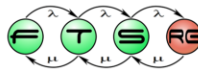


Scriptelés alapok (PowerShell)

Micskei Zoltán



Utolsó módosítás: 2011. február 22.

DEMO Kedvcsináló: MP3 taggelés

- Adott sok MP3 fájl, speciális elnevezési konvencióval
- Név alapján kéne az ID3 tageket kitölteni
- Könnyen automatizálható feladat
 - Később is kellhet
- Biztos van rá freeware/shareware, de
 - Megbízható? Azt csinálja, ami nekünk kell?
 - Informatikusok vagyunk, meg tudjuk írni☺
- Szkript <10 perc alatt elkészülhet



2



```
# Desc: Tag mp3 files based on filename
# Date: 2010.09.26.
# Uses: TagLib# http://developer.novell.com/wiki/index.php/TagLib_Sharp
# Based on: Editing Media Tags from PowerShell, http://huddledmasses.org/editing-media-tags-from-powershell/
```

```
param( [string] $folder = ".")
```

```
[Reflection.Assembly]::LoadFrom( "C:\temp\tools>taglib-sharp-2.0.3.7-windows\Libraries>taglib-sharp.dll" ) >
$null
```

```
$separators = '_,'
```

```
foreach ($file in (Get-Childitem -Path $folder -Filter *.mp3))
{
```

```
    Write-Output "Processing $file,,
```

```
    $media = [TagLib.File]::Create($file.FullName)
```

```
    $tokens = $file.Name.Split( $separators, [StringSplitOptions]::RemoveEmptyEntries )
```

```
    if ( $tokens.Length -lt 3 )
```

```
    {
        Write-Error "Not enough tokens in $file.FullName"
        continue
    }
```

```
    $media.Tag.Performers = ($tokens[0])
```

```
    $media.Tag.Title = ($file.Name.SubString( $file.Name.IndexOf("_") + 1, $file.Name.LastIndexOf("_") -
```

```
    $file.Name.IndexOf("_" - 1)).Replace("_", " ")
```

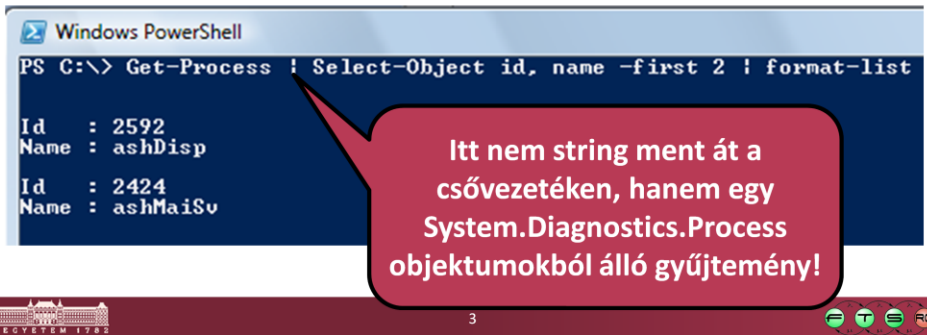
```
    $media.Tag.Comment = $tokens[ $tokens.length - 2 ]
```

```
    $media.Save()
```

```
}
```

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

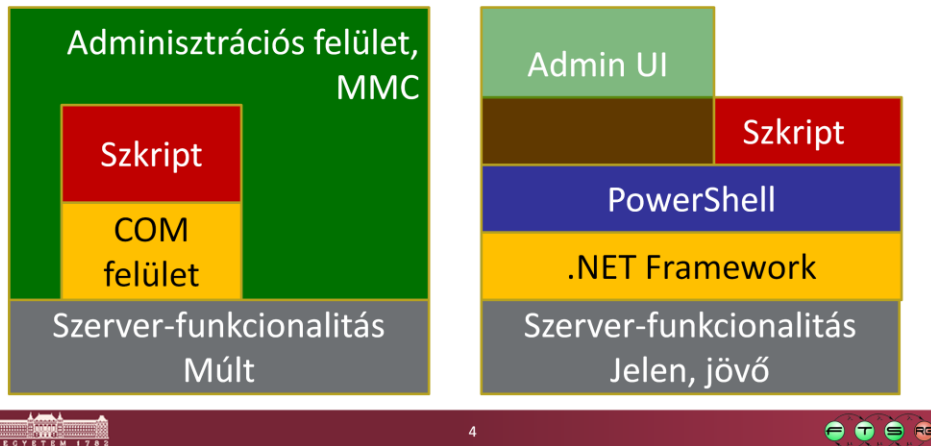
Id      : 2592
Name    : ashDisp
Id      : 2424
Name    : ashMaiSu
```

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?

- Új automatizálási motor a windowsos alkalmazásokhoz:



Forrás: 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 2010: [PowerConsole](#)
- VMware:
 - [PowerCLI](#) – teljes virtualizációs környezet automatizálása
- [Sense/Net 6.0 portál motor](#)
- ...

PowerShell felhasználása

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

Figyelem! Szkriptnyelv!

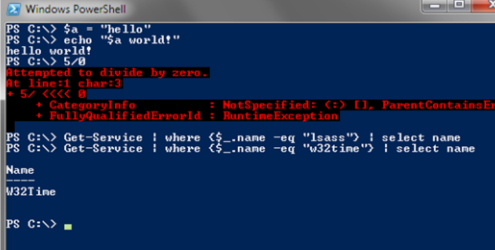
Célok:

- 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:3
+ 5/ <<<< 0
+ ~~~~~
+ CategoryInfo          : NotSpecified: (:) [], ParentContainsErrorRecords
+ FullyQualifiedErrorId : RuntimeException

PS C:\> Get-Service | where {$_.name -eq "lsass"} | select name
PS C:\> Get-Service | where {$_.name -eq "w32time"} | select name
Name
----
W32Time
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-Tárgy elnevezés
 - Adott funkciót megvalósító „parancs”
 - (háttérben: Cmdlet .NET osztály leszármazottai)

- Alap parancsokhoz megszokott aliasok
 - Pl. cp, copy -> Copy-Item

- Nyelv nem kis/nagybetű érzékeny

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

■ Súgó *cmd*letek:

- **Get-Command**: parancsok listázása
 - Szűrés pl.: `Get-Command -Noun csv`
- **Get-Help**: súgó, paraméter leírás, példák
 - `Get-Help Get-ChildItem -examples`
 - `Get-Help about_*`

■ Grafikus formában:



DEMO PowerShell alapok

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

Powershell változók

- Változó: `$nev`
- Típuskonverzió automatikus
 - Pl.: `$a = "Hello"` # `System.String`
 - De: `[int] $ev` # explicit megadás
- Lehet bármilyen .NET objektumot létrehozni:
 - `$list = New-Object System.Collections.ArrayList`
- Mit csinálhatok egy változóval?
 - `Get-Member -InputObject $list`
- Escape szekvenciák: ``t`, ``n` ...

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"]`

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 (%):
`Get-Process | Foreach-Object {Write-Output $_.Name}`
\$_: aktuális elem
- Szűrés (?):
`Get-Process | Where-Object {$_.Id -eq 4}`



A pipe-ban mindig típusos, strukturált objektumok utaznak, így sokkal könnyebb kezelni őket.

A pipe 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

```
# Name:    script.ps1
# Author:  Micskei Zoltán
# Date:    2010.02.17.
# Desc:    Example template for powershell script
# Param:   $hello - string to write out

param(
    [string] $hello = $(throw "Supply the string!")
)

Write-Output $hello
```

Fejkomment

Paraméter
megadás

Paraméterek ellenőrzése

- Param kulcsszó
- Megadható:
 - Típus, alapérték, kötelezőség, hibaüzenet
- ParamTest.ps1:

```
param(  
    [string] $msg = $(throw "Supply the message!"),  
    [int] $num = 2,  
    [switch] $flag  
)
```

- ParamTest meghívására példák:
 - .\ParamTest.ps1 -msg "hello" -flag
 - .\ParamTest.ps1 -num 3 -msg "hello"

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 PowerGUI-t
 - Breakpoint, Variables...
- Írjunk egy scriptet, ami lekérdezi, hogy hány svchost.exe fut, és hogy a legtöbb memóriát foglaló az 10 MB-nál többet használ-e!
- Írjunk egy scriptet, ami egy CSV fájlban tárolt neveket és HF pontokat kiolvassa kiírja az adott emberek átlagát.



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
```

Jegyek feladat:

```
# Name: Compute-HF.ps1
# Author: Micskei Zoltán
# Date: 2010.02.17.
# Desc: Calculate student's average
# Param: $csvFile - csv containing students' results
```

```
param(
[string] $csvFile = $(throw "Supply the path of the CSV file!")
)
```

```
Import-Csv -Path $csvFile | ForEach-Object { [double] $atlag = ([int]$_HF1 +
[int]$_HF2 + [int]$_HF3) / 4; Write-Output "$($_.Nev): $atlag" }
```

CSV fájl:

```
Nev,HF1,HF2,HF3
Micskei Zoltan,3,4,3
Szatmari Zoltan,10,8,9
Toth Daniel,12,12,11
```

.NET osztálykönyvtár használata

- Statikus metódus meghívása:
 - `[nevtér.osztály]::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 elérem...

DEMO .NET osztályok használata

- Friss blogbejegyzések lekérézése
(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

További tippek

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

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

Komplexebb feladat

Fájl jogosultságok beállítása, korábbi HF

Feladat szövege

Készítsen 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, hogy a meglévő jogokkal mi legyen

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)

```
# Name: Create-FoldersWithAcl.ps1
# Author: Micskei Zoltán
# Date: 2009.02.26.
# Desc: Creates some folders from a CSV file, and adds some security descriptors
# Param: 1 - full path of the CSV file

param([string] $csvPath = $(throw "Supply one CSV as parameter"))

foreach ($folderAccess in Import-Csv $csvPath){
    if ( ! (Test-Path $folderAccess.folder) ) {
        New-Item -type directory $folderAccess.folder
    }

    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 -ac/Object $acl $folderAccess.folder
        }
    }
    ...
}
```



További információ

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