PROGRAMMER TO PROGRAMMER™

Visual Basic .NET Deployment Handbook

Visual Basic .NET 部署手册

Rick Delorme
Billy Hollis 等著
王敏 杜玉红 译

清华大学出版社
http://www.tup.com.cn

Visual Basic .NET 部署手册

Rick Delorme
Billy Hollis 等著

王敏　杜玉红　译

清华大学出版社

北京市版权局著作权合同登记号：01-2002-3195

内 容 简 介

随着网络的发展和.NET 战略的推广，如何高效地部署.NET 应用程序已经成为广大开发人员所要考虑的一个重大主题。本书全面介绍了开发人员在部署.NET 应用程序和组件时所需要了解的所有知识，主要内容包括部署.NET 应用程序的可用策略和选项，高级的 Windows Installer 技术，如何保护和维护应用程序，如何使用全局程序集缓存在应用程序之间共享程序集，并行安装同一个程序集的不同版本，各种可用许可选项，版权保护方面的法律和技术问题等。此外，本书还提供了 3 个附录，可以帮助您更好地理解和掌握本书的主要内容。

本书适合那些精通.NET 应用程序开发方法，并且希望掌握更多的应用程序部署经验的 Visual Basic .NET 开发人员。当然，对于任何需要部署.NET 应用程序的相关人员来说，本书都是非常有价值的参考资料。

图书在版编目(CIP)数据

Visual Basic .NET 部署手册/(美)德罗姆等著；王敏，杜玉红译. —北京：清华大学出版社，2003
书名原文：Visual Basic .NET Deployment Handbook
ISBN 7-302-06239-0

Ⅰ.V… Ⅱ.①德…②王…③杜… Ⅲ.BASIC 语言—程序设计—技术手册 Ⅳ.TP312-62

中国版本图书馆 CIP 数据核字(2003)第 002261 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
责任编辑：于平
封面设计：康博
版式设计：康博
印 刷 者：北京通州大中印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：15.25　**字数**：316 千字
版　　次：2003 年 1 月第 1 版　2003 年 1 月第 1 次印刷
书　　号：ISBN 7-302-06239-0/TP・3734
印　　数：0001～4000
定　　价：32.00 元

前　言

无论您的.NET 应用程序设计和构建得如何成功，如果不将其部署到预定的用户，它将变得毫无用处(除非您计划在自己的开发机器上运行所有程序)。正确部署.NET 应用程序和正确开发这些程序同样重要，并且.NET 为软件开发的部署阶段带来了诸多便利。然而，除了将文件从开发机器 XCOPY 到生产机器以外(本书中将阐述相关内容)，我们要做的事还有很多。高效部署应用程序不仅会使用户受益，也有利于减少对帮助服务台的呼叫次数。

0.1　本书读者对象

Visual Basic .NET Handbook 系列中的所有书籍都面向那些需要学习如何完成特定任务的 Visual Basic .NET 开发人员。本书也是如此，对于那些在应用程序的开发上花费了大量时间和精力、并且还需要部署这些应用程序的开发人员来说，本书将是一个理想的伴侣。这意味着本书实际上适用于所有开发人员，无论是内部 Windows 应用程序开发人员、第三方组件开发人员、独立咨询师，或者是其他需要部署 Visual Basic .NET 应用程序的人员。

要从本书获取最多信息，您将需要一个 Visual Basic .NET 标准版或更高版本的副本，或者 Visual Studio .NET 的任何副本。

0.2　本书主要内容

本书包含了开发人员在部署应用程序时应该了解的所有内容。以下是各章内容的一个概述。

第 1 章——部署策略

本章将论述各种传统的部署方案，并将介绍.NET 给部署领域带来的便利。如果您在过去的几年中从事过 Windows 软件的开发，那么对 DLL Hell 这一术语一定不会感到陌生，本章将说明.NET 的设计怎样克服了这一问题以及其他的问题。

第 2 章——.NET 中的部署选项

本章详细说明了在开发人员决定如何部署应用程序时所能够使用的各种选项。其中

包括了.NET 应用程序在客户机上运行的最低需求，介绍了 Windows 安装程序和其他的部署方法。还简单提及了第三方部署产品。

第 3 章——Windows Installer 的特性

上一章已经介绍了 Windows 安装程序，本章将进行详细阐述。其中说明了怎样使用各种可用的编辑器(如 File System 编辑器)定制安装程序包。本章的结尾处有一个综合的示例，该示例阐明了很多在本章中所述的功能，包括和应用程序一起安装自定义数据库的方法。

第 4 章——配置和保护应用程序

本章论述了配置和保护应用程序时的重要事项。我们从说明所有.NET 应用程序的构件(building clock，即程序集)开始，向您展示如何创建程序集以及如何在建立程序集后查看其中的内容。接下来将介绍如何在应用程序中有效地使用配置文件。本章的后半部分是有关保证应用程序安全的内容，涉及到诸如 Code Access Security(代码访问安全性)和.NET Framework Configuration 工具之类的领域。

第 5 章——维护和更新应用程序

在很多.NET 应用程序中，共享程序集是其重要的组成部分。本章将介绍创建共享程序集的过程，包括怎样创建一个强名。我们在全局程序集缓存(Global Assembly Cache)和适用于整个计算机的数据仓库中查看共享程序集，这些程序集位于所有安装了.NET Framework 的计算机中。本章还介绍了“版本化”这一重要主题，它使我们可以在同一台计算机上安装同一程序集的多个版本，而不会遇到所有和 DLL Hell 相关的问题。

第 6 章——授权

一旦开发出应用程序，您就需要赋予其相应的许可条款，从而确保只有有效用户才能从您的辛勤工作中获益。这里包括了多个可用的许可选项，并显示了它们的使用时间和方法。本章的后半部分阐述了开发人员之间进行许可这一主题，这种授权方式是指，一个开发人员创建了一个组件，并将此组件许可给另一个开发人员在他们的应用程序中使用。

第 7 章——保护您的知识产权

本章讲述怎样保护您的劳动成果。我们给知识产权下了定义，并详述了某些未经同意而从您的应用程序受益的方法。本章还提到了一些用于挫败潜在的窃贼或黑客的常用方法(法律或技术)。

附录 A——使用 Active Directory 部署.NET Framework

任何计算机在运行.NET 应用程序之前都必须先安装.NET Framework。该附录详细介绍了怎样利用 Windows 2000 Active Directory(活动目录)将.NET Framework 部署到位

于某个域中的客户计算机上。

附录 B——随应用程序一起部署.NET Framework

您可能需要随同应用程序的安装程序包给客户提供.NET Framework。该附录展示了将两者组合到一个程序包中的方法。

附录 C——一个自动部署组件

此附录节选自清华大学出版社引进并出版的《Visual Basic .NET 解决方案工具箱》，作者为 Rockford Lhotka。它显示如何在客户机上从 DLL 和 EXE 文件中加载和执行代码，而且不需要手动复制任何文件。

目　　录

第1章 部署策略

软件是在开发机器上开发出来的。然而，为了使软件有用，必须将其驻留在适当的生产机器上，这个机器可以是从 Web 服务器到用户工作站的任何设备。将软件放到需要它的系统中的这一过程我们称之为部署。

1.1 .NET 对部署的影响

.NET 对创建和使用软件的方式存在诸多影响。它明显支持 Internet 技术，这使得那些使用 Web 服务、ASP.NET 和 Web Forms 的新应用体系结构给 Web 站点带来了快速的应用开发。从多语言集成到移动设备支持，再到更简单的智能客户程序开发，使用.NET 所带来的改变没有任何缺陷。

然而，.NET 给部署带来的改变非常重要，.NET 提供了许多部署选项，且它们对基于 COM 的软件是不可用的。这些选项完全改变了部署的经济效果。这些改变是如此重要，它们甚至能改变在.NET 中开发的系统的首选体系结构。

要了解其真实的效果，让我们先来看一看.NET 之前的 Visual Basic 系统中的典型部署方案，并讨论其优劣。

1.2 传统的部署方案

自 Visual Basic 在 1991 年问世以来，那些以 VB 编写的系统所用的体系结构经历了很大的发展。所使用的主要体系结构模式包括：

- Standalone client(独立客户)
- Client-server(客户/服务器)
- N-tier(N 层)
- 带有基于浏览器的界面的 N 层

如果将我们的讨论范围仅仅局限于这个列表，那样就过度简化了，但如果讨论一下.NET 之前的 Visual Basic 系统所面临的常见部署难题，那样就会显得比较充分。所有这些体系结构现今都仍在各种不同的系统上使用，并且将适用于以.NET 编写的特定新系统。

在 NT 体系结构出现之前，Windows 环境都基于 MS-DOS 运行。Windows 9x 之前的应用程序的部署模型反映了 DOS 所允许的部署的简易性。要部署一个应用程序，最低的要求只是创建一个目录并将应用程序的文件复制到其中。

随着 32 位系统的出现，情况就变得更加复杂了。Microsoft 引入了 Component Object Model(组件对象模型，简称 COM)，它最终成为了大多数 Microsoft 的 32 位计算结构的基础。COM 具备一些优点，但其部署却并不简易。

1.2.1 COM 引起的常规部署问题

Microsoft 的 COM 标准在开发时面向那些以有限内存(与今天相比)运行 Microsoft Windows 的小系统。COM 这种设计的折衷面向内存共享和硬件的快捷(今天看来是缓慢的)性能。

这意味着 COM 中的 Dynamic Link Library(动态链接库，简称 DLL)在各个应用程序之间共享，并且为了提供优良性能而使用了一个二进制接口标准。为了让系统能迅速找到运行应用程序所需的组件，DLL 必须在本地 Windows 注册表中注册它们的类 ID。

这一要求意味着即便是很简单地使用 COM 的应用程序也需要复杂的安装程序。这一安装程序必须要识别是否存在必需的 DLL 以及这些 DLL 是否经过注册，如果尚未注册，则需要进行注册。如果 DLL 是与其他程序共享的，则安装程序还必须在适当的位置定位 DLL，它们通常都位于 Windows 主目录的一个子目录下。

除了使 DLL 能够完全工作所需的注册保障外，COM 组件还有另外一个很大的部署问题。它们很容易由于版本化问题而导致无法操作。因为与版本化相关的问题而导致的困境就是通常所说的 DLL Hell。

需要组件进行本地注册还导致了其他限制。它使得我们无法将 COM 应用程序放到 CD-ROM 或网络驱动器上，然后在不执行安装过程的情况下从该位置运行程序。由于必需将组件在本地注册，所以必须执行安装过程。

即使当不再需要某一应用程序时，COM 部署的问题还会出现在系统中。卸载基于 COM 的系统是一个很大的挑战。您必需删除所有相关的注册表项，同时还得小心保留仍为其他系统所需的项。要确定使用某个共享组件的应用程序，确定所有活动应用程序都不再需要某个组件的时间，这些都是很困难的事。这样一来，随着时间的推移，很多系统都在它们的共享组件目录中积起了大量“孤立”的 COM 组件。

COM 部署问题的影响极大。例如，它是打包软件供应商遇到的主要问题的始作俑者，一些供应商的报告说，有半数或更多的技术支持电话都跟 DLL 冲突有关。接下来我们将看到，客户机上 COM 部署的昂贵代价使得用户转而采用基于浏览器的界面，尽管有时候这样的界面对用户并不是最理想的。

由于 COM 无处不在，上面提到的所有主要体系结构模式都在部署时遭遇了这些缺

陷。下面让我们来看一看各个模式，并讨论一下有关部署的问题。

1.2.2 独立客户模式

最早的 VB 应用程序大多为独立的本地应用程序。一些最近的应用程序仍然使用这一模型，通常是因为应用程序需要一个智能用户界面，并且不包含需要和其他用户联机共享的数据。这种应用程序的典型示例包括绘图或制图程序、带有文档版式功能的字处理程序，或者是游戏。

这种应用程序可能会使用一个数据库，但该数据库通常都与应用程序位于同一台计算机上。Access 所依赖的 Microsoft JET Engine 过去是这个体系结构最常用的数据库技术，通常用 Data Access Objects(数据访问对象，简称 DAO)或者 ActiveX Data Objects(ActiveX 数据对象，简称 ADO)进行访问。

这种应用程序有可能无法组件化。这对用 VB3 或更早版本编写的早期应用程序来说无关紧要，因为那时不能用 Visual Basic 构建组件。非组件化的独立应用程序(其中用户界面逻辑、业务逻辑和数据访问逻辑都不加区分地混在一起)通常称为胖客户。在胖客户应用程序中，所有逻辑都位于一个较大的 EXE 文件中，或者可能在若干较小的这种 EXE 文件中。

即使只安装 EXE，胖客户应用程序在用传统的 VB 创建 32 位 Windows 应用程序时也会遇到部署困难。即使应用程序不是由 COM 组件构成的，它仍然必须使用 COM 组件来执行诸如数据访问之类的操作。VB4 到 VB6 的库本身就基于 COM。应用程序可能还会使用其他第三方 DLL，如 ActiveX/OCX 控件。因此，这样的应用程序仍然需要安装技术。然而，与一些更晚的体系结构相比，胖客户的安装还不算复杂。对于内部胖客户应用程序，通常用“Visual Basic 程序包”和“部署向导”来创建安装程序。如果必须将应用程序安装在大量客户机上(例如，一个商业软件程序、或用于整个大公司的应用程序)，则最好用第三方工具构建一个更稳定的安装程序。

这种应用程序的维护需要替换全部受影响的 EXE。即便是修复微小的故障也需要将一个较大的模块分布到各个客户机上。

1.2.3 客户/服务器模式

有时候会将胖客户应用程序扩充到在共享服务器上放置数据。在此使用的多用户数据库的功能比 Jet 引擎更多，它提供了诸如存储过程之类的功能。

当胖客户将某些过程转换到共享的数据库服务器时，它们发展成了客户/服务器应用程序。在传统的 Visual Basic 中，客户/服务器应用程序的部署与胖客户应用程序中的

部署类似。仍然必需在每台客户机上安装一种数据访问技术(通常为 ADO)以便进一步配置，从而可以连接到一个远程数据库。在某些情形下，这些应用程序的部署还包括将数据库元素(起始数据、存储过程等)部署到服务器上这种额外工作。

客户/服务器应用程序带来了一个新的部署问题。如果业务逻辑是在客户代码而不是在存储过程(常见情况)中，那么成千上万台独立的计算机上都可能有相同的业务逻辑。如果要更改业务逻辑，那么就需要全部维护和更新，而在协调方式中常常必需执行这种操作使所有系统都在相同的业务规则下运行。

客户/服务器应用程序可以是组件化的，但很多早期的程序却不是。这就使“复制代码”问题更加复杂了。因为在业务规则更新的时候，它通常是更新整个客户应用程序，以此来确保整个公司都得到了更新。随着这些系统的愈加复杂，以及在 Visual Basic 的较新版本中使用了越来越多的组件技术，组件在客户/服务器应用程序中的使用变得日益广泛。

1.2.4 N 层模式

随着组件化这一趋势的确立，一种新的应用程序设计方法开始投入了使用。由于在客户机和服务器之间添加了一个额外层，所以最初被称为 3-tier(3 层)。这一层在共享服务器上运行，并将业务规则封装于该服务器(通常称为“应用服务器”)上的组件中。后来，这一中间层开始分裂为多个层，从而通常都将它称为 N-tier(N 层)。

N-tier 这种体系结构帮助我们在很大程度上简化了业务逻辑的维护。因为在 N-tier 系统中，业务逻辑大多处于服务器的业务对象中，所以要更改逻辑而不影响客户机就显得轻而易举了。然而，将业务逻辑移动到服务器给部署带来了新的挑战。中间层通常都由多组复杂的内部关联的组件构成。COM 所固有的问题使这些组件的部署和维护变得异常困难。

如果 N-tier 体系结构和基于 COM 的客户代码结合在一起，则会产生最糟糕的 COM 部署问题。很多客户机在安装和维护时都需要在部署上投入大量的金钱和精力，再加上应用服务器上复杂的 COM 组件层，这就产生了最糟糕的 COM 的版本冲突和兼容性问题。

1.2.5 基于浏览器的界面

必须有一种方法来克服昂贵的 N-tier 部署。1996 到 1997 年间，一种替代的方法出现了。Internet 技术开始广泛应用，这就提供了在 Web 浏览器中驻留用户界面的途径，使用这种方法在客户系统上进行部署时基本上不需要任何花费。使用基于浏览器的界

面时，系统还具有可用性广泛的优势，即使用户按照地理位置进行分布也不例外。

新应用程序模型使用的 N-tier 设计和上述的相同，只是完全运行于浏览器中的用户界面层比它更薄。由于客户机的处理性能很容易使用，因此这些系统事实上就是把客户机转换成了旧式 IBM 3270 或 Wyse 终端的可视化优化版本。

基于浏览器的应用程序的部署在中间层和数据层上与上面的 N-tier 方案相同。客户层包含在运行于 Web 服务器上的元素中，而且通常由 Active Server Pages (ASP) 编写。因此，Web 服务器的需要驱使对基于浏览器的系统的客户层进行部署，这些需要例如，创建虚拟目录以供 Web 应用程序使用。

对基于浏览器的应用程序进行基于服务器的部署并不简单。COM 的复杂性使应用服务器的安装和维护变得困难，并且运行于 Web 服务器上的应用程序的层通常会包含带有同样问题的 COM 组件。Web 站点通常包含一些在软件开发组外部开发的静态内容，由于软件更改而进行相应内容的更改使情况变得更加复杂。

然而，至少这些部署问题都还仅限于服务器。一旦 Web 应用程序安装并开始运行后，在客户机上进行部署就不是应用程序的责任了。安装浏览器层这一任务由创建厂商负责，而不是由应用程序负责。

在 Web 应用程序的客户机部署中需要注意两种情形。如果应用程序使用了像插件、applet 或者 ActiveX 控件之类的技术，那么要在部署时包含这些技术的客户机部署。如果 Web 应用程序以特定浏览器或浏览器系列作为目标，那么有必要通过将用户指向一个安装位置的方式来帮助兼容浏览器的安装。

部署到 Web 服务器的复杂情况通常由一连串的位置来处理，这些位置接收应用程序中新的部分或更改过的部分。最常见的安装方式如图 1-1 所示。

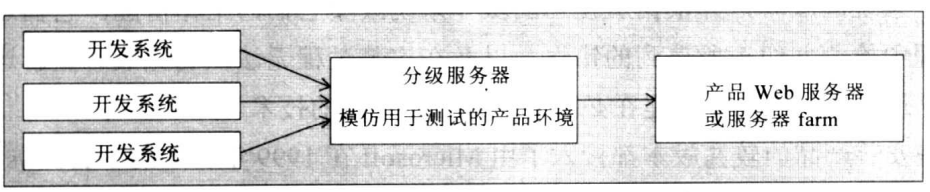

图 1-1

1.2.6 客户部署机制

正如上面所提到的，部署基于浏览器的系统可能会很复杂，但仅仅限于服务器。对那些包含客户端逻辑层的应用程序来说，它们仍然面临着部署到客户机的困难。根据客户机的数量和应用程序的复杂程度的不同，用来解决这种问题的方法也因此而异。

企业内部部署

很多基于 COM 的系统都是为企业内部使用而开发的，只需部署到相对较少的固定

数量的用户。在这种情况下，如果在部署技术上投入大量资金就显得很不经济，因此需要使用简单的安装程序。可能不是每个用户都能够使用该程序，但只要它对于大多数用户都能很好地工作，那其他人就可以通过手工操作完成安装。

VB 打包和部署向导

在为 VB6 系统创建安装程序时，如果系统只是在相对较少的客户机上使用，这时最常用的工具为“VB 打包和部署向导”(有时又称“安装向导”)。以一个 VB6 项目为起始点，向导还会收集系统中包括的一系列组件和其他文件(包括 VB6 运行时文件)。接着，“打包和部署向导”创建一个安装程序，它会复制必要的文件、为组件注册、并创建快捷方式菜单。安装的起始点为一个 Setup.exe 应用程序。它不是一个 Visual Basic 程序，因此可以在安装 VB6 库之前运行。

如果客户机种类很多，它们所使用的操作系统和所安装的其他软件都很不一样，在这种情况下使用“VB 打包和部署向导”很容易遇到 DLL 冲突的问题。然而，典型的企业方案只有少数几个操作系统版本，要担心的产品也只有寥寥几个，因此，只要客户系统不是太大，并且不是太分散的话，克服这些困难还不算是件难事。

现有安装技术

如果要将一个 VB6 应用程序安装到大量的客户机上，则最好不要使用“打包和部署向导”生成的安装程序，而应该采用更好的安装技术。有些供应商给用户提供了专为生成安装程序而开发的产品。最著名的产品是 InstallShield 和 Wise Installer。

这些产品使用户能够编写安装脚本，并对安装过程进行更多控制。例如，它们能够检测操作系统的版本，并根据所使用的操作系统改变它们的工作情况。它们还具有在安装过程中检测并纠正所遇到的错误，以及在安装过程无法顺利完成时重新运行安装的机制。最后，它们自动创建在必要时卸载应用程序的技术。

这些安装产品的较高版本都接入了由 Microsoft 在 1999 年引入的 Windows Installer 技术。Windows Installer 的工作方式是将软件打包成带有.msi 扩展名的文件。在 Windows 2000 或更高的版本中都内置有 Windows Installer 技术，也可以将其改进后用于较早的 32 位 Windows 操作系统。

商业程序包几乎总是使用这些高级安装产品，并且有很多都使用 Windows Installer 技术。计算机和打包软件的安装环境的多样性决定了必须要有高度智能化的安装技术。对基于 COM 的软件来说，花在安装和部署技术上的费用可占到开发成本的 30%，这一事实想必商业程序供应商们都不会感到陌生。