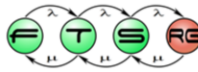


Szkriptelés alapok (PowerShell)

Micskei Zoltán



Utolsó módosítás: 2014. március 3.

DEMO Kedvcsináló: Facebook képek

- Le szeretnénk tölteni a FB ismerőseink profilképét
- Könnyen automatizálható feladat
 - FB-nak van API-ja, később is kellhet még ez
- Biztos van rá freeware/shareware, de
 - Megbízható? Azt csinálja, ami nekünk kell?
 - Informatikusok vagyunk, meg tudjuk írni 😊
- Szkript <10 perc alatt elkészülhet
 - FB API: <https://developers.facebook.com/docs/reference/api/>
 - PowerShell: Invoke-RestMethod, Invoke-WebRequest



2



Ez egy gyorsan összedobott megoldás, majd a gyakorlati anyagok közé bekerül egy szép változat is:

\$AccessToken = "XXXXXX"

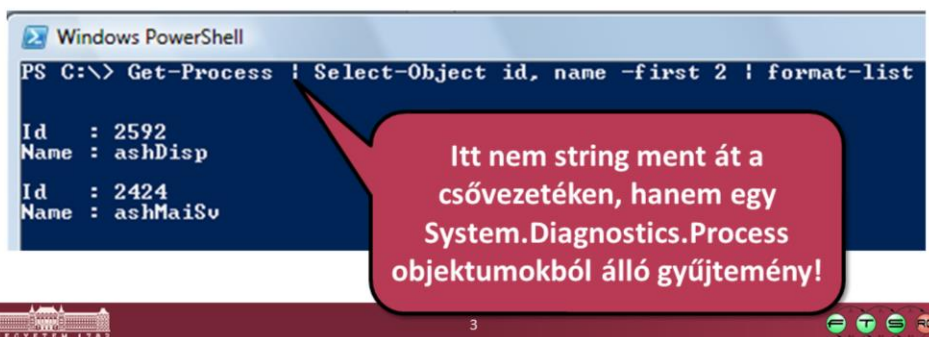
\$friendsUri = "https://graph.facebook.com/me/friends?access_token=" +
\$AccessToken

\$friends = Invoke-RestMethod -Uri \$friendsUri -ErrorAction Stop

```
$friends.data | % {  
    $pictureUri = "https://graph.facebook.com/" + $_.id + "/picture?type=large"  
    Invoke-WebRequest -Uri $pictureUri -OutFile ($_.name + ".jpg") -ErrorAction  
Continue  
}
```

PowerShell

- Új szkript környezet a Windowsban (2006-)
- bash/Perl/stb. tapasztalatok alapján
- Újdonság:
 - teljesen objektumorientált,
 - .NET-tel integrált



```
Windows PowerShell
PS C:\> Get-Process | Select-Object id, name -first 2 | format-list

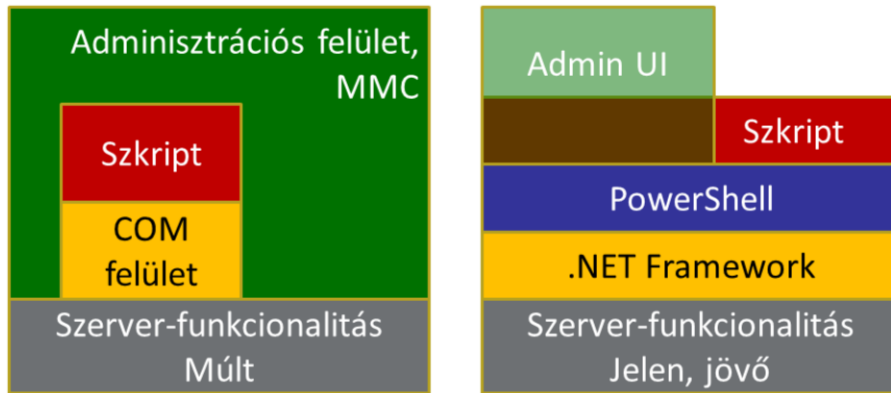
Id      : 2592
Name    : ashDisp
Id      : 2424
Name    : ashMaiSu
```

Itt nem string ment át a csővezetéken, hanem egy System.Diagnostics.Process objektumokból álló gyűjtemény!

Get-Process | Select-Object id, name -first 2 | format-list

Miért fontos a PowerShell?

Ez az új automatizálási motor Windowson



A Microsoft korábban a szervertermékeiben a funkcionalitás nagy részét a GUI-n keresztül tette elérhetővé. Ez a PowerShell megjelenésével megváltozott, azót az az irány, hogy a teljes funkcionitás PowerShellen keresztül érhető el, és ennek a gyakran használt részeit kivezetik a GUI-ra (de a háttérben az is tulajdonképpen PowerShell parancsokat hív).

Ábra forrása: Soós Tibor, Windows Server 2008 { PowerShell },
<http://www.microsoft.com/hun/dl.aspx?id=45d50c9b-c4b5-440c-8eb2-cd6e01a79464>

Milyen alkalmazás nyújt PowerShell API-t?

- Összes újabb MS szerver
 - Exchange, SQL Server, System Center Operations Manager, System Center VMM, IIS...
- Fejlesztő környezet:
 - Visual Studio, [NuGet](#) package manager
- VMware:
 - [PowerCLI](#) – virtualizációs környezet automatizálása
- Dell AppAssure, HP iLO, AWS PS modul
- ...



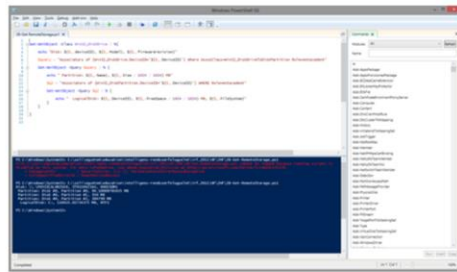
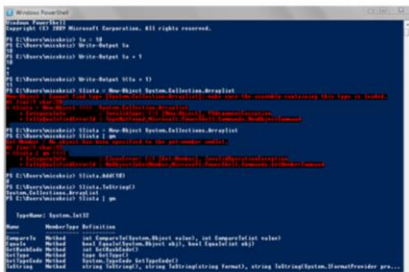
5



- NuGet: <http://docs.nuget.org/docs/reference/package-manager-console-powershell-reference>
- VMware PowerCLI, <http://www.vmware.com/go/powercli>
- AWS: <http://aws.amazon.com/powershell/>

Elindulás

- Powershell **4.0** letöltése
 - Windows Management Framework kiegészítés része
 - Windows 8.1-en fent van, de elérhető Windows 7-re is
- PowerShell ISE
 - GUI szerkesztő, debugger



Windows Management Framework 4.0, <http://www.microsoft.com/en-us/download/details.aspx?id=40855>

A korábbi PowerShell verziók elérhetőek Windows XP-re is, de a 3.0-tól kezdve már nem. Ennek megfelelően Windows XP használata nem javasolt a házi feladatok megoldása során.

PowerShell felhasználása

- Interaktív mód
 - PowerShell konzol
- Szkript készítése és meghívása
 - **ps1** kiterjesztésű fájl
- (PowerShell függvények, modulok készítése)

Figyelem! Szkriptnyelv!

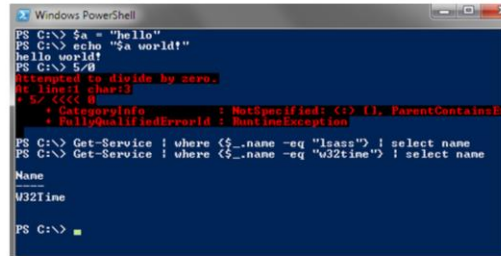
Célok:

- Utasításonként értelmezhető
- Fájl útvonalak könnyen kezelhetők
(ne kelljen escape szekvenciát használni)
- Tömör legyen
 - `ls $home *.txt | ? {$_ .length -gt 100}`
- Könnyű legyen külső programot meghívni
- Siker esetén nincs visszajelzés általában

Emiatt néhol elsőre furcsa a szintaktika!

PowerShell konzol

- PowerShell konzol:



```
Windows PowerShell
PS C:\> $a = "hello"
PS C:\> echo "a world!"
hello world!
PS C:\> 5/0
Attempted to divide by zero.
At line:1 char:1
+ 5/0 <<< 0
+ ~~~~~
+ CategoryInfo          : NotSpecified: (:) [], ParentContainsErrorRecordsException
+ FullyQualifiedErrorId : RuntimeException

PS C:\> Get-Service | where {$_.name -eq "lsass"} | select name
PS C:\> Get-Service | where {$_.name -eq "u32time"} | select name
Name
-----
u32time
PS C:\>
```

- Legfontosabb billentyű: **TAB**
 - Automatikus kiegészítés: cmdlet, paraméter, változók...
 - SHIFT + TAB: visszafelé lépked
- F7 – parancs előzmény
- ESC – aktuális sor törlése

PowerShell alapok

- **cmdlet**
 - Általában Ige-Főnév elnevezés
 - Adott funkciót megvalósító „parancs”
 - (háttérben: Cmdlet .NET osztály leszármazottai)
- Alap parancsokhoz megszokott aliasok
 - Pl. cp, copy -> Copy-Item
- Nyelv nem kis/nagybetű érzékeny



10



Milyen igéket használnak / használjunk a cmdletek nevében:
MSDN. „Approved Verbs for Windows PowerShell Commands”, URL:
[http://msdn.microsoft.com/en-us/library/ms714428\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ms714428(v=vs.85).aspx)

Cmdlet paraméterek

■ Cmdlet paraméterek:

- Ezekre is működik a TAB!
- Lehet kötelező vagy opcionális
- Nevesített, pozícionális

```
Get-ChildItem .\subdir -Filter *.txt -Recurse
```

-Path paraméter,
pozícionális (1.)

Nevesített, értékkel
rendelkezik

Nevesített,
switch típusú

Segítség

- Súlyó *cmdletek*:
 - **Get-Command**: parancsok listázása
 - Szűrés pl.: **Get-Command** -Noun csv
 - **Get-Help**: súlyó, paraméter leírás, példák
 - **Get-Help** Get-ChildItem -examples
 - **Get-Help** about_*
- Grafikus választás: **Show-Command**
- Súlyó frissíthető (**Update-Help**)



12



Súlyó a weben:

- Windows PowerShell Core Cmdlet Help Topics, <http://technet.microsoft.com/en-us/library/hh849695.aspx>
- Windows PowerShell Core About Help Topics, <http://technet.microsoft.com/en-us/library/hh847856.aspx>

DEMO PowerShell alapok

- Get-Command
- man Get-Command -full
 - Get-Command -Verb get
- Show-Command
- Get-ChildItem
- Get-ChildItem | Get-Member
- (Get-ChildItem).Count
- Külső program meghívása:
 - ipconfig /all

Powershell változók

- Változó: `$name`
- Típusok kezelése
 - `$a = "Hello"` # `System.String`
 - `$a + 1` # `Hello1` (aut. konverzió)
 - `[int] $year` # explicit megadás
- Lehet bármilyen .NET objektumot létrehozni:
 - `$list = New-Object System.Collections.ArrayList`
- Mit csinálhatok egy változóval?
 - `Get-Member -InputObject $list`
- Escape szekvenciák: ``t`, ``n` ...



14



Figyeljünk, hogy az escape karakter a backtick (magyar billentyűzeten az AltGr+7)

- ``t`: tabulátor
- ``n`: új sor
- ``$`: `$`
- Továbbiak: `about_Escape_Characters`

Változó behelyettesítések

- Hasonló a Bash-hez

```
$s = "world"
```

```
"Hello $s" # behelyettesít
```

```
'Hello $s' # nem helyettesít be
```

- Kiértékelés kikényszerítése

```
$a = 1
```

```
Write-Output "$a + 1" # 1 + 1
```

```
Write-Output "$($a + 1)" # 2
```

DEMO PowerShell változók

- Expression mód:

```
2 + 2
```

```
3 * 1024Gb
```

```
"hi " + "powershell"
```

- Változók használata:

```
$a = "scripting"; $a.GetType()
```

```
gm -InputObject $a
```

```
$a.Replace("s", "sz")
```

```
echo "hello $a"
```

```
echo "`$a értéke: $a"
```



Tömb, hash tábla

- Tömb létrehozása:
 - `$numbers1 = @()` # üres tömb
 - `$numbers2 = 1, 2, 5`
- Elemre hivatkozás:
 - `$numbers2[0]` # 0-tól indexelődik
- Hash tábla:
`$p = @{"MZ" = 3; "TD" = 4}`
`$p["MZ"]`

Csővezeték (pipe) kezelése

- Pipeline: legfontosabb művelet (jele: |)
`Get-Service | Format-List`
- Rendezés és kiválasztás:
`Get-Service | Select-Object name, status
-first 10 | Sort-Object Status`
- Művelet elvégzése minden elemen (jele: %):
`Get-Process | Foreach-Object {Write-Output $_.Name}`
\$_: aktuális elem
- Szűrés (jele: ?):
`Get-Process | Where-Object {$_.Id -eq 4}`



18



A csővezetékben mindig típusos, strukturált objektumok utaznak, így sokkal könnyebb kezelni őket.

A csővezeték hatékonyan van implementálva, érdemes használni.

DEMO PowerShell parancsok

- Kiválasztás, szűrés, rendezés
 - `Get-ChildItem | select Name, CreationTime`
 - `Get-ChildItem | where {$_.Name -like "D*"}`
 - `Get-ChildItem | Sort-Object LastWriteTime -Descending`
- Művelet elvégzése minden elemen:
 - `Get-ChildItem | % {$_.Name.Split("-")[0]}`
- Összesítés számolása
 - `Get-ChildItem C:\Windows\system32 -Filter *.dll | Measure-Object -Maximum -Property length`



Vezérlési szerkezetek

- C#-ból ismerős szerkezetek:
 - if, switch, foreach, while...
 - Sokszor kiváltható pipe segítségével
 - Pl. for ciklus helyett: `1..10 | % {echo $_}`
- Összehasonlítás:
 - -eq: egyenlő (equal)
 - -lt: kisebb mint (less than)
 - ...
- Logikai operátorok:
 - -and, -or, -not

Egyszerű szkript sablon

```
<#  
.SYNOPSIS  
Writes out a greeting message  
  
.PARAMETER Hello  
Message to write out  
  
.NOTES  
Author: Micskei Zoltan, 2014.03.03.  
#>  
  
param(  
    [Parameter(Mandatory=$true)][string] $Hello  
)  
  
Write-Output $hello
```

Fejkomment

Paraméter
megadás

Utasítások



21



Érdemes ilyen stílusú fejkommentet használni, mert így a Get-Help tud bővebb információt megadni majd később a saját szkriptünkről is.

A fejkommentben használható elemek listája és további tanácsok:
Get-Help about_Comment_Based_Help

Paraméterek ellenőrzése

- **Param** kulcsszó, megadható:
 - Típus, alapérték, kötelezőség, pozíció, ellenőrzés
- Test-Param.ps1:

```
param(  
    [Parameter(Mandatory=$true,Position=0)][string] $Message,  
    [ValidateRange(0,10)][Parameter(Mandatory=$false)][int]  
    $Number = 2,  
    [switch] $Flag  
)
```
- ParamTest meghívására példák:
 - .\Test-Param.ps1 "hello" -Flag
 - .\Test-Param.ps1 -Number 3 -Message "hello"
 - ...



Bővebben a felhasználható tulajdonságok és ellenőrzések:
- PowerShell help: `about_Functions_Advanced_Parameters`,
<http://technet.microsoft.com/en-us/library/hh847743.aspx>

Fontosabb cmdlet-ek

- **Import-Csv** CSV fájl importálása
- **Get-Content** Fájl tartalmát beolvasni
- **Get-ChildItem** Gyerekelemek lekérése
- **New-Item** Új elem (fájl, registry kulcs...)
- **Write-Output** Szöveg kiírása
- **Select-String** Szöveg keresése

- Valamint a teljes .NET Framework !
 - Pl. szöveg manipuláció -> System.String metódusai

DEMO PowerShell scriptek

- Használjunk PowerShell ISE-t
 - Breakpoint, változók értékei, kiíratás...
- Írjunk egy scriptet, ami lekérdezi, hogy hány svchost.exe fut, és hogy a legtöbb memóriát foglaló az 10 MB-nál többet használ-e!



24



Egy lehetséges megoldás:

```
$svchosts = Get-Process | Where-Object {$_.ProcessName -eq "svchost"}
Write-Output "Selected $($svchosts.Length) svchost processes"

if (($svchosts | Measure-Object -property WS -maximum).Maximum -gt 10MB)
{
    Write-Output "Too much memory consumed.."
}
else
{
    Write-Output "Memory ok"
}
```

Vagy powershellesebben:

```
(Get-Process | Where-Object {$_.ProcessName -eq "svchost"} | Measure-Object -
property WS -maximum).Maximum -gt 10MB
```


.NET osztálykönyvtár használata

- Statikus metódus meghívása:

- `[nevter.osztaly]::metodus(param1,param2...)`
- `[System.Math]::Tan(3.14)`

- Új objektum példányosítása:

- **New-Object** cmdlet, pl.:

```
$aes = new-object System.Security.Cryptography.AesManaged  
$aes.GenerateKey()
```

- Metódusait meghívhatom, tulajdonságait elértem...



DEMO .NET osztályok használata

- Friss blogbejegyzések lekérdezése
(forrás: Wikipedia)

```
$rssUrl = 'http://blogs.msdn.com/powershell/rss.aspx'  
$blog = [xml](new-object  
System.Net.WebClient).DownloadString($rssUrl)  
$blog.rss.channel.item | select title -first 4
```

PSDrive

- Sok forrás hasonlóan épül fel
 - Fájlrendszer, registry...
- Kezeljük ezeket azonoson!
 - Get-Item, New-Item...
- Ugyanúgy lehet átváltani:
 - Fájlrendszer `cd c:`
 - Registry `cd HKLM:`
 - Környezeti változó `cd env:`
- PSDrive lista:
 - Get-PSDrive

Közös paraméterek (Common parameters)

- Mindegyik beépített cmdlet ismeri ezeket
 - Debug, Verbose, ErrorAction...
- Használhatjuk saját szkriptben is:
 - [CmdletBinding()]
- Példa:
 - Write-Verbose "text"
 - Csak akkor írja ki, ha a -Verbose paramétert megadjuk



28



Bővebben:

Get-Help about_CommonParameters

Hibakezelés

- Non-terminating / terminating hiba
- **\$error**: bekövetkezett hibák listája
- **-ErrorAction**: mi történjen hiba esetén
 - Continue (alapértelmezett), SilentlyContinue, Stop...
- Kulcsszavak: **try / catch / throw**
- Kiírás: **Write-Warning / Write-Error**
- Figyeljük meg a beépített cmdletek működését!



29



Lásd a súgóban: `about_Throw`, `about_Try_Catch_Finally`

Soós Tibor PS könyvéből:

„A PowerShellben két hibafajta van, a „terminating error”, azaz a futást mindenképpen megszakító, és a „nonterminating error”, azaz a futást nem feltétlenül megszakító hiba.

- Megszakító hibák például a szintaktikai hibák, amikor elgépelünk valamit. Vagy például amikor nullával szeretnénk osztani. Mi magunk is generálhatunk ilyen hibákat a korábban már látott `throw` kulcsszóval, amikor például egy függvényünknek nem ad át a felhasználó minden fontos paramétert.
- Előfordulhatnak olyan hibák, amelyek előállásakor nem kívánjuk, hogy a szkript futása megszakadjon, de azért szeretnénk értesülni ezekről. Ez főleg olyan cmdleteknél és függvényeknél jöhet jól, amelyek csőelemeket dolgoznak fel, és egy-két csőelem esetében megengedjük, hogy ne fusson le a szkript, de azért a többi elemre nyugodtan próbálkozzon.”

További tippek

- & parancs – parancs végrehajtása
- \$? – sikeres volt-e az előző utasítás
- Sortörés: ` (HU billentyűzeten: AltGr + 7)
- Számított tulajdonságok:

```
Get-process | select -property @{n="nev";  
    e={$_.name}}, @{n="nap"; e={$_.StartTime.Day}}
```

Komplexebb feladat

Fájl jogosultságok beállítása

Feladat szövege

Készítsünk egy PowerShell scriptet, ami könyvtárakra állít be további ACL-eket egy paraméterként kapott CSV alapján. A bemeneti CSV:

```
folder,principal,allow,deny  
c:\temp\A,Administrators,Read;Write,  
c:\temp\A,Users,Read,Write
```

Egy sor tehát megad egy adott könyvtárat, egy szereplőt (helyi felhasználót vagy csoportot), akire a jogosultságok érvényesek, valamint engedélyező és tiltó jogokat. Az allow és deny résznél több jog is szerepelhet, ezek ilyenkor pontosvesszővel vannak elválasztva. Az is megengedett, hogy az allow vagy a deny részek valamelyike üres legyen.



Hogyan álljunk neki?

- Megkeresni, hogy hogyan lehet PowerShellben fájlrendszer jogokat kezelni
 - Get-Acl, Set-Acl cmdlet
- Játszani kicsit ezekkel
 - Get-Acl testdir
 - (Get-Acl testdir).Access
- Megnézni, hogy a Set-Acl hogyan működik
 - FileSystemAccessRule objektumokat kell hozzáadni
 - [MSDN leírás](#)
- Nem specifikált: meglévő jogokkal mi legyen



33



MSDN. „FileSystemAccessRule Class”, URL: <http://msdn.microsoft.com/en-us/library/system.security.accesscontrol.filesystemaccessrule.aspx>

Megoldás felépítése

- Fejkomment
- Bemenet ellenőrzése
- CSV-n végigiterálni
 - Import-Csv – típusos feldolgozás!
 - Könyvtár létrehozása, ha kell
 - Allow jogok feldolgozása
 - Deny jogok feldolgozása

DEMO Példakód (nem túl powerShelles)

```
<#  
.SYNOPSIS  
Creates folders from a CSV file, and adds prescribed security descriptors  
.PARAMETER CsvPath  
Full path of the CSV input file  
.NOTES  
Author: Micskei Zoltan, 2013.02.26.  
#>  
  
param( [Parameter(Mandatory=$true)][string] $CsvPath )  
  
foreach ($folderAccess in Import-Csv $CsvPath){  
    if ( ! (Test-Path $folderAccess.folder) ) {  
        New-Item -type directory $folderAccess.folder > $null  
    }  
  
    foreach ($permission in ($folderAccess.Allow).Split(";")){  
        if ( ! ($permission.length -eq 0) ){  
            $acl = Get-Acl $folderAccess.folder  
  
            $accessRule = New-Object System.Security.AccessControl.FileSystemAccessRule `"  
                "$($folderAccess.principal)","$permission","Allow"  
  
            $acl.SetAccessRule($accessRule)  
            Set-Acl -aclObject $acl $folderAccess.folder  
        }  
    }  
  
    #TODO: finish Deny permissions, add error handling  
}
```



További információ

- [SHOT](#) – 10x10 perc online screencast magyarul
- [Soós Tibor: PowerShell 2 tankönyv](#) (magyarul)
- [PowerShell Tutorial](#) (10 részben, kicsit régi már)
- [PowerShell cheat sheet](#)



36



- Soós Tibor. „PowerShell”, TechnetKlub SHOT (Short Online Training), URL: <https://technetklub.hu/shot/#5>
- Soós Tibor. „Microsoft PowerShell 2.0 rendszergazdáknak – elmélet és gyakorlat”, Microsoft Magyarország, 2010. URL: <https://technetklub.hu/Downloads/Browser.aspx?shareid=1&path=PDF\E-Book+-+PowerShell+2.0+tank%C3%B6nyv>
- PowerShell Pro. „PowerShell Tutorial”, URL: <http://www.powershellpro.com/powershell-tutorial-introduction/tutorial-windows-powershell-console/>
- Dzone Refcardz. „Windows PowerShell”, URL: <http://refcardz.dzone.com/refcardz/windows-powershell>