Snap-Ins y módulos

Con PowerShell 1.0, la adición de funcionalidades y commandlets se realiza a través de los Snap-Ins. Por razones de compatibilidad, los Snap-Ins siempre están soportados en PowerShell v2. Sin embargo, los módulos están llamados a reemplazar progresivamente los Snap-Ins. Actualmente, la creación de «complementos» resulta mucho más flexible y fácil, y no está reservada a los desarrolladores como podían estar los Snap-Ins.

1. Los Snap-Ins: Add-PSSnapin, Remove-PSSnapin

Los Snap-Ins son archivos compilados (DLL) que permiten compartir un conjunto de commandlets considerados como extensiones de funcionalidad de PowerShell. En realidad, los Snap-Ins aportan el mismo servicio que los módulos, con la diferencia de que los módulos no son obligatoriamente ficheros compilados.

a. Listar los Snap-Ins instalados

Para conocer la lista de los Snap-Ins presentes en su máquina e importados en la sesión actual, teclee el comando **Get-PSSnapin**.

```
PS > Get-PSSnapin

Name        : Microsoft.PowerShell.Diagnostics
PSVersion   : 2.0
Descripción : Este complemento de Windows PowerShell contiene cmdlets de
Eventos de Windows y de contador de rendimiento.

Name        : Microsoft.WSMan.Management
PSVersion   : 2.0
Descripción : Este complemento de Windows PowerShell contiene cmdlets (como
Get-WSManInstance y Set-WSManInstance) que usa el host de Windows PowerShell
para administrar operaciones WSMan.

Name        : Microsoft.PowerShell.Core
PSVersion   : 2.0
Descripción : Este complemento de Windows PowerShell contiene cmdlets usados
para administrar los componentes de Windows PowerShell.

Name        : Microsoft.PowerShell.Utility
PSVersion   : 2.0
Descripción : Este complemento de Windows PowerShell contiene cmdlets de
utilidad que sirven para manipular datos.

Name        : Microsoft.PowerShell.Host
PSVersion   : 2.0
Descripción : Este complemento de Windows PowerShell contiene cmdlets (como
Start-Transcript y Stop-Transcript) proporcionados para su uso con el host
de la consola de Windows PowerShell.

Name        : Microsoft.PowerShell.Management
PSVersion   : 2.0
Descripción : El complemento Windows PowerShell contiene cmdlets de
administración para administrar los componentes de Windows.

Name        : Microsoft.PowerShell.Security
```

© Editions ENI – Tous droits réservés – Copie personnelle de Dpto. de INFORMATICA IES ZAIDIN-VERGELES

```
PSVersion   : 2.0
Descripción : Este complemento de Windows PowerShell contiene varios cmdlets
para la administración de la seguridad de Windows PowerShell.
```

Get-PSSnapin posee igualmente el switch -Registred. Cuando éste se especifica, permite listar los Snap-Ins disponibles del sistema que no han sido importados en la sesión actual. El resultado del comando no contiene los Snap-Ins necesarios para el funcionamiento de PowerShell.

Ejemplo:

```
PS > Get-PSSnapin -Registred
Name        : VMware.VimAutomation.Core
PSVersion   : 2.0
Descripción : This Windows PowerShell snap-in contains Windows PowerShell
cmdlets used to manage vSphere.
```

Esto significa que este Snap-In está instalado pero no importado en la sesión actual.

b. Importar un Snap-In

La importación se realiza con el comando **Add-PSSnapin**. Tomamos el ejemplo del Snap-in proporcionado por el editor de software de virtualización VMware. VMware proporciona un conjunto de commandlets PowerShell capaz de administrar los servidores VMware y sus máquinas virtuales asociadas. Disponible en el sitio de VMware bajo el nombre vSphere PowerCLI, este grupo de utilidades es un conjunto de ficheros DLL, que una vez instalados son importables como Snap-in mediante el comando **Add-PSSnapin**:

```
PS > Add-PSSnapin -Name VMware.VimAutomation.Core
```

Una vez el Snap-in se ha cargado, si sabemos que el nuevo juego de comandos contiene las letras «VM», podemos listar los comandos de la siguiente forma:

```
PS > Get-Command -Type cmdlet -Name *VM*
CommandType     Name                      Definition
-----------     ----                      ----------
Cmdlet          Add-VMHost                Add-VMHost [-Name] <String> [[...
Cmdlet          Add-VMHostNtpServer       Add-VMHostNtpServer [-NtpServe...
Cmdlet          Get-VM                    Get-VM [[-Name] <String[]>] [-...
Cmdlet          Get-VMGuest               Get-VMGuest [-VM] <VirtualMach...
Cmdlet          Get-VMHost                Get-VMHost [[-Name] <String[]>...
Cmdlet          Get-VMHostAccount         Get-VMHostAccount [[-Id] <Stri...
```

Pero podemos buscar una forma mejor, ya que si el comando no contiene los caracteres "VM" no nos lo listará con el comando propuesto.

c. Listar los comandos de un Snap-In

Un commandlet PowerShell se caracteriza por un determinado número de propiedades. Entre éstas, hay una en particular que indica el Snap-in de pertenencia de cada commandlet. Se trata de la propiedad `PSSnapin`.

Así, gracias a esta propiedad, que nos va a servir de filtro, vamos a poder listar el contenido de los comandos suministrados por un Snap-in determinado.

Ejemplo:

Lista de los comandos contenidos en los Snap-Ins que contienen la palabra «Diagnostics»

© Editions ENI – Tous droits réservés – Copie personnelle de Dpto. de INFORMATICA IES ZAIDIN-VERGELES

```
PS > Get-Command | Where {$_.PSSnapin.name -Match 'Diagnostics'}
```

O si conocemos el nombre exacto del Snap-In:

```
PS > Get-Command | Where {$_.PSSnapin.name -eq 'VMware.VimAutomation.Core'}
```

d. Descargar un Snap-In

Cuando el Snap-in ha dejado de ser útil, podemos eliminarlo de la sesión actual (sistema) usando el comando **Remove-PSSnapin**.

Ejemplo:

```
PS > Remove-PSSnapin -Name 'VMware.VimAutomation.Core'
```

2. Los módulos

Un módulo es una especie de contenedor (package) que agrupa los scripts, los comandos, las variables, los alias y las funciones. La ventaja es que los módulos se pueden compartir fácilmente para poder beneficiar a otros usuarios. La utilización de módulos permite crear scripts que se basan en otros scripts presentes en sus módulos; lo que evita tener que incluir el contenido de un script en el script en curso de desarrollo. Esto facilita su mantenimiento.

La idea del Team PowerShell es construir una gran comunidad de usuarios de PowerShell, de manera que ésta pueda intercambiar y compartir módulos a imagen de la Comunidad CPAN (*Comprehensive Perl Archive Network*) que seguro conocen bien los usuarios del lenguaje PERL.

Los módulos se presentan en forma de carpetas con uno o varios archivos. Estos directorios de módulos están dispuestos en la ubicación siguiente: %UserProfile%\Documents\WindowsPowerShell\Modules. En Windows 7, este directorio no existe por defecto, pero es posible crearlo utilizando la instrucción siguiente:

```
PS > New-Item -Type directory -Path $home\Documents\WindowsPowerShell\Modules
```

La ubicación `$env:UserProfile\Documents\WindowsPowerShell\Modules` hace referencia a la ubicación de los módulos aplicables a los usuarios. Para una aplicación del sistema y por lo tanto accesible a todos los usuarios, la ubicación es la siguiente: `$env:windir\System32\WindowsPowerShell\v1.0\Modules`. Hay que señalar, la existencia de la variable de entorno `$PSModulePath` que reagrupa estas dos ubicaciones.

Con Windows Server 2008 R2, Windows PowerShell viene provisto de varios módulos preinstalados. Basta con utilizar el asistente «Añadir funcionalidad» del gestor de servidor para instalar automáticamente los módulos de funcionalidad que seleccione. Pero si usted recibe un módulo en formato de carpeta conteniendo los archivos, basta simplemente con colocarlo en el directorio módulos para poder importarlo a Windows PowerShell.

Existen varios tipos de módulo, estos últimos se describen a continuación.

Tipo de módulo	Descripción
Script	Un módulo de tipo **Script** es un módulo compuesto por un archivo (.psm1) que contiene código PowerShell. Se trata del tipo más común.
Binary	Un módulo de tipo **Binary** es un módulo que contiene código compilado (archivo .dll).
Manifest	Un módulo de tipo **Manifest** está compuesto por un archivo (.psd1) que contiene diversa información relativa a un módulo como su modo de ejecución, el autor, el número de versión, etc.

© Editions ENI – Tous droits réservés – Copie personnelle de Dpto. de INFORMATICA IES ZAIDIN-VERGELES

Dynamic	Un módulo de tipo **Dynamic** es un módulo que no se encuentra almacenado en disco. Se trata de un módulo de corta duración y por consiguiente no está visible al utilizar el commandlet `Get-Module` (véase a continuación).

a. Listar los módulos

Con el fin de conocer los módulos ya importados, PowerShell v2 está dotado del commandlet `Get-Module`.

`Get-Module` posee los parámetros siguientes:

Parámetro	Descripción
`All <Switch>`	Obtiene todos los módulos exportados para todos los módulos disponibles.
`ListAvailable <Switch>`	Obtiene todos los módulos importables en la sesión.
`Name <String[]>`	Obtiene únicamente el o los módulos especificado(s).

Utilizado solo, `Get-Module` devuelve la lista de los módulos importados en la sesión en curso:

```
PS > Get-Module
```

Si desea listar los módulos instalados (en el directorio Modules) pero que no estén importados, tendrá que utilizar el parámetro `-listAvailable`.

Por ejemplo en Windows 7:

```
PS > Get-Module -listAvailable
ModuleType Name                     ExportedCommands
---------- ----                     ----------------
Manifest   AppLocker                {}
Manifest   BitsTransfer             {}
Manifest   PSDiagnostics            {}
Manifest   TroubleshootingPack      {}
```

Observamos que por defecto están instalados cuatro módulos de tipo «MANIFEST» en Windows 7 (se trata de cuatro módulos ubicados en la ruta `$env:windir\System32\WindowsPowerShell\v1.0\Modules`).

He aquí una tabla recapitulativa de los módulos instalados por defecto en los sistemas operativos Windows 7 y Windows Server R2:

Windows 7	Windows Server 2008 R2	Descripción de los módulos	Comandos disponibles por módulo
AppLocker	AppLocker	Impedir la ejecución de programas no autorizados	Get-AppLockerPolicy Get-AppLockerFileInformation Test-AppLockerPolicy New-AppLockerPolicy Set-AppLockerPolicy
BitsTransfer	BitsTransfer	Transferencia inteligente de archivos en segundo	Start-BitsTransfer Remove-BitsTransfer

© Editions ENI – Tous droits réservés – Copie personnelle de Dpto. de INFORMATICA IES ZAIDIN-VERGELES

		plano	Resume-BitsTransfer Get-BitsTransfer Add-BitsFile Set-BitsTransfer Complete-BitsTransfer Suspend-BitsTransfer
PSDiagnostics	PSDiagnostics	Ayuda al diagnóstico de Windows	Enable-PSTrace Enable-WSManTrace Start-Trace Disable-PSWSManCombinedTrace Disable-PSTrace Disable-WSManTrace Get-LogProperties Stop-Trace Enable-PSWSManCombinedTrace Set-LogProperties
TroubleShootingPack	TroubleShootingPack	Ayuda a la resolución de problemas	Get-TroubleshootingPack Invoke-TroubleshootingPack
	ADRMS	Microsoft Windows Active Directory Rights Management Services Module	Uninstall-ADRMS Update-ADRMS Install-ADRMS
	BestPractices	Best Practices Module	Get-BpaModel Set-BpaResult Invoke-BpaModel Get-BpaResult
	ServerManager	Server Manager Module	Remove-WindowsFeature Get-WindowsFeature Add-WindowsFeature

b. Importar un módulo

Cuando un módulo está correctamente instalado en el directorio módulo, será necesario importarlo a continuación. Esta operación se efectuará con el commandlet `import-module <nombre módulo>`.

Ejemplo:

```
PS > Import-Module BitsTransfer
```

© Editions ENI – Tous droits réservés – Copie personnelle de Dpto. de INFORMATICA IES ZAIDIN-VERGELES

El comando `Import-Module` importa los módulos a su sesión de usuario de PowerShell. Para que la importación de los módulos sea efectiva para el conjunto de usuarios, se recomienda añadir el comando `Import-Module` al perfil de máquina de PowerShell (véase la sección Personalizar PowerShell modificando su perfil en este capítulo).

Una vez esté instalado el módulo, el commandlet `Get-Module`, visto anteriormente, le confirmará que el módulo está correctamente importado.

```
PS > Get-Module

ModuleType Name         ExportedCommands
---------- ----         ----------------
Manifest   BitsTransfer {Start-BitsTransfer, Remove-BitsTransfer, Resume-...
```

c. Listar los comandos de un módulo

Para conocer los comandos introducidos por el módulo importado, consulte la propiedad ExportedCommands como se muestra a continuación:

```
PS > (Get-Module BitsTransfer).ExportedCommands

Name                           Value
----                           -----
Start-BitsTransfer             Start-BitsTransfer
Remove-BitsTransfer            Remove-BitsTransfer
Resume-BitsTransfer            Resume-BitsTransfer
Get-BitsTransfer               Get-BitsTransfer
Add-BitsFile                   Add-BitsFile
Set-BitsTransfer               Set-BitsTransfer
Complete-BitsTransfer          Complete-BitsTransfer
Suspend-BitsTransfer           Suspend-BitsTransfer
```

Para listar los comandos introducidos por un módulo, lo más sencillo es cargarlo previamente con el comando `Import-Module`. Sin embargo, también puede explorar la estructura de archivos y directorios que componen los módulos si usted no desea cargarlos.

Para importar en una operación todos los módulos disponibles en su sistema puede usar el comando siguiente: `Get-Module -ListAvailable | Import-Module`

d. Descargar un módulo

El commandlet `Remove-Module` permite suprimir el módulo. El usuario no puede utilizarlo, pero no obstante, no se ha suprimido del sistema.

```
PS > Remove-Module BitsTransfer
```

Para suprimir en una operación todos los módulos importados de su sesión puede usar el comando siguiente: `Get-Module | Remove-Module`

© Editions ENI – Tous droits réservés – Copie personnelle de Dpto. de INFORMATICA IES ZAIDIN-VERGELES