# 第 22 章　优化可传参脚本

在之前章节，我们给你留下了许多非常好的可传参的脚本。可传参脚本的思想是脚本的使用者无须关心或者干预脚本的内容。脚本的使用者只需通过设计好的界面提供输入——也就是参数，使用者能够修改的部分只有参数。在本章，我们将对该部分内容进一步探索。

## 22.1　起点

为了确保我们在同一起点，让我们使用代码清单 22.1 作为起点。该脚本以基于注释的帮助为特点，包括两个输入参数和一个使用输入参数的命令。我们基于之前章节做了小幅修改：我们将输出结果输出为被选择的对象，而不是格式化之后的表格。

**代码清单 22.1　起点：Get-DiskInventory.ps1**

```
<#
.SYNOPSIS
Get-DiskInventory retrieves logical disk information from one or
more computers.
.DESCRIPTION
Get-DiskInventory uses WMI to retrieve the Win32_LogicalDisk
instances from one or more computers. It displays each disk's
drive letter, free space, total size, and percentage of free
space.
.PARAMETER computername
The computer name, or names, to query. Default: Localhost.
.PARAMETER drivetype
The drive type to query. See Win32_LogicalDisk documentation
for values. 3 is a fixed disk, and is the default.
```

```
.EXAMPLE
Get-DiskInventory -computername SERVER-R2 -drivetype 3
#>
param (
  $computername = 'localhost',
  $drivetype = 3
)
Get-WmiObject -class Win32_LogicalDisk -computername $computername `
 -filter "drivetype=$drivetype" |
 Sort-Object -property DeviceID |
 Select-Object -property DeviceID,
     @{name='FreeSpace(MB)';expression={$_.FreeSpace / 1MB -as [int]}},
     @{name='Size(GB';expression={$_.Size / 1GB -as [int]}},
     @{name='%Free';expression={$_.FreeSpace / $_.Size * 100 -as [int]}}
```

为什么我们使用 Select-Object 而不是 Format-Table？因为我们通常感觉写一个脚本所产生的结果是已格式化的并不是一个好主意。毕竟，如果某个用户需要 CSV 格式的文件，而脚本输出格式化后的表，该用户就无法完成工作。通过本次修改，我们可以通过下述方式获得格式化后的表。

```
PS C:\> .\Get-DiskInventory | Format-Table
```

或者通过下述方式运行获取 CSV 文件。

```
PS C:\> .\Get-DiskInventory | Export-CSV disks.csv
```

关键点是输出对象（也就是 Select-Object 完成的工作），对照格式化的显示结果，将会使得我们的脚本从长远角度来说更加灵活。

## 22.2 让 PowerShell 去做最难的工作

我们只需在上述脚本的基础上再多加一行脚本，就能展现 PowerShell 的奇妙。这使得从技术上来说，把我们的脚本变为所谓的“高级脚本”，使得大量 PowerShell 能做的事得以展现。代码清单 22.2 展现了修订后的脚本。

**代码清单 22.2 将 Get-DiskInventory.ps1 变为高级脚本**

```
<#
.SYNOPSIS
Get-DiskInventory retrieves logical disk information from one or
more computers.
.DESCRIPTION
Get-DiskInventory uses WMI to retrieve the Win32_LogicalDisk
instances from one or more computers. It displays each disk's
```

```
drive letter, free space, total size, and percentage of free
space.
.PARAMETER computername
The computer name, or names, to query. Default: Localhost.
.PARAMETER drivetype
The drive type to query. See Win32_LogicalDisk documentation
for values. 3 is a fixed disk, and is the default.
.EXAMPLE
Get-DiskInventory -computername SERVER-R2 -drivetype 3
#>
[CmdletBinding()]
param (
  $computername = 'localhost',
  $drivetype = 3
)
Get-WmiObject -class Win32_LogicalDisk -computername $computername `
 -filter "drivetype=$drivetype" |
 Sort-Object -property DeviceID |
 Select-Object -property DeviceID,
     @{name='FreeSpace(MB)';expression={$_.FreeSpace / 1MB -as [int]}},
     @{name='Size(GB';expression={$_.Size / 1GB -as [int]}},
     @{name='%Free';expression={$_.FreeSpace / $_.Size * 100 -as [int]}}
```

在基于备注的帮助代码后面，将[CmdletBinding()]指示符置于脚本的第一行非常重要。PowerShell 只会在该位置查看该指示符。加上这个指示符之后，脚本还会正常运行。但我们已经启用了好几个功能，我们会在接下来进行探索。

## 22.3 将参数定义为强制化参数

我们对现有的脚本并不满意，这是由于它提供了默认的-ComputerName 参数。我们并不确定是否真正需要该参数。我们更倾向于选择提示用户输入值。幸运的是，PowerShell 中实现该功能很简单——同样，只需要添加一行代码就能完成，如代码清单 22.3 所示。

**代码清单 22.3 为 Get-DiskInventory.ps1 添加一个强制参数**

```
<#
.SYNOPSIS
Get-DiskInventory retrieves logical disk information from one or
more computers.
.DESCRIPTION
Get-DiskInventory uses WMI to retrieve the Win32_LogicalDisk
instances from one or more computers. It displays each disk's
drive letter, free space, total size, and percentage of free
```

```
space.
.PARAMETER computername
The computer name, or names, to query. Default: Localhost.
.PARAMETER drivetype
The drive type to query. See Win32_LogicalDisk documentation
for values. 3 is a fixed disk, and is the default.
.EXAMPLE
Get-DiskInventory -computername SERVER-R2 -drivetype 3
#>
[CmdletBinding()]
param (
  [Parameter(Mandatory=$True)]
  [string]$computername,
  [int]$drivetype = 3
)
Get-WmiObject -class Win32_LogicalDisk -computername $computername `
 -filter "drivetype=$drivetype" |
 Sort-Object -property DeviceID |
 Select-Object -property DeviceID,
     @{name='FreeSpace(MB)';expression={$_.FreeSpace / 1MB -as [int]}},
     @{name='Size(GB';expression={$_.Size / 1GB -as [int]}},
     @{name='%Free';expression={$_.FreeSpace / $_.Size * 100 -as [int]}}
```

**补充说明**

当某个用户使用你写的脚本，却没有为强制参数提供值时，PowerShell 将会提示他输入参数值。有两种方式可以使得 PowerShell 给用户提供有意义的提示。

首先，使用有意义的参数名称。提示用户为名称为“comp”的参数赋值，远不如提示用户为名称为“ComputerName”的参数赋值有意义。所以请尝试使用具有自描述性的参数名称，并与其他 PowerShell 命令使用的参数名称保持一致。

你还可以添加一条帮助信息。

```
[Parameter(Mandatory=$True, HelpMessage="Enter a computer name to query")
```

某些 PowerShell 宿主程序会将帮助信息作为提示的一部分，使得用户获得更简洁的帮助信息。但并不是所有的宿主应用程序都会使用该标签，所以你测试的时候没有看到提示的帮助信息也不用沮丧。当我们写一些给他人使用的脚本时，我们喜欢在脚本中将帮助信息包含在内。这么做永远不会有任何坏处。但是为了简便起见，我们不会在本章的示例中添加帮助信息。

仅仅使用`[Parameter(Mandatory=$True)]`这样一个描述符，会使得当用户忘记提供计算机名称时，PowerShell 就会提示用户输入该参数。为了更进一步帮助 PowerShell 识别用户传入的参数，我们定义两个输入参数的数据类型：`-computerName` 定义为`[string]`类型，而`-drivetype` 定义为 INT（也就是整型）。

将这类标签添加到参数会让人困惑，因此让我们更进一步查看 Param()代码块的语法，如图 22.1 所示。

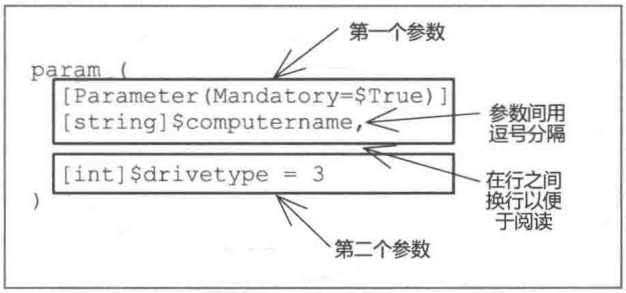


图 22.1 分解 Param()代码段的语法

下面是需要注意的重点。

- 所有的参数都必须被包括在 Param()代码段的括号内。
- 可以对一个参数添加多个修饰符，多个修饰符既可以是一行，也可以是图 22.1 中那样的多行。我们认为多行更易于阅读，但重点是即使是多行，它们也是一个整体。Mandatory 标签仅修饰-computerName 参数——它对-drivetype 参数并没有影响。
- 除了最后一个参数之外，所有的参数之间以逗号分隔。
- 为了更好的可读性，我们还喜欢在参数之间添加空格。我们认为空格会使得从视觉上分隔参数更加容易，从而减少 Param()代码段导致的困惑。
- 我们在定义参数时，就好像参数是变量——$computername 和$drivetype——但使用该脚本的人会将其当作普通的 PowerShell 命令行参数，比如说-computername 参数和-drivetype 参数。

**动手实验：**将代码清单 22.3 中的脚本保存，并在 Shell 中运行。不要为-computername 参数赋值，从而可以查看 PowerShell 以何种方式将这些信息提示给你。

## 22.4 添加参数别名

当你想到计算机名称时，“computername”是否是你想到的第一个词？或许不是。我们使用-computerName 作为参数名称，是因为该参数名称与其他 PowerShell 命令一致。查看 Get-Service、Get-WmiObject、Get-Process 以及其他命令，你可以发现这些命令都使用-computerName 作为参数名称。所以我们也同样使用该名称作为参数名称。

但假如你认为-hostname 更容易记忆的话，你可以将该名称作为备用名称添加，也就是参数别名。这只需要另外一个修饰符，如代码清单 22.4 所示。

**代码清单 22.4 为 Get-DiskInventory.ps1 添加一个别名**

```
<#
.SYNOPSIS
Get-DiskInventory retrieves logical disk information from one or
more computers.
.DESCRIPTION
Get-DiskInventory uses WMI to retrieve the Win32_LogicalDisk
instances from one or more computers. It displays each disk's
drive letter, free space, total size, and percentage of free
space.
.PARAMETER computername
The computer name, or names, to query. Default: Localhost.
.PARAMETER drivetype
The drive type to query. See Win32_LogicalDisk documentation
for values. 3 is a fixed disk, and is the default.
.EXAMPLE
Get-DiskInventory -computername SERVER-R2 -drivetype 3
#>
[CmdletBinding()]
param (
  [Parameter(Mandatory=$True)]
  [Alias('hostname')]
  [string]$computername,
  [int]$drivetype = 3
)
Get-WmiObject -class Win32_LogicalDisk -computername $computername `
 -filter "drivetype=$drivetype" |
 Sort-Object -property DeviceID |
 Select-Object -property DeviceID,
     @{name='FreeSpace(MB)';expression={$_.FreeSpace / 1MB -as [int]}},
     @{name='Size(GB)';expression={$_.Size / 1GB -as [int]}},
     @{name='%Free';expression={$_.FreeSpace / $_.Size * 100 -as [int]}}
```

完成小幅修改后，我们现在可以运行下述代码。

```
PS C:\> .\Get-DiskInventory -host SERVER2
```

**注意：**请记住，你只需输入足够让 PowerShell 分辨出是哪个参数的部分参数名即可。在本例中，-host 足以让 PowerShell 识别出指的是-hostname 参数。当然，我们也可以输入完整的参数名称。

再次声明，新增的标签是-computerName 参数的一部分，因此对-drivetype 参数不生效。现在-computerName 参数的定义占用了 3 行。当然，我们也能将三行连成一行。

```
[Parameter(Mandatory=$True)][Alias('hostname')][string]$computername,
```

我们只是认为这种方式更加难以阅读。

## 22.5 验证输入的参数

让我们和-drivetype 参数打打交道。根据 MSDN 中 WIN32_LogicalDisk 这个 WMI 类的文档（搜索类名称，在结果中，前几条记录中就有该文档），驱动器类型 3 是本地磁盘。类型 2 是可移动磁盘。可移动磁盘也会计算容量以及可用空间。驱动类型 1、4、5、6 更少被使用（还有人在继续使用类型 6 的 RAM 驱动器吗？），在某些情况下，有一些磁盘没有可用空间（比如类型为 5 的光盘）。所以我们希望阻止使用我们脚本的用户使用这些类型。

代码清单 22.5 展示了我们所需做的小幅修改。

**代码清单 22.5 为 Get-DiskInventory.ps1 添加参数验证**

```
<#
.SYNOPSIS
Get-DiskInventory retrieves logical disk information from one or
more computers.
.DESCRIPTION
Get-DiskInventory uses WMI to retrieve the Win32_LogicalDisk
instances from one or more computers. It displays each disk's
drive letter, free space, total size, and percentage of free
space.
.PARAMETER computername
The computer name, or names, to query. Default: Localhost.
.PARAMETER drivetype
The drive type to query. See Win32_LogicalDisk documentation
for values. 3 is a fixed disk, and is the default.
.EXAMPLE
Get-DiskInventory -computername SERVER-R2 -drivetype 3
#>
[CmdletBinding()]
param (
  [Parameter(Mandatory=$True)]
  [Alias('hostname')]
  [string]$computername,
  [ValidateSet(2, 3)]
  [int]$drivetype = 3
)
Get-WmiObject -class Win32_LogicalDisk -computername $computername `
 -filter "drivetype=$drivetype" |
 Sort-Object -property DeviceID |
 Select-Object -property DeviceID,
     @{name='FreeSpace(MB)';expression={$_.FreeSpace / 1MB -as [int]}},
     @{name='Size(GB)';expression={$_.Size / 1GB -as [int]}},
     @{name='%Free';expression={$_.FreeSpace / $_.Size * 100 -as [int]}}
```

新的标签告诉 PowerShell，对于参数-drivetype，只允许传入 2 和 3 这两个值，并且 3 是默认值。

还有一系列其他可以添加到参数的验证技术。如果这样做有意义，可以将多个修饰符添加到同一个参数上。运行 `help about_functions_advanced_parameters` 可以获得完整列表——目前为止，我们只使用 `ValidateSet`。Jeffery 还写了一个关于其他可能用上的“验证”标签的系列博客——你可以在网站 http://jdhitsolutions.com/blog/ 上查看到该系列博客（搜索“validate”）。

**动手实验：**将这段代码保存并再次运行——尝试指定-drivetype 参数为 5，看看 PowerShell 是如何响应的。

## 22.6 通过添加详细输出获得易用性体验

在第 19 章中，我们提到，很多脚本使用者喜欢看到脚本输出执行的进度，我们倾向于使用 `Write-Verbose` 而不是 `Write-Host` 产生这些信息。下面让我们来看一个实际例子。

我们在代码清单 22.6 中添加一些详细输出。

**代码清单 22.6 为 Get-DiskInventory.ps1 添加详细输出**

```
<#
.SYNOPSIS
Get-DiskInventory retrieves logical disk information from one or
more computers.
.DESCRIPTION
Get-DiskInventory uses WMI to retrieve the Win32_LogicalDisk
instances from one or more computers. It displays each disk's
drive letter, free space, total size, and percentage of free
space.
.PARAMETER computername
The computer name, or names, to query. Default: Localhost.
.PARAMETER drivetype
The drive type to query. See Win32_LogicalDisk documentation
for values. 3 is a fixed disk, and is the default.
.EXAMPLE
Get-DiskInventory -computername SERVER-R2 -drivetype 3
#>
[CmdletBinding()]
param (
   [Parameter(Mandatory=$True)]
   [Alias('hostname')]
   [string]$computername,
   [ValidateSet(2, 3)]
```

```
    [int]$drivetype = 3
)
Write-Verbose "Connecting to $computername"
Write-Verbose "Looking for drive type $drivetype"
Get-WmiObject -class Win32_LogicalDisk -computername $computername `
 -filter "drivetype=$drivetype" |
 Sort-Object -property DeviceID |
 Select-Object -property DeviceID,
     @{name='FreeSpace(MB)';expression={$_.FreeSpace / 1MB -as [int]}},
     @{name='Size(GB)';expression={$_.Size / 1GB -as [int]}},
     @{name='%Free';expression={$_.FreeSpace / $_.Size * 100 -as [int]}}
Write-Verbose "Finished running command"
```

下面尝试以两种方式运行该脚本。第一次尝试不会显示任何详细输出。

```
PS C:\> .\Get-DiskInventory -computername localhost
```

下面是第二次尝试，也就是我们希望显示详细输出。

```
PS C:\> .\Get-DiskInventory -computername localhost -verbose
```

*动手实验：当你自己动手尝试时就会发现脚本很棒——尝试运行我们展示的脚本，并查看两次运行的差别。*

太酷了，不是吗？当你想要详细输出时，就能获得详细输出——并且完全无须编写 -Verbose 参数的任何实现代码。当添加[CmdletBinding()]时，就可以无成本拥有详细输出。最妙的部分是，该标签还会激活脚本中所包含命令的详细输出！所以你使用的任何被设计可以产生详细输出结果的命令都会自动输出详细结果。该技术使得启用或禁用详细输出变得非常容易，相比 Write-Host 更加灵活。而且你无须通过操作 $VerbosePreference 变量就能将输出结果显示在屏幕上。

同时，注意在详细输出中我们如何使用 PowerShell 的双引号技巧：通过将变量（$computername）包含在双引号中，输出内容就可以包含变量的内容，所以我们可以看到 PowerShell 输出该变量的内容。

## 22.7 动手实验

**注意：**对于本次动手实验来说，你需要运行 PowerShell v3 或更新版本 PowerShell 的计算机。

本次动手实验需要你回忆起在第 12 章所学的内容，因为你需要将下述命令参数化，并将其存入脚本——正如你在第 21 章所做的那样。但这次我们还需要你将 -ComputerName 参数变为强制参数，并给它一个名称为 hostname 的别名。并且使得你的脚本可以在运行命令之前和之后显示详细输出。请记住，你必须将计算机名称参数化——这也是在本次案例中你唯一需要参数化的参数。

请在修改之前运行下述命令，从而确保下述命令可以在你的系统上运行。

```
get-wmiobject win32_networkadapter -computername localhost |
 where { $_.PhysicalAdapter } |
 select MACAddress,AdapterType,DeviceID,Name,Speed
```

重申一下，这里是你需要完成的任务列表。

- 确保该命令在修改之前可以正常运行。
- 将计算机名称参数化。
- 将-ComputerName 参数变为强制参数。
- 给予计算机名称参数一个别名 hostname。
- 至少一个基于注释的帮助，帮助内容是如何使用本脚本。
- 在命令运行之前和之后添加详细输出结果。
- 将脚本保存为 Get-PhyscialAdapters.ps1。

## 22.8 动手实验答案

```
#Get-PhysicalAdapters.ps1

<#
.Synopsis
Get physical network adapters
.Description
Display all physical adapters from the Win32_NetworkAdapter class.
.Parameter Computername
The name of the computer to check.
.Example
PS C:\> c:\scripts\Get-PhysicalAdapters -computer SERVER01
#>
[cmdletbinding()]
Param (
[Parameter(Mandatory=$True,HelpMessage="Enter a computername to query")]
[alias('hostname')]
[string]$Computername
)

Write-Verbose "Getting physical network adapters from $computername"

Get-Wmiobject -class win32_networkadapter -computername $computername |
 where { $_.PhysicalAdapter } |
 select MACAddress,AdapterType,DeviceID,Name,Speed

Write-Verbose "Script finished."
```