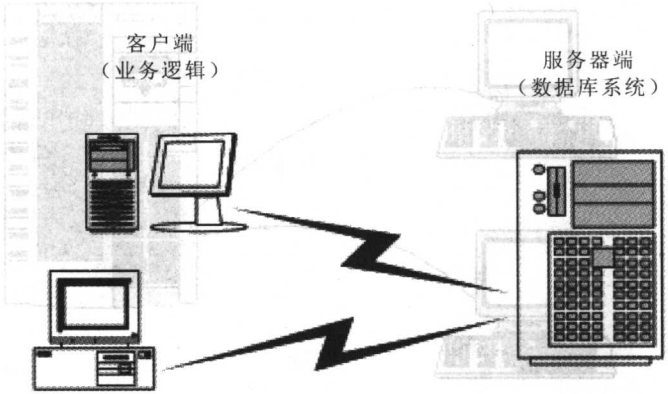


图 1-2 客户/服务器结构

- 基于 RPC 的三层客户/服务器结构

尽管传统客户/服务器结构改变了主机（服务器）系统负担过大而客户端负担过轻的现状，但“胖”客户端的出现却造成客户机不堪重负的事实。随着应用越来越复杂，全部业务逻辑和与之相关的复杂计算在客户机上实现的代价也太昂贵了，同时，其他一些问题也暴露得越来越明显。例如网络负担较重，业务变更困难，客户端部署和维护成本较高，对客户端的安全防护措施要求较高，支持通信的列集层要依靠某个特定的协议，要想使用其他的网络协议，需要开发各自的列集软件等。这些问题丞待