

WScript.Shell

A classe WScript.Shell é f para qualquer utilizador de escrita do Windows VBScript scripts de gerenciamento. Ele contém uma coleção de métodos úteis para a criação de scripts no Windows.

$wshell = New-Object -com WScript.Shell

$wshell | Get-Member

Veja alguns exemplos de métodos e propriedades;

|  |  |  |
| --- | --- | --- |
| Tipo | Info | Descrição |
| AppActivate | Method | bool AppActivate (Variant, Variant) |
| CreateShortcut | Method | Dispatch CreateShortcut (string) |
| Exec | Method | WshExec Exec (string) |
| ExpandEnvironmentStrings | Method | string ExpandEnvironmentStrings (string) |
| LogEvent | Method | bool LogEvent (Variant, string, string) |
| Popup | Method | int Popup (string, Variant, Variant, Variant) |
| RegDelete | Method | void RegDelete (string) |
| RegRead | Method | Variant RegRead (string) |
| RegWrite | Method | void RegWrite (string, Variant, Variant) |
| Run | Method | int Run (string, Variant, Variant) |
| SendKeys | Method | void SendKeys (string, Variant) |
| Environment | Property | IWshEnvironment Environment (Variant) {get} |
| CurrentDirectory | Property | string CurrentDirectory () {get} {set} |
| SpecialFolders | Property | IWshCollection SpecialFolders () {get} |

Exemplos

Um dos que eu mais gosto de usar no WScript.Shell é o Popup

$wshell.Popup("O PowerShell É Muito loco mesmo")

Você executar aplicativos usando o WScript.Shell. Como por exemplo a calculadora do Windows.

$wshell.Run("Calc")

Outro método interessante é o SendKey.

$wshell = New-Object -com WScript.Shell

$wshell.Run("Notepad")

$wshell.AppActivate("Notepad")

Start-Sleep 1

$wshell.SendKeys("Interessante!!!")

Links

Neste link abaixo é possível encontrar uma função onde podemos usar o recurso de popup onde é possível criar caixas de dialogo bem incrementadas.



<https://powershell.org/tag/wscript-shell/>