PowerShell y objetos COM: ¿Siguen siendo útiles? Comparativa práctica con FileSystemObject

Uno de los aspectos más interesantes de PowerShell es su capacidad para interactuar con tecnologías heredadas de Windows, incluyendo los tradicionales objetos COM (Component Object Model). Aunque PowerShell está basado en .NET y ofrece cmdlets modernos para prácticamente cualquier tarea administrativa, en algunos escenarios los objetos COM siguen ofreciendo ventajas, especialmente en rendimiento.

Creando carpetas mediante FileSystemObject

Uno de los objetos COM más utilizados históricamente es **Scripting.FileSystemObject**, ampliamente empleado en VBScript para gestionar archivos y directorios.

Para crear una carpeta utilizando este objeto desde PowerShell podemos usar el siguiente código:

```
$fso = New-Object -ComObject Scripting.FileSystemObject
$fso.CreateFolder("C:\fso2")
```

La primera línea crea una instancia del objeto COM y la almacena en la variable $fso. Posteriormente se invoca el método CreateFolder() para crear el directorio.

La alternativa nativa de PowerShell

PowerShell incorpora cmdlets específicos para trabajar con el sistema de archivos, por lo que la misma tarea puede

realizarse de forma mucho más sencilla:

```
New-Item -Path C:\fso2 -ItemType Directory
```

Incluso existe una forma aún más breve:

```
mkdir C:\fso2
```

¿Cuál es más rápido?

Podemos medir el tiempo de ejecución mediante el cmdlet Measure-Command.

Medición usando FileSystemObject

```
Measure-Command {
    $fso = New-Object -ComObject Scripting.FileSystemObject
    $fso.CreateFolder("C:\fso2")
}
```

Resultado aproximado:

```
TotalMilliseconds : 1.5
```

Medición usando New-Item

```
Measure-Command {
    New-Item -Path C:\fso2 -ItemType Directory
}
```

Resultado aproximado:

```
TotalMilliseconds : 3.4
```

A simple vista parece que FileSystemObject es más rápido. Sin embargo, sacar conclusiones basándose en unos pocos milisegundos no es una buena práctica.

Realizando una prueba más realista

Para obtener resultados más fiables podemos repetir la

operación miles de veces.

Script usando cmdlets de PowerShell

```
for ($i = 1; $i -le 10000; $i++) {
    New-Item -Path C:\fso2
             -Name "Folder_$i"
             -ItemType Directory
}
```

Guardamos el código como:

CreateFolders.ps1

Y medimos su ejecución:

```
Measure-Command {
    & "C:\FSO\CreateFolders.ps1"
}
```

Tiempo aproximado:

40 segundos

La misma prueba utilizando COM

```
$fso = New-Object -ComObject Scripting.FileSystemObject

for ($i = 1; $i -le 10000; $i++) {
    $fso.CreateFolder("C:\fso2\Folder_$i")
}
```

Guardamos el script como:

CreateFoldersFSO.ps1

Y ejecutamos:

```
Measure-Command {
    & "C:\FSO\CreateFoldersFSO.ps1"
}
```

Tiempo aproximado:

7 segundos

Conclusión de la prueba

Los resultados muestran que para una creación masiva de directorios el objeto COM FileSystemObject puede ser varias veces más rápido que el cmdlet nativo New-Item.

Método	Tiempo aproximado
New-Item	40 segundos
FileSystemObject	7 segundos

Esto demuestra una realidad importante: los cmdlets de PowerShell suelen ser más cómodos y legibles, pero no siempre son la opción más rápida.

Accediendo a otros objetos COM

PowerShell permite utilizar multitud de objetos COM instalados en Windows.

Por ejemplo, podemos obtener la ruta del escritorio del usuario mediante el objeto WScript.Shell:

```
$wshShell = New-Object -ComObject WScript.Shell
$wshShell.SpecialFolders.Item("Desktop")
```

Salida:

```
C:\Users\Usuario\Desktop
```

También puede hacerse en una sola línea:

```
(New-Object -ComObject WScript.Shell).SpecialFolders.Item("Desktop")
```

Aunque funcionalmente es equivalente, suele ser recomendable almacenar primero el objeto en una variable para mejorar la legibilidad del código.

Migrando scripts BAT a PowerShell

Muchas organizaciones siguen utilizando antiguos archivos .bat.

Por ejemplo:

```
md C:\fso2
copy C:\fso\* C:\fso2 > C:\fso2\auditLog.txt
C:\fso2\auditLog.txt
```

La versión moderna en PowerShell sería:

```
New-Item -Path C:\fso2 -ItemType Directory

Copy-Item C:\fso\* C:\fso2

Get-Content C:\fso2\auditLog.txt
```

O incluso:

```
mkdir C:\fso2
copy C:\fso\* C:\fso2
notepad C:\fso2\auditLog.txt
```

Recomendaciones finales

- Utiliza cmdlets nativos de PowerShell siempre que sea posible.
- Recurre a objetos COM cuando necesites compatibilidad con tecnologías antiguas.
- Mide siempre el rendimiento antes de optimizar.
- No tomes decisiones basadas en diferencias de unos pocos milisegundos.
- En tareas masivas, los objetos COM todavía pueden ofrecer ventajas significativas.

Aunque los objetos COM son una tecnología veterana, PowerShell sigue proporcionando una excelente integración con ellos,

permitiendo reutilizar herramientas y componentes desarrollados hace décadas sin necesidad de reescribir completamente el código.