PROGRAMMER TO PROGRAMMER™

Visual Basic .NET Windows Services Handbook

VB.NET Windows 服务开发手册

Richard Conway
Robin Dewson 等著
杨 硕 李 铭 译

清华大学出版社

VB.NET Windows 服务开发手册

Richard Conway 等著
Robin Dewson

杨 硕 李 铭 译

清华大学出版社

北 京

北京市版权局著作权合同登记号：01-2002-3199

内 容 简 介

Windows 服务是可连续运行在 Windows NT 系列操作系统后台的应用程序，也是开发人员必须掌握的重要内容。本书全面介绍了如何利用 Visual Basic .NET 创建 Windows 服务；深入地介绍了在创建 Widnows 服务时用于派生其他类的类；并说明如何构造基于应用程序的 Windows 服务；还描述了如何在大型系统中高效使用 Windows 服务的应用模型。

本书适合那些熟悉.NET 开发环境和 Visual Basic .NET 编程方法、并希望能够利用.NET 语言创建和应用 Windows 服务的各级开发人员。同时也可以作为各大中专院校和培训机构的参考教材。

EISBN：1-86100-772-8
Visual Basic .NET Windows Services Handbook
Richard Conway, Robin Dewson et al

图书在版编目(CIP)数据

VB.NET Windows 服务开发手册/(英)康威等著；杨硕，李铭译.—北京：清华大学出版社，2003
书名原文：Visual Basic .NET Windows Services Handbook
ISBN 7-302-06521-7

Ⅰ.V… Ⅱ.①康…②杨…③李… 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 024757 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：北京市清华园胶印厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：10.75　**字数**：223 千字
版　　次：2003 年 4 月第 1 版　2003 年 4 月第 1 次印刷
书　　号：ISBN 7-302-06521-7/TP・4891
印　　数：0001～4000
定　　价：25.00 元

前　言

Windows 服务是在后台运行的应用程序，它通常没有用户界面。在过去，因为 Visual Basic 不能够创建可靠的、可伸缩的服务应用程序，所以这些应用程序通常是由 C++创建的。.NET 的出现改变了这种状况。虽然我们不能使用所有 C++程序员可以使用的功能，但是.NET Framework 已经为此发布了一个命名空间和一个类的集合，这使得 VB 程序员可以创建作为一种服务运行的托管代码。这种服务可以被添加至服务控制管理器(Service Control Manager，简写 SCM)，也可以使用 Windows 管理工具来查看和配置 Windows 服务。

Windows 服务可以用来创建服务器应用程序，这些应用程序或者响应某些事件，或者不连续地执行一些任务。用户可以创建 Web 服务程序、电子邮件服务程序、记录服务程序、消息服务程序等。读完本书后，您将能够创建功能完备的服务器应用程序，这些程序不需要任何的用户干预。下面将介绍如何使这些服务具有更强的可控制性、可配置性、可伸缩性，以及网络互联特性。

读者对象

本书面向的读者是那些需要创建一项解决特定业务问题的 Windows 服务的专业 Visual Basic .NET 开发人员。阅读本书需要一定的 Windows 服务编程基础知识，但这里假定您没有在.NET 环境或者其他环境中进行服务编程的经验。

假定您已经熟悉 Visual Basic .NET 语言，并且有使用.NET Framework、创建和编译项目，以及在.NET 开发环境下工作的经验。

使用的条件

尽管用户可以打开以及编译用其他版本的 Visual Studio .NET 创建的 Windows 服务项目，但是却不能用 Visual Basic.NET Standard 版本来创建这些项目。在本书的代码下载文件中包含一个空白的 Windows 服务项目。然而，本书建议至少要用 Visual Studio .NET Professional 版本来运行本书中的代码，因为它能提供更好的调试以及开发工具支持，而且仅仅使用 VB.NET Standard 版本，也不能创建所有的示例。

主要内容

本书涉及开发、调试、部署，以及缩放 Windows 服务应用程序的每个阶段。这些内容将在以下各章中介绍。

第 1 章：Windows 服务简介

这一章将提供入门简介、一些历史记录的详细报告，以及在 Visual Basic .NET 中建立一些服务的可能性。服务与标准 VB 应用程序之间的主要区别非常明显，这样就可以知道在服务开发中可以实现的功能。本章主要通过 Visual Studio .NET 提供的向导来介绍创建一种简单服务的过程。在结尾部分，您将能够准确地理解服务的概念以及服务可以实现的功能。

第 2 章：Windows 服务设计

所有的服务都适合某些特定设计模型，了解这些模型有助于更好地应用它们。在 Visual Basic .NET 中，只能遵循对这些模型的选择，因为 VB.NET 不能与低级服务直接交互作用，例如某些设备。您将在本章中学习到 VB.NET 开发人员可以利用的模型、这些模型的优缺点、以及如何针对特定任务选择模型。

第 3 章：编写 Windows 服务代码

本章的主要内容是关于在 Windows 服务里执行的代码和它们的环境交互作用的详细信息。从中将了解到，安全性上下文是如何影响服务的功能、服务是如何通信和记录活动日志的，以及如何安装和调试服务。

第 4 章：Windows 服务的配置和控制

这一章主要介绍如何从外部影响 Windows 服务、如何配置它们，以及在服务运行时如何控制它们。您将认识到服务控制管理器的作用、如何以编程方式访问它，以及如何使用它向服务传递自定义命令；还将介绍如何通过系统托盘上的图标、Web 、MMC 管理单元，以及 Windows Management Interface(Windows 管理接口，简写 WMI)来构建能够控制正在运行的 Windows 服务的接口。

第 5 章：面向网络的服务

Windows 服务通常要求支持某种联网功能，实际上，Windows 服务通常是在一台计算机上提供一个网络服务器的最好方式。这一章将探讨把 Windows 服务部署到远程互联计算机时的一些问题，介绍如何为网络可访问的服务编写代码。

第 6 章：可伸缩性和性能问题

许多 Windows 服务需要同时为多个客户提供长时间运行的服务,这种服务要求在后台执行活动，并且对前台任务造成的干扰应该最小。本章研究如何管理内存和线程，以创建具有可伸缩性，且可以在一定负荷下持久并良好地执行的服务。

第 7 章：部署 Windows 服务

将一项创建好的服务安装在它将要运行的计算机上，并且配置系统从而使这项服务获得运行所需要的所有权限和信息，这是在服务创建中经常被忽视的一个方面。在将一项服务安装到一个实时系统的过程中，以及通过安装脚本自动部署一个服务时会遇到很多问题。本章将对这些问题进行分析。

目　　录

第1章 Windows服务简介

Windows 服务是 NT 服务的下一代(如同 Windows 2000 与 Windows NT 一样)产品；后台任务通常是在启动时由内核加载，它们不依赖于任何用户登录，并且由 Service Control Manager(服务控制管理器，简写 SCM)控制着它们的生存期，您今后将了解到更多信息。下面请立即找到服务管理工具，并查看在您的操作系统上已经启动的服务。可以发现它们的数目相当多。

Windows 操作系统下的服务有以下三条通用特征：

- 它们缺少一个用户界面
- 它们能在安全上下文中运行，不需要进行用户登录
- 它们可以在没有用户交互的情况下启动

用 VB 编写一项服务像学习 C++一样复杂，而用 C++编写这项服务似乎是个更好的选择。本书的重要之处正在于此。由于.NET Framework 的推出，现在用 VB.NET 创建一项 Windows 服务正像用 C#或者 C++创建它们一样简单。

本章是介绍性章节，涉及的主题包括：

- 服务的简史
- 构建简单 Windows 服务的方式
- 使 VB.NET 服务编程得以实现的 System.ServiceProcess 命名空间

假如您以前有过 VB 编程的经验，在学完本章后，您将由衷地赞叹建立一项基本服务是如此简单。假如您刚刚接触服务编程，您可能会奇怪这有什么难的。如果是这种情况，那将意味着.NET Framework 设计师们的工作做得很出色。

1.1 服务的简史

这不是一本历史书，但是理解服务的由来有助于认识它们的发展方向。服务作为一个概念很多年以来就是大部分操作系统上的计算的支柱之一，并且短期内它们所服务的功能不会因为.NET 的出现而变得多余。事实上恰恰相反，由于增强了可用性以及更加便于创建，它们有可能在日常的企业应用程序中得到更加广泛的应用。

1.1.1 UNIX 连接

Windows 现在所使用的许多事件驱动和时间驱动的服务都基于一些最初为 UNIX

编写的进程。在 UNIX 平台上，这些进程分别指的是众所周知的 daemon(端口监控程序)和 cron(时钟守护程序)作业。

1. daemon

daemon 不是“魔鬼”。无论怎样，UNIX 都没有任何邪恶之处。在古代英语中，daemon 意味着“被神化的人”，它最初被用来描述一个人性格的某些方面，正像在糟糕的电视场面中，谚语中的天使和恶魔同时出现在面临作出困难决定的人的肩上。UNIX 端口监控程序的含义是指它们一直潜伏在后台，直到被激活以帮助应用程序实现一些功能。

这样，daemon 静待事物的发展：事实上，至今它们仍然存在于 UNIX 和类似于 UNIX 的平台上。像 Apache、FTP 客户端以及邮件传输代理这样的 Web 服务器都是 daemon 的示例。它们作用于通过以下方式到达的信息——通过网络端口到达的信息，或者直接到达目录的信息，或者通过用户界面到达的信息。

随后，因为 daemon 被转换成了能在 DOS 操作系统下运行的版本，这些新的派生物变成“终止并驻留(terminate and stay resident，简写 TSR)”的应用程序，这个术语至今仍在使用。

2. cron

Unix 服务的另一个方面就是 cron。它的目的是基于时间(cron——即 chronological)启动和终止活动。服务器管理员编辑 crontab 文件，这种文件包括要实现的任务和要实现的时间，cron 在文件中保留制表符，等待着下一点来触发操作。与此同时，它占用内存，在后台驻留。

daemon 和 cron 都是服务类型的示例，这些示例能够通过使用 Windows 服务来实现，并且它们能够反映过去所使用的主题。像 NT 世界悄悄进入商业领域一样，系统需要类似的功能性，于是 Windows Scheduler(Windows 调度程序)首次被添加到了系统中。

1.1.2 调度程序和任务管理器

跳过前面提到的 DOS 和 TSR 应用程序，Windows 的早期版本使用服务(自发布之日起)来执行事件驱动活动，并使用 Windows Scheduler 来跟踪基于时间的作业。如同 cron 一样，它也驻留在内存中以跟踪时间，但它不同于文件，它在调度程序注册表数据库中保持它必须启动或终止的作业。管理员有权使用 AT 命令对它进行更改。

早期的 Windows 也引入了 SysTray，它是一个类似于服务的程序，为基于界面(基于窗体和基于 Windows)的程序提供像服务一样运行的机会。在现今的 Windows 中它仍然存在(通常是在屏幕的右下角有一个时钟和其他图标)，它赋予有用户界面的应用程序

同等机会，并且使用户有机会访问像声音控制、网络和病毒扫描这样的服务。

虽然它工作得很出色，Microsoft 在 1997 年发布 IE 4.01 更新版时，还是用 Task Manager(任务管理器)代替了调度程序——一个 SysTray 应用程序。尽管这些术语看上去没有什么联系，但是经证实 Task Manager 比调度程序更加稳定，而且至今它仍然是 Windows 的组成部分。

1.1.3 NT 服务

今天的 Windows 服务和迄今为止已经提及的服务之间的最后的连接纽带就是 NT 服务(它们伴随着 Windows NT 的发布而产生)。它们在设计上与当今的服务最接近，原则上它们是相同的。更确切地说，它们没有用户界面，在指定用户的上下文中运行，并且在非作业期间也驻留在内存中。

传统上，NT 服务与操作系统或服务器的硬件紧密相连，通常执行以下任务之一：

- 硬件监控器和驱动程序
- DCOM 组件
- 通信服务
- 数据库管理服务
- Internet Protocol 管理器

一项服务要在低层工作，它需要能够访问许多 Windows API 来实现其功能。因此，通常需要用 C、C++或 Delphi 这样能够给开发人员提供访问权限的语言来编写这些服务。

另一方面，Visual Basic 的设计师设计它的目的是用于编写窗体应用程序，因而完全没有考虑低层 API 调用的问题。随着有经验的编程人员的数量日益增多，任务变得越来越复杂，VB 的高级功能逐渐被开发出来。然而，这些功能没有经过很好地测试，甚至在语言本身内部实现都无尚佳表现。对于那些有空闲的开发机器和一本 Dan Appleman 编写的《Windows API bible for VB programmers》的编程人员来说，不久就会深刻地认识到还有许多功能他们不能实现——加密、描绘和创建服务只是其中的三个。

幸运的是，C++和 VB 编程人员之间的不平衡已经得到改善。有了.NET 中的 Common Language Runtime(公共语言运行库)以后，VB 编程人员可以编出与 C#编程人员相媲美的 API 代码。这就意味着能力。与以往相比，VB 已经显示出其更易于学习，编码更迅速的特性。现在它与 C 类型语言拥有相同的功能。

1.2 MyFirstService

下面并没有马上建立基本服务，而是采用更好的开始方式。我们将先为服务生成和

安装可执行文件，其主要功能是将时间写进事件日志里。这诚然不是十分有用，但是它展示了服务的许多主要部分，包括启动和停止方法、逻辑和安装程序。它还将展示建立和安装一项服务所需要的基本步骤——写代码、编写和运行安装程序，以及接下来的启动和使用(或消费)服务。

(1) 打开 Visual Studio .NET，并单击起始页上的 New Project 按钮。

(2) 在 New Project 对话框里，创建一个新的 Visual Basic Windows Service 项目。

(3) 将新项目名称改为 MyFirstService，如图 1-1 所示。

(4) 确信您的 Location 位于本地驱动器上，否则在调试时将发生违反安全的错误。

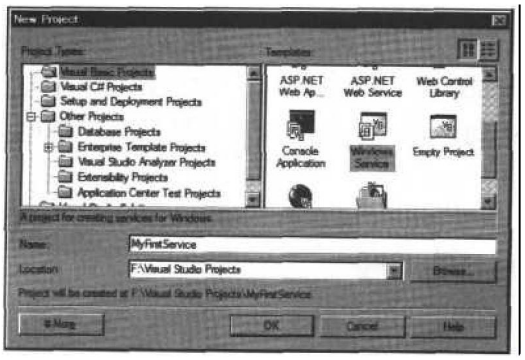

图 1-1

(5) 按 OK 按钮。Visual Studio .NET 将创建一个包含合适文件的新项目，并提供一个设计器屏幕，如图 1-2 所示。

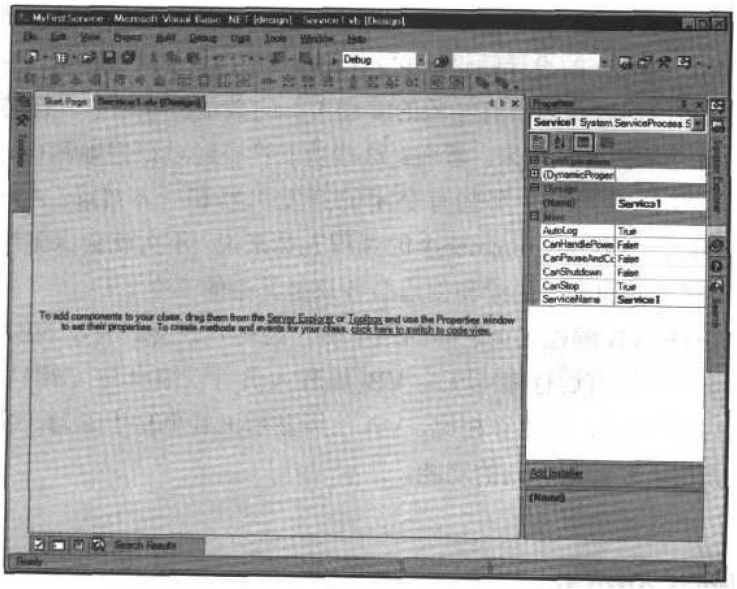

图 1-2

1.2.1　使用设计器或独立编写代码

在这一点上，我们可以选择以可视化方式或者自己编写所有代码的方式来开发服务。但假如默认的服务没有可视元素的话，也许会有些出人意料。如果您选择通过可视化方式来设计基于窗体的应用程序，但可能并没有想到也可以使用这种方式来设计服务。

实际上，当您思索 Visual Studio .NET 的设计工作原理时，也许它不像您设想的那样毫无意义。

1. 设计器

Visual Studio .NET 是一种高效的应用程序开发工具，它的确名副其实。工具箱包含许多非可视组件，这些组件可以被添加到位于服务的核心的逻辑和 WinForms 控件中，其中 WinForms 控件用于建立基于窗体的应用程序的用户界面。像控件一样，我们所需要做的工作仅仅是把需要的组件从工具箱拖动到设计器上，其代码会自动生成。

当然，我们也可以独立编写代码——但是它可能会和使用可视化设计器生成的代码完全一致。如何创建组件将根据您的选择而定，但是接下来我们将从可视化工具开始。

(6) 找到工具箱，切换到 Components 区域。

(7) 从工具箱拖出一个 EventLog 对象和一个 Timer 对象放在设计器区域，如图 1-3 所示。

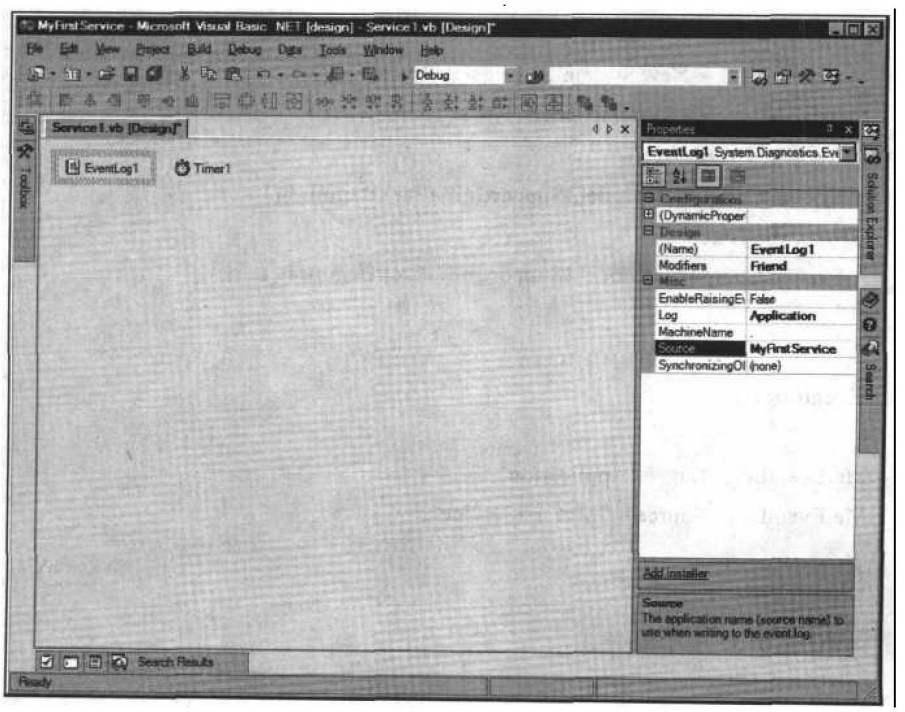

图　1-3

像 VB6 中的 Timer，每个非可视元素都有它的可视化表示，不仅如此，更重要的是，还有一个属性面板与之相匹配。那些用来创建和初始化每个组件的代码也已经生成。不能仅仅因为此前从未在创建服务时用过这个可视 RAD 方法而放弃该方法。它是在您不得不亲自动手以前生成启动代码的一种快速可靠的方法。

(8) 选择设计器上的 EventLog 对象，并在应用程序的属性面板中修改日志属性。

(9) 仍然是 EventLog 对象，将它的 Source 属性修改为 MyFirstService。

随后，我们将在事件查看器中查找这些名称，以便证实服务正在运行。

2. 代码视图

下面让我们来查看一下 VS .NET 已经自动生成的代码。

(10) 在设计器窗口右击，并在弹出的上下文菜单中选择 View Code 命令。

您可以看到，VS.NET 已经生成了相当数量的样板代码，而如果您选择独立编写服务代码，将不得不自己编码，由此可见，应用设计器确实是一个相当不错的主意。如果对此比较感兴趣，当您展开组件设计器生成的代码段时，可以在 InitializeComponent() 方法中发现创建 EventLog 和 Timer 的代码。

```
Friend WithEvents EventLog1 As System.Diagnostics.EventLog
Friend WithEvents Timer1 As System.Timers.Timer
<System.Diagnostics.DebuggerStepThrough()>
  Private Sub InitializeComponent()

  Me.EventLog1 = New System.Diagnostics.EventLog()
  Me.Timer1 = New System.Timers.Timer()
  CType(Me.EventLog1,
    System.ComponentModel.ISupportInitialize).BeginInit()
  CType(Me.Timer1,
    System.ComponentModel.ISupportInitialize).BeginInit()

  '
  'EventLog1
  '
  Me.EventLog1.Log = "Application"
  Me.EventLog1.Source = "MyFirstService"

  '
  'Timer1
  '
  Me.Timer1.Enabled = True
```

```
    '
    'Service1
    '
    Me.ServiceName = "Service1"
    CType(Me.EventLog1,
      System.ComponentModel.ISupportInitialize).EndInit()
    CType(Me.Timer1, System.ComponentModel.ISupportInitialize).EndInit()

End Sub
```

注意直接位于 InitializeComponent()上方的警告，并且将这段代码留给 VS.NET。把我们自己编写的代码添加到文件的其余部分，随后如果想要使服务起作用，还需要给它一些指导。

1.2.2　添加一些功能

明确我们第一项服务的目标，是在一个给定的时间段之后将时间写入事件日志。我们设置计时器在服务启动时运行，在服务停止时结束。在这两个事件之间，服务所需要采取的惟一操作就是当一个给定的时间段过去后，将时间写入事件日志。我们可以通过编写一个 Timer.Elapsed 事件处理程序来实现。

服务可能总需要在启动时进行某些初始化以及在停止时进行清除，从而使 VS.NET 在默认情况下能够为这些事件自动生成空白处理程序。下面我们将对启动和停止服务的事件处理程序填充代码。

(11) 为使计时器有效，将下列代码添加到 OnStart 方法。

```
Protected Overrides Sub OnStart(ByVal args() As String)
    'Set timer interval to 6 seconds
    Timer1.Interval = 6000

    'Enable the timer. This will throw the Elapsed event every Interval
    Timer1.Enabled = True
End Sub
```

(12) 在将代码添加到服务的 OnStop()处理程序之前，问自己几个简单的问题。“如何撤销在 OnStart()方法里的操作？用什么方法可以将这项服务完全清除？”当您找到答案，您就应该添加代码到 OnStop()方法中。在这项服务的情况下，答案非常简单。

```
Protected Overrides Sub OnStop()
```

```
    'Disable the timer
    Timer1.Enabled = False
End Sub
```

(13) 最后，但也是相当重要的，为 Timer1.Elapsed 事件处理程序添加下列代码，其作用是将时间写入事件日志。

```
Private Sub Timer1_Elapsed(ByVal sender As System.Object, _
  ByVal e As System.Timers.ElapsedEventArgs) Handles Timer1.Elapsed

  'Write the time to the event log.
  EventLog1.WriteEntry(DateTime.Now.ToShortTimeString().ToString())

End Sub
```

上面已经给出演示所需要的全部代码。现在需要做的就是编译和运行这些代码。

(14) 建立解决方案并检查错误(Ctrl + Shift + B)。

(15) 单击 Debug/Start 菜单命令(F5)。

遗憾的是，这还没有完成全部的工作，正如在按 F5 后很快出现的错误消息(如图 1-4 所示)所提示的那样。

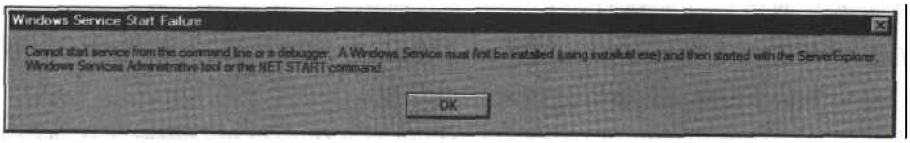

图 1-4

1.2.3 为服务创建一个安装程序

按下 F5 弹出的消息表明服务不能从 Visual Studio 环境运行，因为它们需要服务控制管理器(Service Control Manager，简称 SCM)才能正常运行，而且为了使 SCM 运行这项服务，必须有它的某些元信息和它需要的资源。

幸运的是，Visual Studio .NET 不仅包括对提供安装工具的预建类的引用，而且包括简化这些类的配置的向导。构建一项服务的安装程序比构建服务本身要简单。

(1) 切换到 Service1.vb 的设计器窗口，并且，为了不选中任何组件，在设计器窗口上单击。

(2) 按下 F4 查看 Properties 面板。它将显示服务的属性，如图 1-5 所示。