

11. Análisis de módulos

Wednesday, September 9, 2020 8:45 PM

Las funciones se pueden considerar unidades manejables de código, que nos brindan código más eficiente y fácil de leer. Sin embargo, las funciones no tienen porque existir solo en un script o en una sesión de PS.

Los módulos son grupos de funciones similares que se "empaquetan" juntos y son distribuidos para que otros los utilicen en múltiples scripts.

De forma simple, podemos decir que un módulo es un archivo de texto con extensión .psm1 y, de forma opcional, metadata.

Para usar un módulo, lo primero que se debe hacer es instalarlo. Después, cuando un comando dentro de ese módulo requiere ser utilizado, ese módulo tiene que ser importado en tu sesión. En PowerShell v3 y posteriores, los módulos son auto importados cuando un comando es referenciado.

Explorando módulos

Si quieres explorar los módulos importados en tu sesión actual, puedes usar el comando `Get-Verb`.

ModuleType	Version	Name	ExportedCommands
Manifest	3.1.0.0	Microsoft.PowerShell.Management	{Add-Computer, Add-Content, Checkpoint-Computer, Clear-Content...}
Manifest	3.1.0.0	Microsoft.PowerShell.Utility	{Add-Member, Add-Type, Clear-Variable, Compare-Object...}
Script	2.0.0	PSReadline	{Get-PSReadLineKeyHandler, Get-PSReadLineOption, Remove-PSReadLineKey...}

Los primeros 2 módulos listados se importan en cualquier sesión de PowerShell por default.

Para ver los comandos importados por cada módulo puedes usar `Get-Command`.

CommandType	Name	Version	Source
Function	ConvertFrom-SddlString	3.1.0.0	Microsoft.PowerShell.Utility
Function	Format-Hex	3.1.0.0	Microsoft.PowerShell.Utility
Function	Get-FileHash	3.1.0.0	Microsoft.PowerShell.Utility
Function	Import-PowerShellDataFile	3.1.0.0	Microsoft.PowerShell.Utility
Function	New-Guid	3.1.0.0	Microsoft.PowerShell.Utility
Function	New-TemporaryFile	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Add-Member	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Add-Type	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Clear-Variable	3.1.0.0	Microsoft.PowerShell.Utility

Existen funciones que se encuentran dentro de módulos o scripts que no pueden ser invocadas fuera de estos, por lo que son llamadas **funciones privadas** o **funciones de ayuda**.

Los comandos anteriores nos dieron acceso a todos los módulos instalados e importados a la sesión, pero también podemos visualizar todos los módulos disponibles agregando un parámetro al comando.

ModuleType	Version	Name	ExportedCommands
Binary	2.0.2.102	AzureADPreview	{Add-AzureADApplicationOwner, C...
Script	1.0.1	Microsoft.PowerShell.Operation.V...	{Get-OperationValidation, Invok...
Binary	1.0.0.1	PackageManagement	{Find-Package, Get-Package, Get...
Script	3.4.0	Pester	{Describe, Context, It, Should...
Script	1.0.0.1	PowerShellGet	{Install-Module, Find-Module, S...
Script	2.0.0	PSReadline	{Get-PSReadLineKeyHandler, Set-...

Directory: C:\Program Files\WindowsPowerShell\Modules			
ModuleType	Version	Name	ExportedCommands
Binary	2.0.2.102	AzureADPreview	{Add-AzureADApplicationOwner, C...
Script	1.0.1	Microsoft.PowerShell.Operation.V...	{Get-OperationValidation, Invok...
Binary	1.0.0.1	PackageManagement	{Find-Package, Get-Package, Get...
Script	3.4.0	Pester	{Describe, Context, It, Should...
Script	1.0.0.1	PowerShellGet	{Install-Module, Find-Module, S...
Script	2.0.0	PSReadline	{Get-PSReadLineKeyHandler, Set-...

Directory: C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules			
ModuleType	Version	Name	ExportedCommands
Manifest	1.0.0.0	AppBackgroundTask	{Disable-AppBackgroundTaskDiagr...
Manifest	2.0.0.0	AppLocker	{Get-AppLockerFileInfo...
Manifest	1.0.0.0	AppvClient	{Add-AppvClientConnectionGroup,

El comando anterior busca módulos en los siguientes path:

- C:\Windows\System32\WindowsPowerShell\1.0\Modules - **Módulos del sistema**. No agregues módulos a este folder. Es para módulos internos de PS.
- C:\Program Files\WindowsPowerShell\Modules - **Módulos de todos los usuarios**. Aquí puedes colocar todos los módulos que quieras disponibles para todos los usuarios.
- C:\Users<LoggedInUser>\Documents\WindowsPowerShell\Modules - **Módulos del usuario actual**. Aquí encontrarás todos los módulos que has creado o descargado y que están disponibles para el usuario actual.

Importando módulos

El PowerShell actual, al momento de llamar a una función, auto importa el módulo que la contiene. Aún así, si queremos importar manualmente un módulo podemos usar el comando `Import-Module`.

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> Import-Module -Name Microsoft.PowerShell.Management
PS C:\WINDOWS\system32> Import-Module -Name Microsoft.PowerShell.Management -Force
PS C:\WINDOWS\system32> Remove-Module -Name Microsoft.PowerShell.Management
PS C:\WINDOWS\system32> Get-Module

ModuleType Version Name ExportedCommands
---- -- -- -----
Manifest 3.1.0.0 Microsoft.PowerShell.Utility {Add-Member, Add-Type, Clear-Variable, ...
Script 2.0.0 PSReadline {Get-PSReadLineKeyHandler, Get-PSReadLineOption, Set-PSReadLineOption}

PS C:\WINDOWS\system32> Import-Module -Name Microsoft.PowerShell.Management -Force
PS C:\WINDOWS\system32> Get-Module

ModuleType Version Name ExportedCommands
---- -- -- -----
Manifest 3.1.0.0 Microsoft.PowerShell.Management {Add-Computer, Add-Content, Check-Computer, ...
Manifest 3.1.0.0 Microsoft.PowerShell.Utility {Add-Member, Add-Type, Clear-Variable, ...
Script 2.0.0 PSReadline {Get-PSReadLineKeyHandler, Get-PSReadLineOption}
```

Componentes de un módulo de PowerShell

Todo archivo con extensión `.psm1` puede ser un módulo pero, para que sea útil, debe tener funciones dentro y, aunque no es obligatorio, se recomienda que todas las funciones del módulo sean construidas bajo el mismo concepto. Un tip al respecto es que todas las funciones tengan el mismo Sustantivo (hablando de la nomenclatura Verbo-Sustantivo) y si requieres cambiar de sustantivo, tal vez es necesario pensar en otro módulo.

El manifiesto del módulo

Aunque no es obligatorio, se recomienda que por cada módulo se cree un archivo `.psd1`, el cual es un archivo de texto escrito en la forma de una hashtable, cuyos elementos describen la metadata sobre el módulo.

Puede crearse este archivo de forma manual, pero PowerShell permite crarlo con el comando:

```
PS > Import-Module -Path 'Liga completa al módulo'
```

Trabajando con módulos de terceros

Búsqueda de módulos

Cuando requieras resolver un problema, antes de escribir tu propio código, se recomienda buscar en la *PowerShell Gallery* (<https://www.powershellgallery.com/>), el cual es un repositorio de módulos y scripts de PowerShell.

Para buscar, guardar, instalar y hasta publicar módulos, podemos usar el comando `PowerShellGet`, si requieres averiguar las funciones disponibles ejecuta:

```
PS > Get-Command -Module PowerShellGet
```

Ejemplo de búsqueda de módulos:

```
PS > Find-Module -Name *VMware*
```

Este comando busca todos los módulos que tienen VMware en su nombre.

Instalando y desinstalando módulos

Ejemplo de instalación:

```
PS > Find-Module -Name VMware.PowerCLI | Install-Module
```

Ejemplos de desintalación:

```
PS > Uninstall-Module -Name VMware.PowerCLI
```

Creando módulos

1. Crear la carpeta donde lo guardaremos

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> mkdir 'C:\Program Files\WindowsPowerShell\Modules\Software'

Directory: C:\Program Files\WindowsPowerShell\Modules

Mode                LastWriteTime        Length Name
----                -              -          -
d----- 9/16/2020  5:03 PM           0 Software
```

2. Hacemos el archivo .psm1 en blanco:

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> Add-Content -Path 'C:\Program Files\WindowsPowerShell\Modules\Software\Software.psm1'

cmdlet Add-Content at command pipeline position 1
Supply values for the following parameters:
Value[0]:
```

3. Crea el manifiesto:

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> New-ModuleManifest -Path 'C:\Program Files\WindowsPowerShell\Modules\Software\Software.psd1' -Author 'Perla Viera' -RootModule Software.psm1 -Description 'Modulo de prueba con funcion simple'
```

4. Agrega las funciones a tu módulo:

```
Ejemplo15.ps1 Software.psm1
1 function Show-Message {
2     param([Parameter(Mandatory,valueFromPipeline)] [string] $msg,
3           [Parameter(Mandatory)] [ValidateSet(62,64)] [int]$grupo)
4     begin{
5         Write-Host "Bienvenido a mi función" }
6     process {
7         Write-Host "Mensaje:" $msg "`nAtte.- Grupo" $grupo }
8     end{
9         Write-Host "La función ha concluido con éxito" }
10 }
```

5. Prueba que se haya creado correctamente:

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> Get-Module -Name Software -List

Directory: C:\Program Files\WindowsPowerShell\Modules

ModuleType Version    Name                                ExportedCommands
----          --          --
Script       1.0        Software                           Show-Message
```

Referencias:

- Bertram, Adam (2020) . *PowerShell for SysAdmins*. "no starch press" <https://nostarch.com/powershellsysadmins>