# 第 5 章　使用提供程序

PowerShell 中较难以理解的一部分是如何使用提供程序。在这里提前声明，本章某些内容或许对你们来说有点难。我们期许读者对 Windows 的文件系统比较熟悉，比如，你们可能知道如何通过 Windows 命令行窗口管理文件系统。请记住，我们会利用与命令行管理文件系统类似的方式解释一些内容，读者可以借助之前所熟悉的文件系统的知识作为铺垫，从而更好地使用提供程序。同时，也请谨记，PowerShell 并不是 Cmd.exe。你可能觉得某些东西看起来差不多，但是我们确信它们大有不同。

## 5.1　什么是提供程序

一个 PowerShell 的提供程序，或者说 PSProvider，其本质上是一个适配器。它可以访问某些数据存储介质，并使得这些介质看起来像是磁盘驱动器一样。你可以通过下面的命令查看当前 Shell 中已经存在的提供程序。

```
PS C:\Users\gaizai> Get-PSProvider

Name                 Capabilities                          Drives
----                 ------------                          ------
Alias                ShouldProcess                         {Alias}
Environment          ShouldProcess                         {Env}
FileSystem           Filter, ShouldProcess, Credentials    {C, D, A}
Function             ShouldProcess                         {Function}
Registry             ShouldProcess, Transactions           {HKLM, HKCU}
Variable             ShouldProcess                         {Variable}
```

我们可以通过模块或者管理单元将一些提供程序添加到 PowerShell 中，这也是 PowerShell 仅支持的两种扩展方式。（我们会在第 7 章中讲解该部分知识。）有些时候，如果启用了某些 PowerShell 功能，可能也会新增一个新的 PSProvider。比如，当开启了

远程处理时（将在第 13 章中讨论该话题），会新增一个 PSProvider，比如：

```
PS C:\Users\gaizai> Get-PSProvider
Name                 Capabilities                          Drives
--------             ---------------                       -------
Alias                ShouldProcess                         {Alias}
Environment          ShouldProcess                         {Env}
FileSystem           Filter, ShouldProcess, Credentials    {C, D, A}
Function             ShouldProcess                         {Function}
Registry             ShouldProcess, Transactions           {HKLM, HKCU}
Variable             ShouldProcess                         {Variable}
WSMan                Credentials                           {WSMan}
```

我们可以看出每个提供程序都有各自不同的功能。这非常重要，因为这将决定我们如何使用这些提供程序。下面是常见的一些功能描述。

- ShouldProcess——意味着这部分提供程序支持-WhatIf 和-Confirm 参数，保证我们在正式执行这部分脚本之前可以对它们进行测试。
- Filter——意味着在 Cmdlet 中操作提供程序的数据时，支持-Filter 参数。
- Credentials——意味着该提供程序允许使用可变更的凭据去连接数据存储。这也就是-Credentials 参数的作用。
- Transactions——意味着该提供程序支持事务，也就是允许你在该提供程序中将多个变更作为一个原子操作进行提交或者全部回滚。

你也可以使用某个提供程序去创建一个 PSDrive。PSDrive 可以通过一个特定的提供程序去连接到某些存储数据的介质。这和在 Windows 资源管理器中类似，本质上是创建了一个驱动器映射。但是由于 PSDrive 使用了提供程序，除了可以连接磁盘之外，还能连接更多的数据存储介质。运行下面的命令，可以看到当前已连接的驱动器。（译者注：返回基于 PowerShell 3.0）

```
PS C:\Users\gaizai> Get-PSDrive

Name     Used (GB)     Free (GB) Provider      Root
------   ---------     --------- --------      ------
A                                FileSystem    A:\
Alias                            Alias
C           29.40         97.60 FileSystem    C:\
Cert                             Certificate   \
D           1.26         283.74 FileSystem    D:\
Env                              Environment
Function                         Function
HKCU                             Registry      HKEY_CURRENT_USER
HKLM                             Registry      HKEY_LOCAL_MACHINE
```

```
Variable                                Variable
WSMan                                   WSMan
```

在上面返回的列表中，可以看到有三个驱动器使用了 FileSystem 提供程序，两个使用了 Registry 提供程序，等等。PSProvider 会适配对应的数据存储，通过 PSDrive 机制使得数据存储可被访问，然后可以使用一系列 Cmdlets 去查阅或者操作每个 PSDrive 呈现出来的数据。通常我们可以通过下面的命令来查询某个 PSDrive 的 Cmdlets 中有哪些命令的名称中包含“Item”字符（译者注：返回结果基于 PowerShell 3.0）。

```
PS C:\Users\gaizai> Get-Command -noun *Item*
CommandTypeName      Name
---------------      ----------------
 Function            Get-DAEntryPointTableItem      DirectAccessClientComponents
 Function            New-DAEntryPointTableItem      DirectAccessClientComponents
 Function            Remove-DAEntryPointTableItem   DirectAccessClientComponents
 Function            Rename-DAEntryPointTableItem   DirectAccessClientComponents
 Function            Reset-DAEntryPointTableItem    DirectAccessClientComponents
 Function            Set-DAEntryPointTableItem      DirectAccessClientComponents
 Cmdlet              Clear-Item                     Microsoft.PowerShell.Management
 Cmdlet              Clear-ItemProperty             Microsoft.PowerShell.Management
 Cmdlet              Copy-Item                      Microsoft.PowerShell.Management
 Cmdlet              Copy-ItemProperty              Microsoft.PowerShell.Management
 Cmdlet              Get-ChildItem                  Microsoft.PowerShell.Management
 Cmdlet              Get-ControlPanelItem           Microsoft.PowerShell.Management
 Cmdlet              Get-Item                       Microsoft.PowerShell.Management
 Cmdlet              Get-ItemProperty               Microsoft.PowerShell.Management
 Cmdlet              Invoke-Item                    Microsoft.PowerShell.Management
 Cmdlet              Move-Item                      Microsoft.PowerShell.Management
 Cmdlet              Move-ItemProperty              Microsoft.PowerShell.Management
 Cmdlet              New-Item                       Microsoft.PowerShell.Management
 Cmdlet              New-ItemProperty               Microsoft.PowerShell.Management
 Cmdlet              Remove-Item                    Microsoft.PowerShell.Management
 Cmdlet              Remove-ItemProperty            Microsoft.PowerShell.Management
 Cmdlet              Rename-Item                    Microsoft.PowerShell.Management
 Cmdlet              Rename-ItemProperty            Microsoft.PowerShell.Management
 Cmdlet              Set-Item                       Microsoft.PowerShell.Management
 Cmdlet              Set-ItemProperty               Microsoft.PowerShell.Management
 Cmdlet              Show-ControlPanelItem          Microsoft.PowerShell.Management
```

我们将在系统中使用上述 Cmdlet 或者它们的别名来调用提供程序。或许对你而言，文件系统应该算是最熟悉的提供程序了，所以我们将会从文件系统 PSProvider 开始学习。

## 5.2 FileSystem 的结构

Windows 文件系统主要由三种对象组成：磁盘驱动器、文件夹和文件。磁盘驱动器是最上层的对象，包含文件夹和文件。文件夹是一种容器对象，它可以包含文件以及其他文件夹。文件不是一种容器对象，它更可以算作最小单位的对象。

你或许习惯于通过 Windows 资源管理器来查看 Windows 的文件系统，如图 5.1 所示。在图中，我们可以直观地观察到磁盘驱动器、文件夹和文件的层级分布。

PowerShell 中的术语和文件系统中的略有不同。因为 PSDrive 可能不是指向某个文件系统——比如 PSDrive 可以映射到注册表（显然注册表并不是一种文件系统），所以 PowerShell 并不会使用“文件”以及“文件夹”的说法。相反，PowerShell 采用更通俗的说法“项”（Item）。一个文件或者一个文件夹都叫作项，尽管本质上是两种不同的项。这也就是为什么前面返回的 Cmdlet 名字中都有“Item”字符。

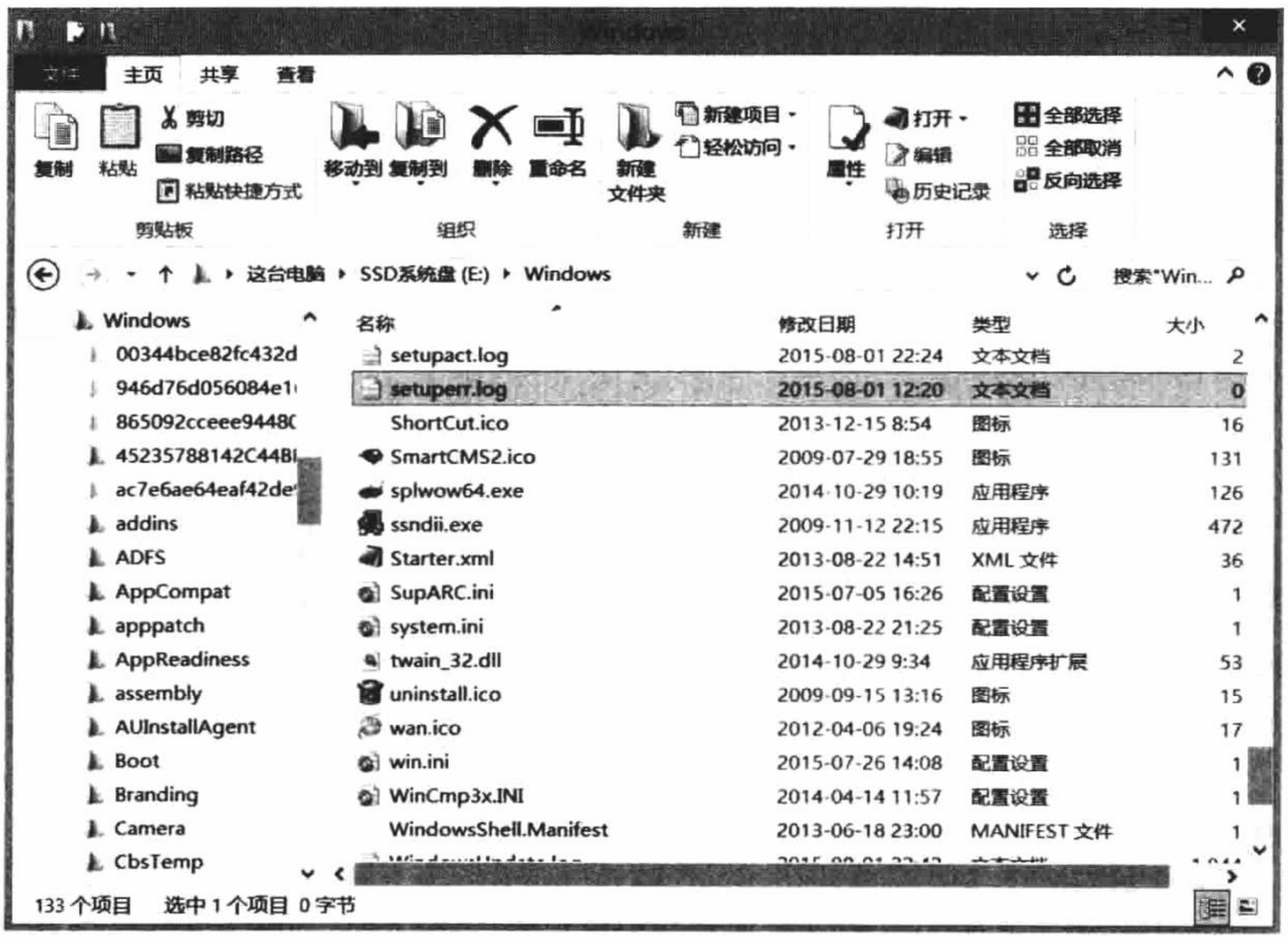


图 5.1 在 Windows 资源管理器中查看文件、文件夹及磁盘

每个项基本上都会存在对应的属性。比如，一个文件项可能有最后写入的时间，是否只读等属性。一些项，比如文件夹，可能包含子项（子项包含在文件夹项中）。了解这些信息会有助于你们理解前面演示的命令列表中的名词以及动词：

- 比如 Clear、Copy、Get、Move、New、Remove、Rename 以及 Set 等动词可以应用于这些项（比如文件或者文件夹）以及它们对应的属性（比如该项最后写入的时间或者该项是否只读）。
- 单个对象对应的项名词，比如文件或者文件夹。
- ItemProperty 代表一个项对应的属性。比如只读、项创建时间、长度等。
- 子项代表一个项（比如文件或者子文件夹）包含于另外一个项（文件夹）中。

需要记住的是，这些 Cmdlet 都是通用的，因为它们需要处理各种不同的数据源。但是某些 Cmdlet 在某些特定场合下不一定能正常工作。比如，FileSystem 提供程序不支持事务，所以文件系统驱动器下的 Cmdlet 中的命令都不支持-UseTransaction 参数。再比如，注册表不支持 Filter 功能，所以注册表驱动器下的 Cmdlet 也都不支持-Filter 参数。

某些 PSProvider 并不具有对应的项属性。比如，Environment 这个 PSProvider 主要用来构造 PowerShell 中可用的 ENV：类型驱动器（如 Env:\PSModulePath）。这个驱动器主要的作用是访问 Windows 中的环境变量，但是如下所示，它并没有对应的项属性。

```
PS C:\Users\gaizai> Get-ItemProperty -Path Env:\PSModulePath
Get-ItemProperty : 无法使用接口。此提供程序不支持 IPropertyCmdletProvider 接口。
所在位置 行:1 字符: 17
+ Get-ItemProperty <<<<  -Path Env:\PSModulePath
    + CategoryInfo          : NotImplemented: (:) [Get-ItemProperty], PSNotSup
   portedException
    +FullyQualifiedErrorId: NotSupported, Microsoft.PowerShell.Commands.
   GetItemPropertyCommand
```

对刚接触 Windows PowerShell 的朋友而言，由于每个 PSProvider 都不尽相同，可能会导致你们无法很好地理解各种提供程序。你们必须去了解每个提供程序能够实现什么功能，并且认识到即便 Cmdlet 知道如何实现某些功能，但是并不意味着该提供程序真正支持对应的操作。

## 5.3 文件系统——其他数据存储的模板

其他形式的数据源存储衍生于文件系统，所以严格算起来，文件系统可以算作其他数据存储的模板。例如，图 5.2 展示了 Windows 注册表的结构。

注册表以类似文件系统的结构呈现，其中注册表的键等同于文件系统中的文件夹，对应的键值类似于文件系统中的文件，等等。正是这种广泛的相似性，使得文件系统成为其他形式数据源的最佳模板。所以当用 PowerShell 访问其他数据存储的时候，显

示为驱动器的形式（可以依次展开为项以及查看对应的属性）。但是相似性到这一层级也就结束了：如果你再继续向下展开，那么你会发现不同形式的存储其实差别很大。这也就是为什么各种项的 Cmdlet 支持如此多的功能，但是并不是每个功能在每种存储中都能运行。

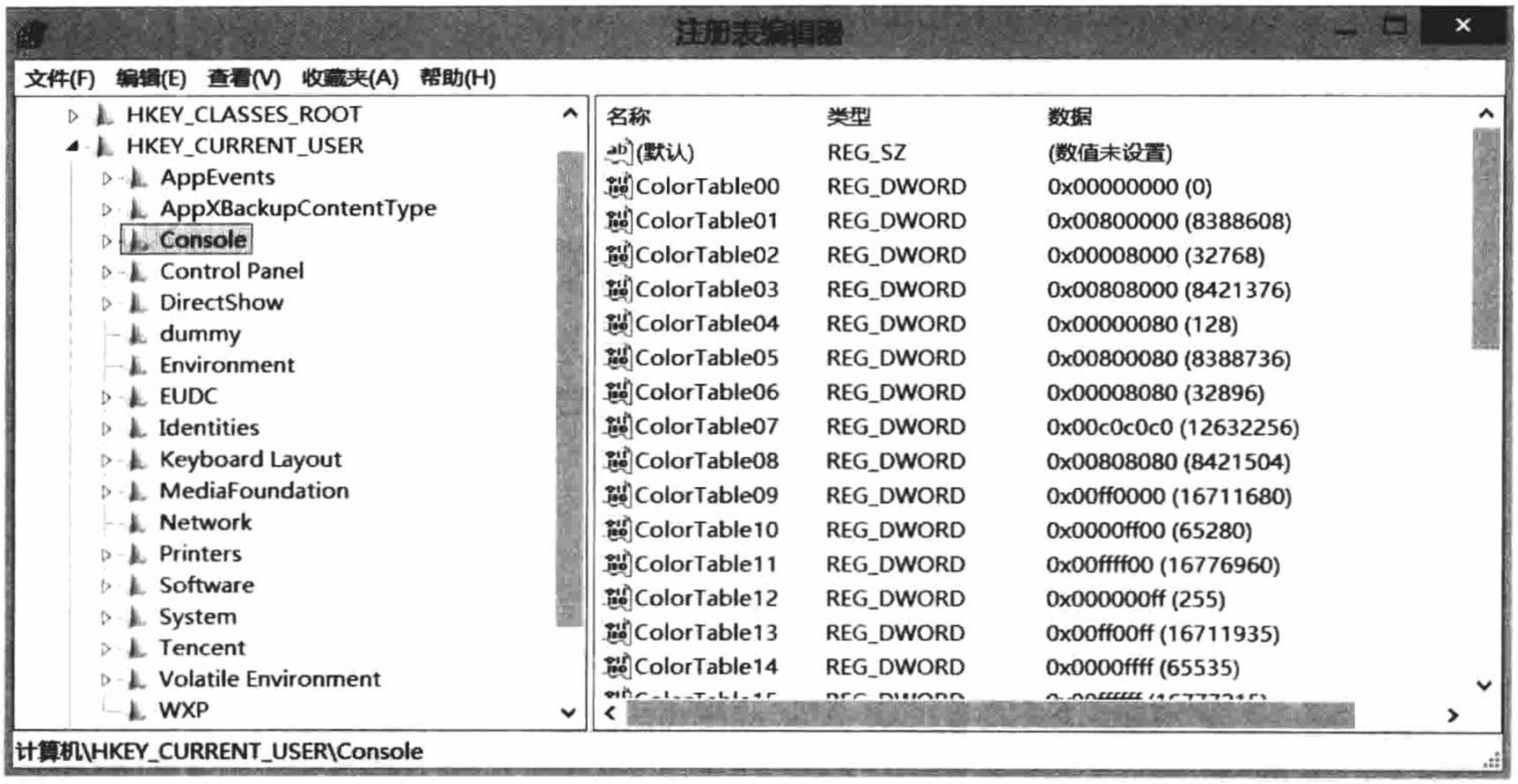


图 5.2 注册表和文件系统具有相同的分层结构

## 5.4 使用文件系统

在使用提供程序时，需要熟悉的另外一个 Cmdlet 是 Set-Location。该参数的功能是将 Shell 中当前路径变更为不同路径，比如变更到另一个文件夹下：

```
PS C:\Users\gaizai> Set-Location -Path C:\Windows
PS C:\Windows>
```

你可能对该命令的另一种写法 cd 更为熟悉，其实就是 cmd.exe 中的 change directory 的简写。

```
PS C:\Windows> cd 'C:\Program Files'
PS C:\Program Files>
```

这里我们使用了该别名，然后传入特定的路径作为位置参数。

PowerShell 中另外一个比较棘手的任务是创建新的项。比如，如何创建一个新的目录。执行 New-Item，之后会弹出一个新的窗口。

```
PS C:\Users\gaizai> New-Item testFolder
Type:
```

需要注意的是，New-Item 这个 Cmdlet 在很多地方都是通用的——它根本无法得知你是想新建一个文件夹。这个 Cmdlet 可以用来新建文件夹、文件、注册表项以及其他项，所以你必须告知你希望创建的类型是什么。

```
PS C:\Users\gaizai> New-Item testFolder
Type: Directory

    目录: C:\Users\gaizai
Mode                 LastWriteTime         Length Name
----                 -------------         ------ -------
d-----          2015/1/5   14:18                testFolder
```

PowerShell 中也包含 MKDir 命令。很多人都认为该命令是 New-Item 的别名，但是你们在执行 MKDir 之后，并不需要输入类型。

```
PS C:\Users\gaizai> Mkdir test2

    目录: C:\Users\gaizai
Mode                 LastWriteTime         Length Name
----                 -------------         ------ -------
d-----          2015/1/5   14:22                test2
```

你可能已经发现，Mkdir 是一个函数，而并不是一个别名。但是实际上，它仍然调用了 New-Item，只不过隐式赋予了-Type Directory 这个参数，这样使得 MkDir 看起来更像一种 Cmd.exe。请记住这一点以及其他的一些小细节，当使用到这些提供程序时，会很有用处。你知道并不是每个提供程序都是一样，并且项的 Cmdlet 又是非常通用的，所以在真正使用这些提供程序之前需要思考更多。

## 5.5 使用通配符以及绝对路径

大部分项的 Cmdlet 都包含了-Path 属性。默认情况下，该属性支持通配符输入。比如，我们查看 Get-ChildItem 的完整帮助文档，如下所示：

```
PS C:\Users\gaizai> Get-Help Get-ChildItem -Full
-Path <String[]>
    指定一个或多个位置的路径。允许使用通配符。默认位置为当前目录 (.)。

    是否必需?                    False
    位置?                        1
    默认值                       Current directory
```

```
是否接受管道输入?        true (ByValue, ByPropertyName)
是否接受通配符?          True
```

“*”通配符代表 0 个或者多个字符，“?”通配符仅代表单个字符。你应该曾经多次使用过这两种通配符，当然你可能使用的是 Get-ChildItem 的别名 Dir。

```
PS C:\windows> Dir *.exe

    目录: C:\windows

Mode                 LastWriteTime         Length Name
----                 -------------         ------ -------
-a---          2012/7/26      3:08          75264 bfsvc.exe
-a---           2013/6/1     11:34        2391280 explorer.exe
-a---          2012/11/6      4:20         883712 HelpPane.exe
-a---          2012/7/26      3:08          17408 hh.exe
-a---          2012/7/26      3:08         159232 regedit.exe
-a---          2012/7/26      3:08         126464 splwow64.exe
-a---          2012/7/26      3:21          10752 winhlp32.exe
-a---          2012/7/26      3:08          10752 write.exe
```

前面例子中列出来的通配符和微软的文件系统中一样（都是采用 MS-DOS 中的方式）。“*”和“?”比较特殊，它们是通配符，所以在文件或者文件夹中不允许带有“*”或者“?”字符。但是在 PowerShell 中，并不仅支持文件系统格式的数据存储。在大部分其他类型的数据存储中，“*”和“?”都可以包含在 Item 的名称中。比如，在注册表中，你可以看到一些项的名称中包含“?”字符。你应该发现了，这将导致一个问题。当在一个路径中使用了“*”或者“?”，PowerShell 会如何对待，是作为一个通配符还是一个特定的字符？比如，如果你输入 Windows?来寻找某个项，你到底是想寻找名字就是 Windows?的项，还是将“?”看成一个通配符，然后返回 Windows 7 或者是 Windows 8 的值。

针对此问题，PowerShell 给出的解决办法是新增一个参数-LiteralPath。该参数并不支持通配符。

```
-LiteralPath <String[]>
    指定一个或多个位置的路径。与 Path 参数不同，LiteralPath 参数的值严格按
    照其键入形式使用。不会将任何字符解释为通配符。如果路径包括转义符，请将
    其括在单引号中。单引号会告知 Windows PowerShell 不要将所有字符都解释为
    转义序列。

    是否必需?                    True
    位置?                        named
    默认值
    是否接受管道输入?            true (ByValue, ByPropertyName)
    是否接受通配符?              False
```

如果需要查询名字中带有*或者?，就要使用-LiteralPath 参数，而不要使用-Path。需要注意的是，-LiteralPath 这个参数不可隐式赋予。如果确定需要使用该参数，必须显式申明-LiteralPath 参数。如果你在一开始就提供了路径，如我们前面例子所示的*.exe，那么它会被隐式转化为-Path 参数，通配符也是如此。

## 5.6 使用其他提供程序

如果想对其他提供程序有个大致的认识，以及了解其他项的 Cmdlet 如何工作，最好的办法就是去尝试一下非文件系统格式的 PSDrive。在 PowerShell 内置的提供程序中，注册表应该算是比较好的一个示例，我们可以进行简单的尝试。（选择注册表作为示例，部分原因是它在每个版本的 Windows 中都存在）。下面的例子中，我们最终要达成的目的是关闭 Windows 中的桌面透明特性。

我们现在先将路径切换到HKEY_CURRENT_USER，在PowerShell中显示为HKCU:驱动器。

```
PS C:\windows> Set-Location -Path HKCU:
```

接下来看注册表的右边

```
PS HKCU:\> Set-Location -Path SoftWare
PS HKCU:\SoftWare> Get-ChildItem


   Hive: HKEY_CURRENT_USER\SoftWare

Name                           Property
----                           --------
AppDataLow
Microsoft
Mine                           (default) : {}
Policies
Wow6432Node
Classes

PS HKCU:\SoftWare> Set-Location Microsoft
PS HKCU:\SoftWare\Microsoft> Get-ChildItem


   Hive: HKEY_CURRENT_USER\SoftWare\Microsoft

Name                           Property
----                           --------
Active Setup
Advanced INF Setup
```

```
Assistance
Command Processor              PathCompletionChar : 9
                               EnableExtensions   : 1
                               CompletionChar     : 9
                               DefaultColor       : 0
CTF
EventSystem
Feeds
FTP                            Use PASV : yes
IME
Internet Connection Wizard     Completed : 1
Internet Explorer
MSF
ServerManager                  InitializationComplete       : 1
                               CheckedUnattendLaunchSetting : 1
SystemCertificates
WAB
Windows
Windows NT
Wisp
```

测试到这里基本上就结束了（后面的部分基本上都是重复的命令）。你可以看到，我们前面都是用 Cmdlet 的全称，并没有使用它们的别名，这样可以更加强化我们对 Cmdlet 本身的认识。

```
PS HKCU:\SoftWare\Microsoft> Set-Location .\Windows
PS HKCU:\SoftWare\Microsoft\Windows> Get-ChildItem

   Hive: HKEY_CURRENT_USER\SoftWare\Microsoft\Windows

Name                           Property
----                           --------
CurrentVersion
DWM                            Composition                  : 1
                               ColorizationColor            : 3226847725
                               ColorizationColorBalance     : 87
                               ColorizationAfterglow        : 3226847725
                               ColorizationAfterglowBalance : 10
                               ColorizationBlurBalance      : 3
                               EnableWindowColorization     : 1
                               ColorizationGlassAttribute   : 1
Roaming
Shell
Windows Error Reporting        Disabled                : 0
                               MaxQueueCount           : 50
```

```
DisableQueue            : 0
LoggingDisabled         : 0
DontSendAdditionalData  : 0
ForceQueue              : 0
DontShowUI              : 0
ConfigureArchive        : 1
MaxArchiveCount         : 500
DisableArchive          : 0
LastQueuePesterTime     : 130649182874302227
LastQueueNoPesterTime   : 130649188485744670
```

你可以在该列表中看到 EnableWindowColorization 的键值，现在将它修改为 0。

```
PS HKCU:\SoftWare\Microsoft\Windows> Set-ItemProperty -Path DWM -PSPropert Enabl
eWindowColorization -Value 0
```

下面再执行之前的命令来确认修改已经生效。

```
PS HKCU:\SoftWare\Microsoft\Windows> Get-ChildItem

   Hive: HKEY_CURRENT_USER\SoftWare\Microsoft\Windows

Name                           Property
----                           --------
CurrentVersion
DWM                            Composition                    : 1
                               ColorizationColor              : 3226847725
                               ColorizationColorBalance       : 87
                               ColorizationAfterglow          : 3226847725
                               ColorizationAfterglowBalance   : 10
                               ColorizationBlurBalance        : 3
                               EnableWindowColorization       : 0
                               ColorizationGlassAttribute     : 1
Roaming
Shell
Windows Error Reporting        Disabled           : 0
                               MaxQueueCount          : 50
                               DisableQueue           : 0
                               LoggingDisabled        : 0
                               DontSendAdditionalData : 0
                               ForceQueue             : 0
                               DontShowUI             : 0
                               ConfigureArchive       : 1
                               MaxArchiveCount        : 500
                               DisableArchive         : 0
                               LastQueuePesterTime    : 130649182874302227
                               LastQueueNoPesterTime  : 130649188485744670
```

到这里，这个示例已经全部完成，可以看到这个值的修改已经生效。采用这种方法，你可以处理其他提供程序类似的问题。

## 5.7 动手实验

**注意：**本章动手实验环节，需要运行 3.0 版本或者版本更新的 PowerShell。

完成如下任务：

1．在注册表中，定位到 HKEY_CURRENT_USER\software\microsoft\Windows \currentversion\explorer。选中“Advanced”项，然后修改 DontPrettyPath 的值为 0。

2．创建空文件 C:\Test.txt（使用 New-Item 命令）。

3．尝试使用 Set-Item 去修改 Test.txt 的内容为 TESTING，是否可行？或者是否有报错？同时，也请想一下：为什么会报错？

4．Get-ChildItem 的-Filter、-Include 和-Exclude 参数之间有什么不同？

## 5.8 进一步学习

你可以看到，对大部分的其他软件程序包而言都存在提供程序，比如 Internet Information Service（IIS）、SQL Server，甚至是活动目录。很多时候，这些产品的开发者都会选择使用提供程式，因为这样他们的产品才会具有动态扩展功能。他们不知道以后还会有什么功能加到他们的产品中，所以他们并不会写一个静态的命令集。提供程序可以保证开发者能一致性地动态扩展他们的结构，所以特别是对 IIS 和 SQL Server 团队而言，都会搭配使用 Cmdlet 和提供程序。

如果你需要使用这些产品（如果是 IIS，那么请使用 7.5 或者之后的版本；如果是 SQL Server，我们建议使用 SQL Server 2012 或者之后的版本），请花费一定的时间去研究一下对应的提供程序。那么你会发现，这些产品研发部门已经将其“驱动器”结构安排得很好，因此你很容易发现如何使用本章中讲解到的 Cmdlet 命令去查看以及修改对应的配置选项或者其他的详细配置。