单击 OK 按钮。

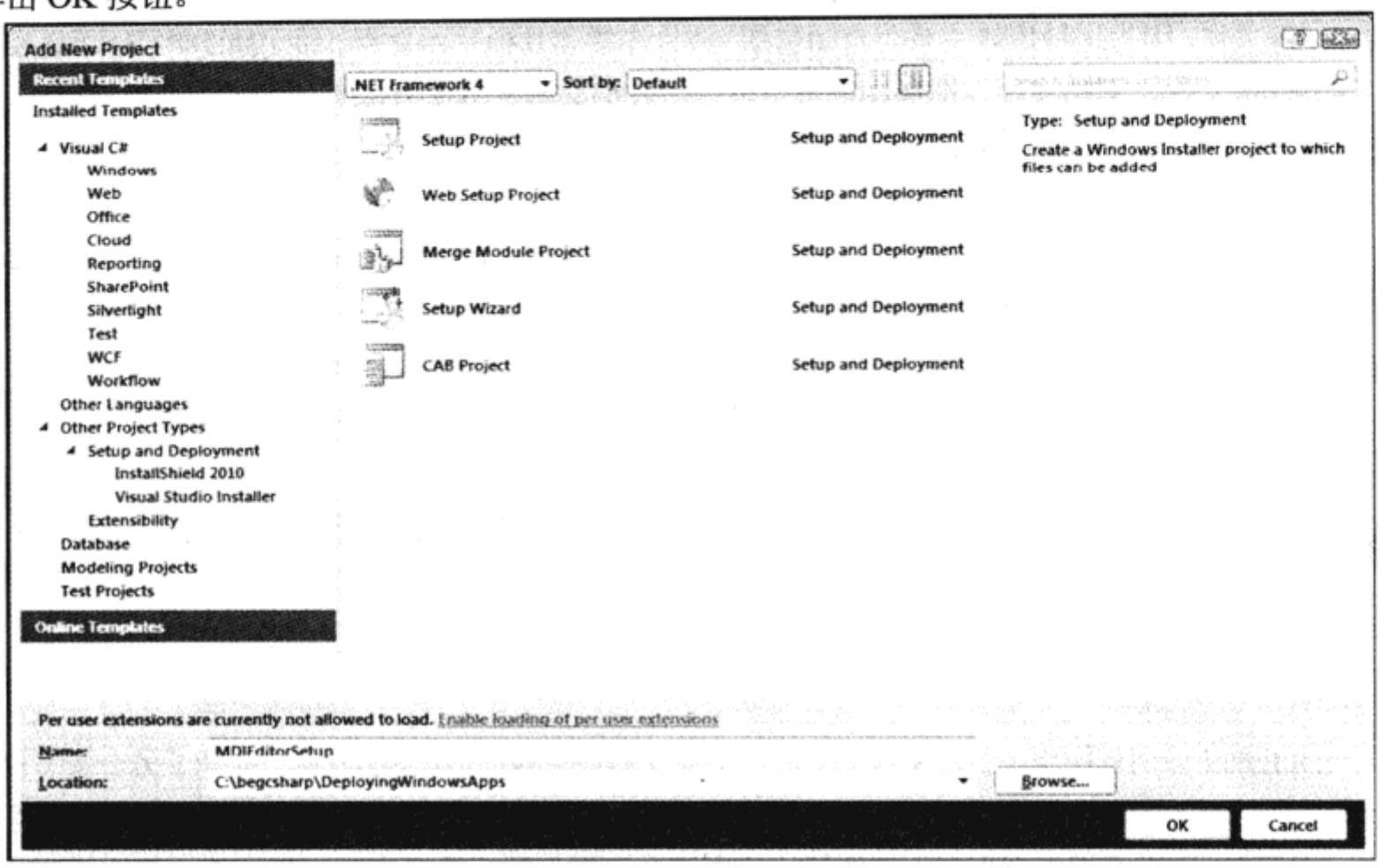

图 17-22

17.5.3 项目属性

到目前为止，我们仅仅拥有一个项目文件可以用于安装解决方案。必须定义要安装的文件，还必须配置项目属性。为此，必须了解 Packaging 和 Bootstrapper 选项的含义。

1. 打包

MSI 是启动安装的数据库，但是我们可以定义如何使用如图 17-23 所示的 3 个选项打包要安装的文件。右击 MDIEditorSetup 项目，选择 Properties，就会打开此对话框。

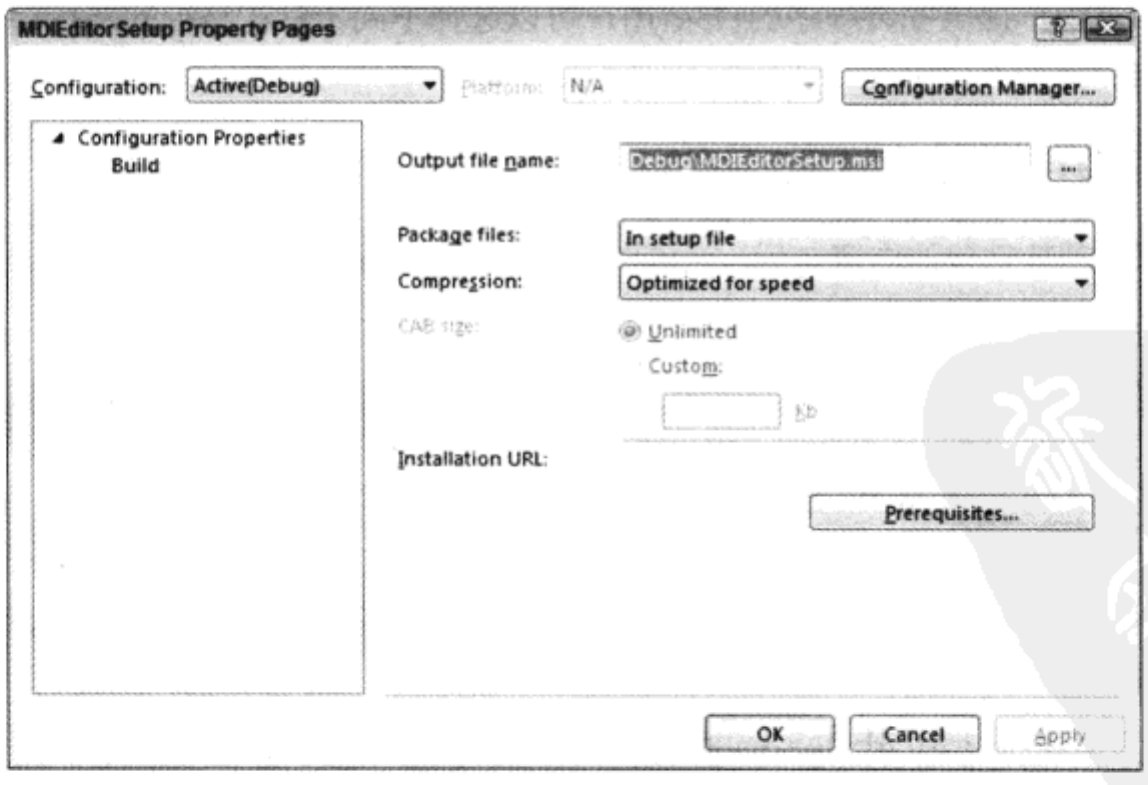

图 17-23

Package files 下拉列表中的选项如下所示：

- As loose uncompressed files 选项将所有程序和数据文件都原样存储，不进行文件压缩。
- In setup file 选项会把所有的文件合并、压缩到 MSI 文件中。可以为软件包中的单一组件重写此选项。如果将所有文件放到一个 MSI 文件中，就必须注意安装程序的大小要适合于希望使用的介质，比如 CD 或软盘。如果安装的文件太多，超过了一张软盘的容量，可以试试在 Compression 下拉列表中选择 Optimized for size 选项，改变压缩选项。如果其容量仍然不适合，则可以选择下一个打包选项。
- 对文件打包的第三种方法是 In cabinet file(s)。在此方法中，MSI 文件仅用于加载和安装 CAB 文件。使用 CAB 文件可以设置文件的大小，以便从 CD 或软盘上安装(对于从软盘的安装，可以设置 1440KB 的安装容量)。

2. 预先安装的软件包

在上面的对话框中可以配置预先安装的软件包，即在安装应用程序之前必须安装的部分。单击 Installation URL 文本框旁边的 Prerequisites 按钮，就会弹出 Prerequisites 对话框，如图 17-24 所示。可以看出，.NET Framework 4 Client Profile 默认选中，作为预先安装的软件包。如果客户系统没有安装.NET Framework，就从安装程序中安装它。还可以选择其他预先安装选项，如下所示。

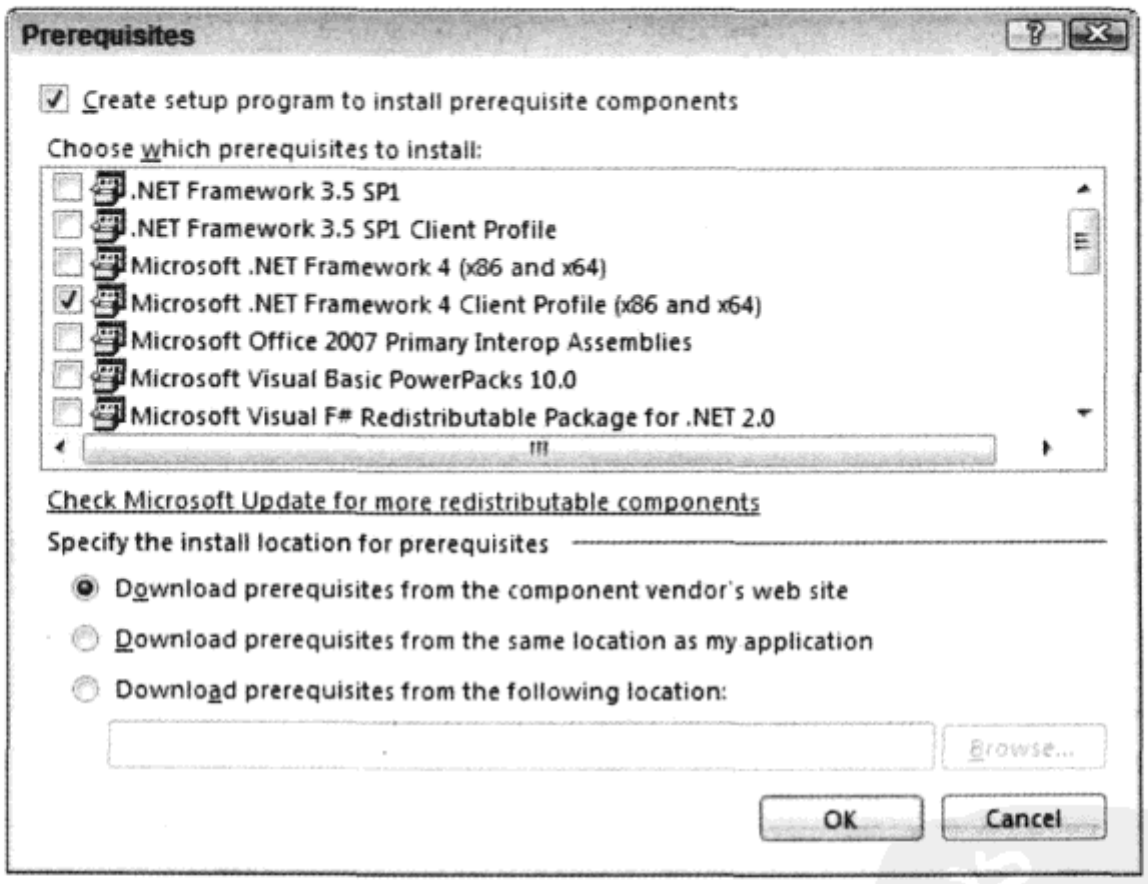

图　17-24

- Windows Installer 3.1：Windows Installer 3.1 用于通过 Visual Studio 2010 创建的安装软件包。如果目标系统是 Windows Vista 或 Windows Server 2008，该安装程序就已经安装在系统上了。在较旧的系统上，可能没有 Windows Installer 的正确版本，所以应选择这个选项，在安装程序中包含 Windows Installer 3.1。Windows 7 使用 Windows Installer 5 版本，而 Visual Studio 2010 使用 Windows Installer 4.5 版本。

- **SQL Server 2008 Express**：如果需要在客户系统上有一个数据库，就可以在安装程序中包含 SQL Server 2008 Express 版本。有关用 ADO.NET 访问 SQL Server 的详细内容见第 24 章。
- **Microsoft Office 2007 Primary Interop Assemblies**：对于使用 Office 自动功能的应用程序，可以用这个组件安装 Office 2007 的主要互操作程序集。
- **Visual Basic PowerPacks 10.0**：提供了用 Visual Basic 编程的额外特性，Visual Basic 可以用这个组件来安装。
- **Visual F# Redistributable Package**：对于用编程语言 F#编写的应用程序，需要这个软件包。

试一试：配置项目

(1) 将 Property 页面上的 Prerequisites 选项(View | Property Pages)改为包含 Windows Installer 3.1，这样应用程序就可以安装在没有 Windows Installer 3.1 的系统上。将输出文件名改为 WroxMDIEditor.msi，如图 17-25 所示。然后单击 OK 按钮。

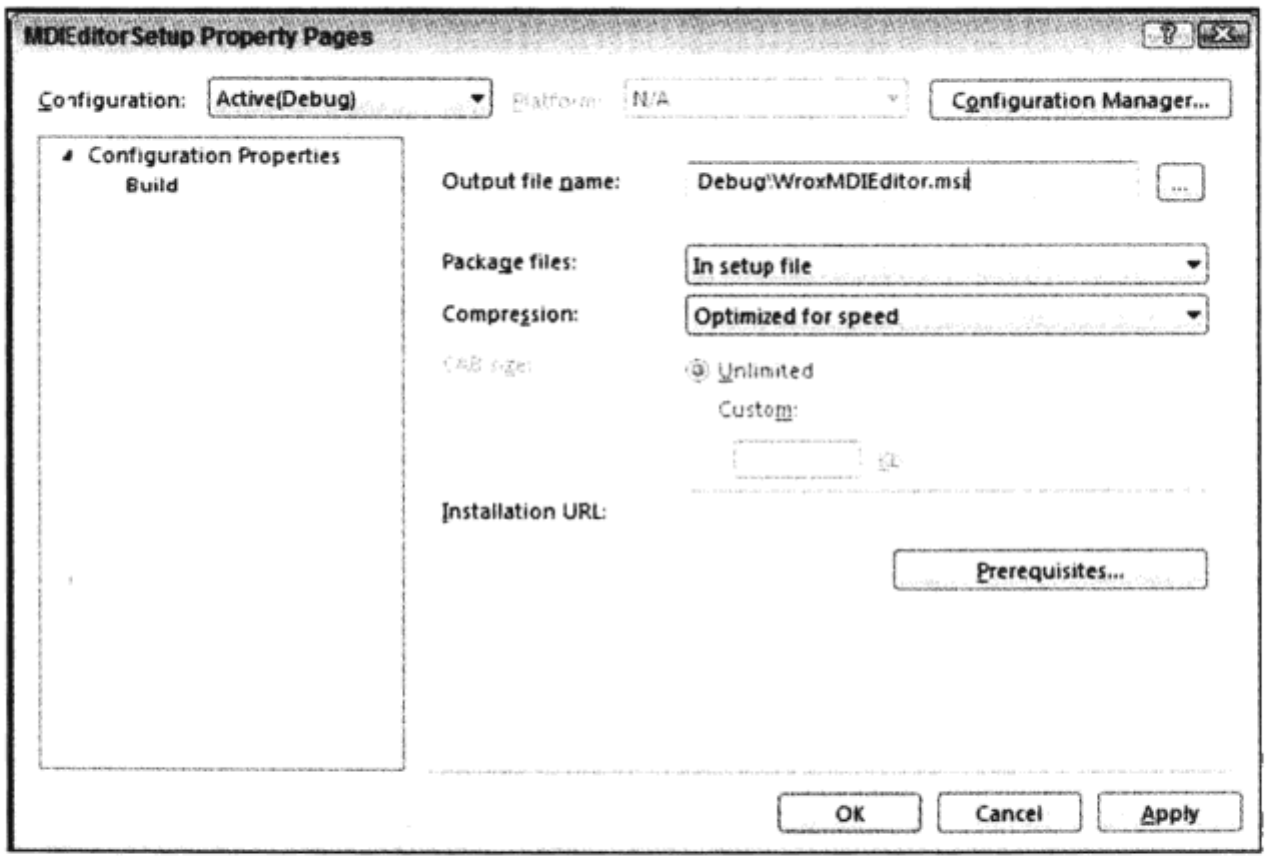

图　17-25

(2) 使用 Properties 窗口将项目属性设置为表 17-1 中的值。

表　17-1

属　性	值
Author	Wrox Press
Description	要打印和编辑文本文件的 MDI Editor
Keywords	Installer、Wrox Press、MDI Editor
InstallAllUsers	True
Manufacturer	Wrox Press
ManufacturerUrl	http://www.wrox.com

(续表)

属　性	值
Product Name	Wrox MDI Editor
SupportUrl	http://p2p.wrox.com
Title	MDI Editor 的安装演示
Version	1.0.0

17.5.4　安装编辑器

Visual Studio 2010 的 Setup Project 有 6 个编辑器。选择编辑器的方式如下：打开部署项目，选择 View | Editor 菜单项。

- File System 编辑器用于向安装软件包中添加文件。
- 使用 Registry 编辑器，可以为应用程序创建注册表项。
- File Types 编辑器允许注册应用程序的特定文件扩展名。
- 使用 User Interface 编辑器，可以添加和配置在安装程序期间显示的对话框。
- Custom Actions 编辑器允许在安装和卸载期间启动定制的程序。
- 使用 Launch Conditions 编辑器，可以指定对应用程序的要求，比如，必须已安装NET 运行库。

17.5.5　File System 编辑器

使用 File System 编辑器，可以向安装软件包中添加文件，并配置安装它们的位置。通过 View | Editor | File System 菜单可以打开此编辑器。还会自动打开一些预定义的特殊文件夹，如图 17-26 所示。

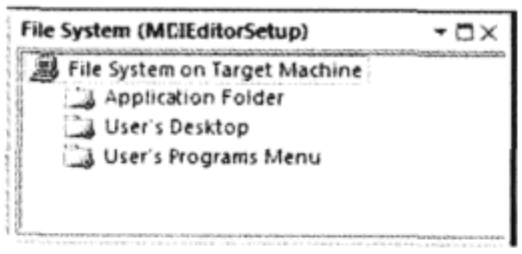

图　17-26

- Application Folder 文件夹用于存储可执行文件和库。其位置定义为[ProgramFilesFolder]\[Manufacturer]\[ProductName]。在 English 语言系统上，[ProgramFilesFolder]解析为 C:\Program Files。[Manufacturer]和[ProductName]的目录由 Manufacturer 和 ProductName 项目属性定义。
- 如果希望在桌面上放置图标，可以使用 User's Desktop 文件夹。此文件夹的默认路径是 C:\Users\username\Desktop 或 C:\Users\All Users\Desktop，这取决于安装是用于单个用户还是所有用户。
- 用户通常在 All Programs 菜单中启动程序。默认路径是 C:\Documents and Settings*username*\Start Menu\Programs。在此菜单中可以把快捷方式放到应用程序上。快捷方式的名称包括公司和应用程序的名称，这样用户就很容易识别应用程序，如 Microsoft Excel。

一些应用程序创建了子菜单，由此可以启动多个应用程序，如 Microsoft Visual Studio 2010。根据 Windows 准则，许多程序由于错误的原因而列举了大量不需要的程序：在此菜单中不应该放置卸载程序，因为此功能可以在 Control Panel 的 Programs and Features 中获取，卸载程序应该在控制面

板中进行。此菜单中也不应该放置帮助文件，因为帮助文件可以直接从应用程序中获取。因此对于许多应用程序而言，在 All Programs 菜单中直接放置应用程序的快捷方式就足矣。这些限制的目的是不要在 Start 菜单中放置太多的项，避免使它过于拥挤。

此信息的参考信息位于“Microsoft Windows 7 桌面应用程序规范”中。该文档在 http://msdn.microsoft.com/en-us/windows/dd203105.aspx 上。

右击并选择 Add Special Folder，还可以添加其他文件夹，其中一些文件夹如下：

- Global Assembly Cache (GAC)Folder 表示安装共享程序集的文件夹。GAC 用于在多个应用程序之间共享的程序集。
- User's Personal Data Folder 是用户的默认文件夹，其中存储了文档。C:\Users\[username] \My Documents 是默认路径。此路径是 Visual Studio 用于存储项目的默认目录。
- 当选择文件时，User's Send To Menu 中的快捷方式可以扩展 Send To 关联菜单。使用此关联菜单，用户通常可以将文件发送到目标位置，如软驱、邮件接收者或 My Documents 文件夹。

1. 向特殊文件夹添加项

选择文件夹，再选择菜单 Action | Add Special Folder，可以从列表中选择一些项，添加到特定的文件夹中。可以选择 Project Output、Folder、File 或 Assembly。给文件夹添加项目的输出，会自动添加生成的输出文件，以及一个.dll 或.exe，这取决于所添加的项目是组件库还是应用程序。选择 Project Output 或 Assembly 可以给文件夹自动添加所有的依赖关系(所有引用的程序集)。

2. 文件属性

在文件夹中选择文件的属性，就可以设置表 17-2 中所示的属性。根据文件类型，可能无法应用其中一些属性，也可能应用其中未列出的一些属性。

表 17-2

属 性	说 明
Condition	可以使用此属性定义一个条件，确定是否选择应安装的文件。如果仅在特定的操作系统版本上添加该文件，或者用户必须在对话框中进行一些选择，该属性就非常有用
Exclude	如果不安装此文件，则该属性设置为 True。这样文件就可以保留在项目中，而不会安装。如果可以肯定它不是相关文件，或它已存在于要部署应用程序的每个系统中，则可以排除该文件
PackageAs	利用 PackageAs 可以重写文件添加到安装软件包的默认方式；例如，如果项目配置是 in setup file，就可以使用此选项将特定文件的软件包配置改为 Loose，这样此文件就不会添加到 MSI 数据库文件中。如果希望添加用户在开始安装之前读取的 ReadMe 文件，这就非常有用。显然，即使压缩了其他文件，也不应压缩此文件
Permanent	将此属性设置为 True 意味着，在卸载之后文件仍会保留在目标计算机上。这可以用于配置文件。例如，安装 Microsoft Outlook 的新版本时，如果配置了 Microsoft Outlook，则在卸载并再次安装它时，就不需要再次配置它，因为上一次的安装配置并未删除
ReadOnly	此属性在安装时设置只读文件特性
Vital	此属性表示，此文件对于此产品的安装至关重要。如果此文件的安装失败，就终止完整的安装进程，只能回滚

下面的示例给 Windows 安装程序包添加应部署的文件。

试一试：向安装软件包添加文件

(1) 向安装程序项目的 Application 文件夹添加 MDI Editor 项目的主要输出，为此使用 Project | Add | Project Output 菜单项。在 Add Project Output Group 对话框中，选择 Primary Output，如图 17-27 所示。

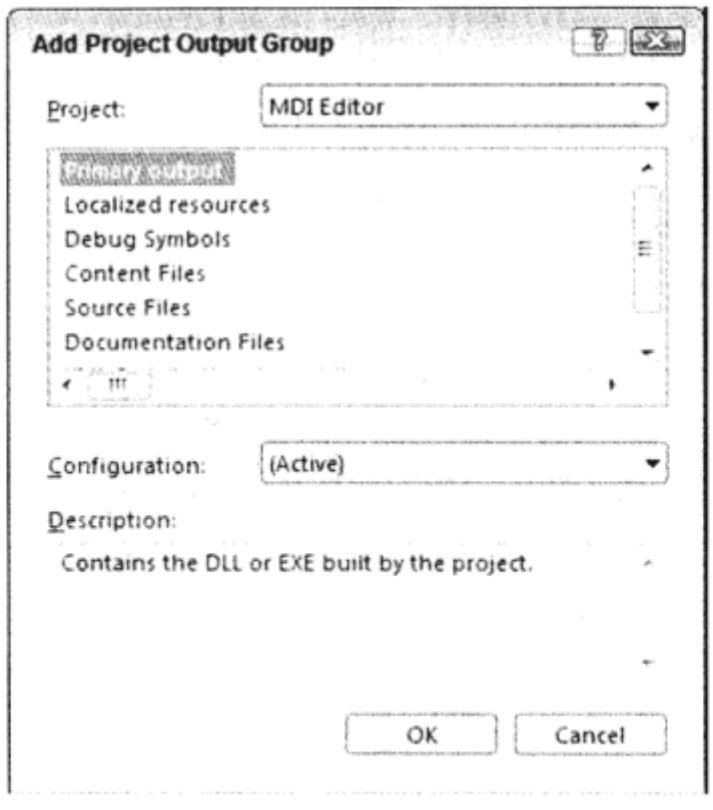

图　17-27

单击 OK 按钮，把 MDI Editor 项目的主要输出结果添加到自动打开的 File System Editor 中的 Application Folder。在此，主要输出就是 MDI Editor.exe。

(2) 要添加的附加文件有徽标、许可和 ReadMe 文件。在 File System Editor 中，选择 Application Folder，并选择菜单项 Action | Add | Folder，在 Application Folder 中创建名为 Setup 的子目录。

在 Visual Studio 中，只有选择了 setup editor 中的项，才能使用 Action 菜单。如果选择了 Solution Explorer 或 Class View 中的项，就不能使用 Action 菜单。

(3) 右击 Setup 文件夹，选择 Add | File，将文件 wroxlogo.bmp、wroxsetuplogo.bmp、readme.rtf 和 license.rtf 添加到 Setup 文件夹中。这些文件可以从本书的下载代码中得到，也可以自己创建这些文件。可以使用许可和 ReadMe 信息填充这些文本文件。无需改变这些文件的属性。这些文件会用在安装程序的对话框中。

位图 wroxsetuplogo.bmp 的大小应是 500 像素宽，70 像素高。该位图左边的 420 像素应只是一个背景图形，因为安装对话框的文本会覆盖这个区域。

(4) 将文件 readme.txt 添加到 Application Folder 中。在开始安装之前，用户应能读取这个文件。将属性 PackageAs 设置为 vsdpaLoose，这样此文件就不会压缩到 Installer 软件包中。还要将属性 ReadOnly 设置为 True，这样就不会改变文件。

项目现在包含两个 ReadMe 文件：readme.txt 和 readme.rtf。文件 readme.txt 可以由安装应用程序的用户在安装开始之前读取，文件 readme.rtf 用于在安装对话框中显示一些信息。

(5) 将文件 demo.txt 拖放到 User's Desktop 文件夹中。只有在询问过用户是否希望安装之后，才能安装此文件。因此，将此文件的 Condition 属性设置为 CHECKBOXDEMO。CHECKBOXDEMO 是用户可以设置的条件，其值必须大写。此条件只有设置为 TRUE，才能安装此文件。稍后将定义一个对话框，来设置此属性。

(6) 要从 Start | Programs 菜单中启动程序，需要 MDI Editor 程序的一个快捷方式。

在 Application Folder 中选择 Primary output from MDI Editor 项，并打开菜单项 Action | Create Shortcut to Primary output from MDIEditor。将所生成的快捷方式的 Name 属性设置为 Wrox MDI Editor，并将此快捷方式拖放到 User's Programs 菜单中。

17.5.6 File Types 编辑器

如果应用程序使用了定制的文件类型，并注册文件扩展名，希望用户双击带有该扩展名的文件时就启动应用程序，就可以使用 File Types 编辑器。此编辑器可以使用 View | Editor | File Types 启动，图 17-28 显示的是添加了一个定制文件扩展名的 File Types 编辑器。

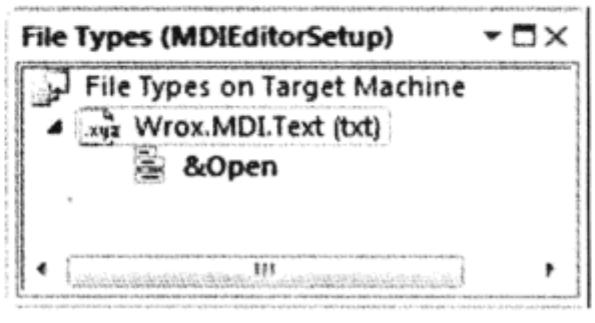

图 17-28

利用 File Type 编辑器可以配置可在应用程序中处理的文件扩展名。文件扩展名具有表 17-3 所示的属性。

表 17-3

属 性	说 明
Name	在此添加描述文件类型的有效名称。此名称显示在 File Type Editor 中，并写入到注册表中。名称应该是唯一的。.doc 文件类型的示例是 Word.Document.12。无须像在 Word 中那样使用 ProgID，也可以为.dochtml 文件扩展名使用简单的文本，如 wordhtmlfile
Command	使用 Command 属性可以指定可执行文件，当用户打开此类型的文件时，就会启动这个可执行文件
Description	在此可以添加说明
Extensions	应用程序可以进行注册的文件扩展名，在注册表中注册的文件扩展名
Icon	为文件扩展名指定要显示的图标

创建操作

在 File Type Editor 中创建了文件类型之后，就可以添加操作(action)。自动添加的默认操作是 Open。也可以添加其他动作，如 New 和 Print，或者是程序可以对文件执行的任何操作。对于操作，

必须定义 Arguments 和 Verb 属性。Arguments 属性指定传递给应用程序的参数，已经为文件扩展名对其进行了注册。例如，"%1"的意思是，文件名传递给应用程序。Verb 属性指定要发生的操作。如果应用程序支持的话，对于打印操作可以添加/print。

下面给 MDIEditor 安装程序添加操作。我们希望注册文件扩展名，这样就可以在 Windows Explorer 中使用 MDIEditor 应用程序，打开带有扩展名.txt 的文件。在注册后，可以双击这些文件打开它们，自动启动 MDIEditor 应用程序。

试一试：设置文件扩展名

(1) 通过 View | Editor | File Types 菜单项启动 File Types 编辑器。使用菜单项 Action | Add File Type 添加新文件类型，其属性如表 17-4 所示。

表　17-4

属　性	值
(Name)	Wrox.MDIEditor.Text
Command	MDI Editor 的主要输出
Description	文本文档
Extensions	Txt

也可以设置 Icon 属性，为打开的文件定义图标。还可以设置 MIME 类型。

Open 操作的属性使用默认值，这样就将文件名作为应用程序参数传递。

17.5.7　Launch Condition 编辑器

通过 Launch Condition 编辑器，可以在安装之前对目标系统指定一些要求。选择菜单项 View | Editor | Launch Conditions，启动 Launch Condition 编辑器，如图 17-29 所示。

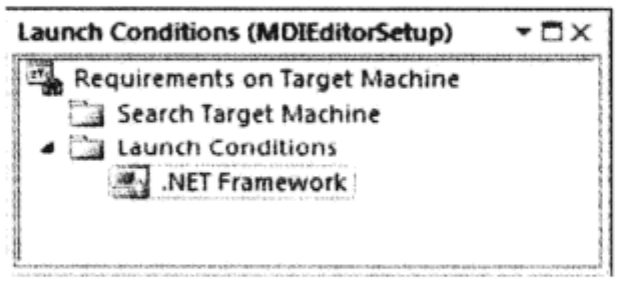

图　17-29

可以在这个编辑器的如下两部分指定要求：Search Target Machine 和 Launch Conditions。在第一部分中，可以指定要搜索什么文件或注册表项，如果搜索失败，则在第二部分中定义错误消息。下面研究一些可以使用 Action 菜单定义的启动条件。

- File Launch Condition 在开始安装前在目标系统上搜索文件。
- Registry Launch Condition 允许在安装之前查找注册表项。
- Windows Installer Launch Condition 可以搜索必须存在的 Windows Installer 组件。
- .NET Framework Launch Condition 检查目标系统是否已经安装了.NET Framework。

- Internet Information Services Launch Condition 检查已经安装的 Internet Information Services。在 Internet Information Services 已安装的情况下，如果添加这个启动条件，会搜索定义好的特定注册表项，并添加检查特定版本的条件。

在默认情况下包含了.NET Framework Launch Condition，其属性设置为预定义的值：message 属性设置为[VSDNETMSG]，这是预定义的错误消息。如果未安装.NET Framework 4，就弹出一个消息，告诉用户安装.NET Framework。InstallUrl 默认设置为 http://go.microsoft.com/ fwlink/?linkid=131000，这样用户就很容易地启动.NET Framework 的安装。

17.5.8 User Interface 编辑器

使用 User Interface 编辑器，可以定义用户在配置安装时看到的对话框。通过它可以向用户显示许可协议，询问安装路径和配置应用程序的其他信息。

下面的示例将启动 User Interface 编辑器，以配置安装应用程序时显示的对话框。

试一试：启动 User Interface 编辑器

(1) 选择 View | Editor | User Interface 菜单项，启动 User Interface 编辑器。

(2) 使用 User Interface 编辑器可以为预定义的对话框设置属性。图 17-30 显示了自动生成的对话框和两个安装模式。

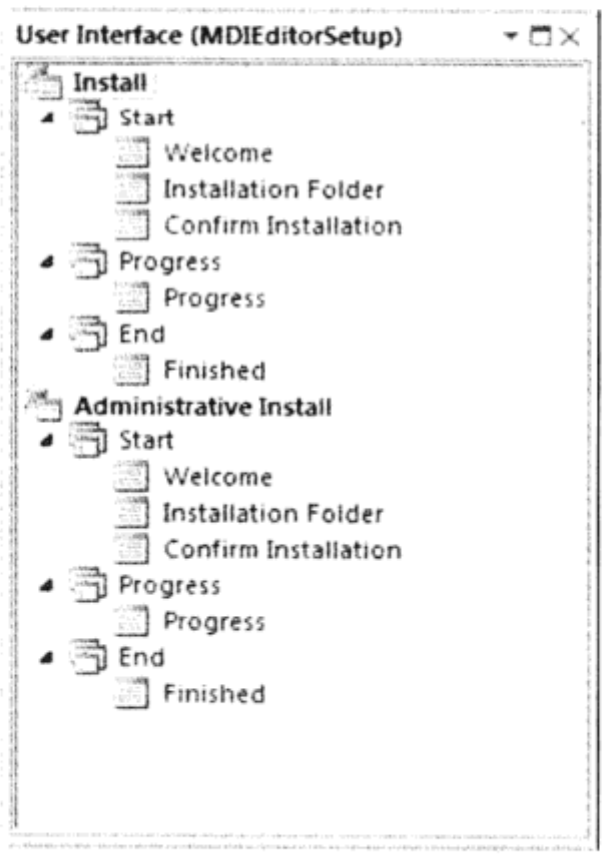

图 17-30

示例的说明

如图 17-30 所示，有两种安装模式：Install 和 Administrative Install。Install 模式一般用于在目标系统上安装应用程序。而 Administrative Install 模式可以在网络共享上安装应用程序的映像，然后用户就可以从网络安装应用程序。

在两种安装模式中，显示对话框有 3 个阶段：Start、Progress 和 End。下面介绍默认的对话框：

- Welcome 对话框向用户显示欢迎消息。可以使用自己的消息代替默认的欢迎文本。用户只能取消安装或单击 Next 按钮。
- 通过第二个对话框 Installation Folder，用户可以选择安装应用程序的文件夹。如果添加了定制的对话框(稍后介绍)，就必须在此对话框之前添加。
- Confirm Installation 对话框是安装开始之前的最后一个对话框。
- Progress 对话框显示一个进度控件，以便用户查看安装进度。
- 安装结束后会显示 Finished 对话框。

即使没有在解决方案中打开 User Interface 编辑器，默认对话框也会在安装时自动显示，但是应配置这些对话框，显示对应用程序有用的消息。

下面的示例将配置在安装应用程序时显示的默认对话框。这里忽略了 Administrative Install 路径，只配置一般安装路径。

试一试：配置默认对话框

(1) 选择 Welcome 对话框。在属性窗口中，可以看到用于对话框的 3 个属性：BannerBitmap、CopyrightWarning 和 WelcomeText。选择 BannerBitmap 属性，方式是在组合框中单击 Browse，然后选择 Application Folder\Setup 文件夹中的文件 wroxsetuplogo.bmp。此文件存储的位图会显示在对话框顶部。

属性 CopyrightWarning 的默认文本为：

```
WARNING: This computer program is protected by copyright law and international treaties.
Unauthorized duplication or distribution of this program, or any portion of it, may result
in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible
under the law.(警告：此计算机程序受到版权法和国际公约保护。未经授权，复制或销售此程序，或其中的一部分，
都会受到法律许可的最大限度的民事或刑事处罚。)
```

此文本也会显示在 Welcome 对话框中。如果希望发出更严厉的警告，可以改变此文本。属性 WelcomeText 可以定义在对话框中显示的更多文本，其默认值如下：

```
The installer will guide you through the steps required to install [ProductName] on your
computer.(安装程序会引导您完成在计算机上安装[ProductName]的全部步骤)。
```

也可以改变此文本。字符串[ProductName]会自动被在项目属性中定义的属性 ProductName 代替。

(2) 选择对话框 Installation Folder。此对话框仅有两个属性 BannerBitmap 和 InstallAllUsersVisible。InstallAllUsersVisible 的默认值是 true。如果这个值设置为 false，就只有已登录的用户可以安装该应用程序。按照 Welcome 对话框的方式，把 BannerBitmap 属性改为 wroxsetuplogo.bmp 文件。每个对话框都可以用这个属性显示位图，所以也可以在其他的对话框中改变此属性。

1. 其他对话框

如果设计了一个定制对话框，您无法将其添加到 Visual Studio 安装程序的安装序列中。完成这个任务需要更专业的工具，如 Windows 的 InstallShield 或 Wise。但是有了 Visual Studio 安装程序，可以在 Add Dialog 窗口中添加和定制许多预定义的对话框。