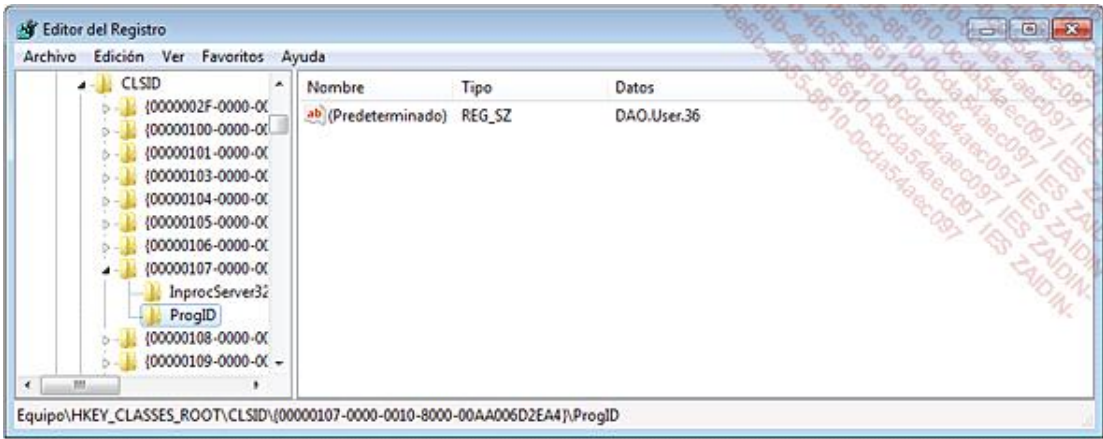


Manipular objetos COM

La utilización de los objetos COM no presenta gran complicación en sí misma. Pero para poder utilizarlos, hará falta saber si están disponibles en su puesto de trabajo. Para ello, habrá que buscar en el registro de Windows, y más concretamente en la ruta HKey_Clases_Root\CLSID. Como puede ver en la figura siguiente, los CLSID son los identificadores codificados en hexadecimal que permiten identificar de una forma única cada objeto COM. Poco inteligibles para los humanos, los CLSID poseen lo que se denomina un ProgID (Identificador de programación). Este ProgID que no es más que una representación en forma de cadena de caracteres de un objeto COM está estructurado de manera que exprese lo más explícitamente posible el rol del objeto COM. Así, todos los ProgID están compuestos de la siguiente forma: <Programa>.<componente>.<número de versión>. Observe que es más fácil utilizar Word.Application.1 que {00000107-0000-0060-8000-00AA006D2EA4}



Vista del ProgID en el editor del Registro

1. Buscar un objeto


Por supuesto, en lugar de recorrer gráficamente todas las referencias contenidas en el registro de Windows para encontrar un ProgID, es mucho más interesante automatizar una búsqueda mediante una función PowerShell. La función Get-ProgID que le proponemos, responde a esta necesidad. Recorre el registro de Windows, compara cada entrada y muestra la lista de ProgID cuyo nombre corresponde a la búsqueda realizada.

```
Function Get-ProgID
{
param([string]$ProgID = '.')
Get-ChildItem REGISTRY::HKey_Clases_Root\clsid\*\progid |
Where-Object {$_.GetValue('') -match $ProgID} |
Foreach-Object{$_.GetValue('')}
}
```

Una vez realizada esta función, podrá recuperar rápidamente el nombre de un ProgID disponible en su sistema operativo Windows. El ejemplo siguiente nos muestra los resultados devueltos tras una búsqueda de un identificador de programa en cuyo nombre aparezca la palabra «Explorer»:


```
PS > Get-ProgID Explorer

InternetExplorer.Application.1
Shell.Explorer.2
Groove.WorkspaceExplorerApplication
Shell.Explorer.1
```

 La obtención de los ProgID también es posible llamando a la clase WMI Win32_ClassicCOMClassSetting. Ejemplo de la función Get-ProgID realizada utilizando la clase WMI:

```
Function Get-ProgID
{
param([string]$ProgID = '.')

Get-WmiObject Win32_ClassicCOMClassSetting |
Where-Object {$_.ProgId -match $ProgID} |
Select-Object ProgID,Description
}
```

 Observe que esta vez también mostramos la propiedad descripción de cada objeto contenido en la clase. Ejemplo de utilización con la palabra clave «Explorer»:

```
PS > Get-ProgID Explorer

ProgID                                Description
-----                                -
InternetExplorer.Application.1       Internet Explorer(Ver 1.0)
Shell.Explorer.2                     Microsoft Web Browser
Groove.WorkspaceExplorerApplication   Groove WorkspaceExplorerApplication
Shell.Explorer.1                     Microsoft Web Browser Version 1
```

2. Crear un objeto

Una vez que hemos encontrado el objeto COM correspondiente a nuestras necesidades, la instanciación se realiza con el commandlet **New-Object** que, si recordamos, también permite crear objetos .NET. Este commandlet posee realmente un doble uso. Pero, en esta ocasión, para crear un objeto COM, es necesario utilizar el parámetro **-ComObject** y proporcionar el ProgID que permitirá identificar el objeto COM deseado. Utilizado con objetos COM, el commandlet **New-Object** posee dos parámetros (a parte de los parámetros comunes), veamos el detalle:

Parámetro	Descripción
-ComObject <String> 0 -Com <String>	Especifica al commandlet el ProgID del objeto COM.
-Strict	Devuelve un mensaje de error si el objeto COM pasado por parámetro es en realidad un objeto .NET «wrappeado».

Los objetos COM no se escapan a la regla (salvo en casos particulares), sus propiedades y métodos están disponibles con un simple **Get-Member**. Ejemplo con la creación de un objeto COM `wscript.shell` (representación del entorno de ejecución WSH, ver al final de este capítulo).

```
PS > $WShell = New-Object -ComObject WScript.Shell
PS > $WShell | Get-Member

TypeName: System.__ComObject#{41904400-be18-11d3-
a28b-00104bd35090}

Name                MemberType Definition
----                -
AppActivate         Method      bool AppActivate (Variant, Variant)
CreateShortcut       Method      IDispatch Create Shortcut (string)
```

Exec	Method	IWshExec Exec (string)
ExpandEnvironmentStrings	Method	string Expand EnvironmentStrings ...
LogEvent	Method	bool LogEvent (Variant, string, st...
Popup	Method	int Popup (string, Variant, Varian...
RegDelete	Method	void RegDelete (string)
RegRead	Method	Variant RegRead (string)
RegWrite	Method	void RegWrite (string, Variant, Var...
Run	Method	int Run (string, Variant, Variant)
SendKeys	Method	void SendKeys (string, Variant)
Environment	Parameter...	IWshEnvironment Environm...
CurrentDirectory	Property	string CurrentDirectory () {get} {set}
SpecialFolders	Property	IWshCollection SpecialFolders () {get}