Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

Architektúra

- .NET osztályok:
  - Objektumok
    - PSObject modell
  - Gyűjtemények:
- WMI (Windows Management Instrumentation)
- COM (Component Object Model):
  - OLE
  - ActiveX
- ADSI (Active Directory Service Interface)

Objektum-orientáltság

- Osztály:
  - Változók
  - Metódusok:
    - Metódus overload
    - Statikus metódus
  - Absztrakt osztály
- Példány (objektum):
  - Példányváltozó, példánymetódus
  - Futás közbeni kötés
  - Gyűjtemények
- Öröklődés

A .NET FrameWork

- CLR (Common Language Runtime)
- Class Library
- Nyelvek:
  - C#
  - VB
  - Jscript
- IL (Intermediate Language) kód.
- Futtatás (JIT Compiler).
- Assembly-k.

PowerShell alapelemek

- Cmdlet:
  - Get-Command
  - Get-Help
  - Get-Member
- Cmdlet felépítése:
  - Ige (verb) – főnév (noun)
    - Get-*; Set-*; New-* stb.
      - Get-Item; Set-Item; New-Item stb.
  - Paraméterek:
    - Megnevezett
    - Sorrendi (pozicionális)

Metódusok és property-k

- A visszaadott objektum típusa meghatározza:
  - A tulajdonságokat (property).
  - A végrehajtható műveleteket (method).
  - Az örökölt metódusokat és tulajdonságokat.
- Object.GetType() metódus.
- Get-Member cmdlet.

Objektumviszonyok

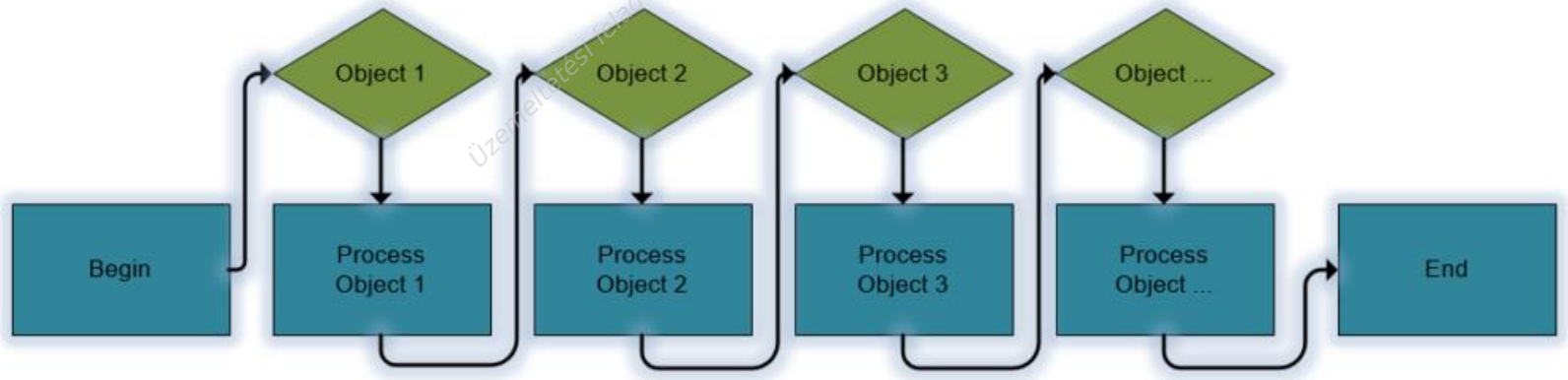
PowerShell objektumnézet	Leírás
PSBase	Az eredeti objektum
PSAdapted	A PowerShellben megvalósított nézet
PSExtended	A PowerShell által hozzáadott elemek
PSObject	A PowerShell által előállított objektum

Kifejezés és parancsfeldolgozás

- Kifejezés-feldolgozó üzemmód:
  - ha a beírt szöveg számmal vagy egy pont karaktert követő számmal (PS C:\> 2+2),
  - idézőjelek közé tett karakterlánccal (PS C:\>"Hello"),
  - vagy $ jellel kezdődik (PS C:\> $a).
- Parancs-feldolgozó üzemmód:
  - ha a beírt szöveg bármilyen betűvel (PS C:\> Get-Date),
  - a & karakterrel (PS C:\> &"Get-Date"),
  - egy pont utáni szóközzel, vagy pont utáni betűvel kezdődik (PS C:\>. .\StartDemo.ps1).

Csővezeték (pipe)

- Parancs | parancs
  - Parancsok kimenete: objektum!
  - Parancsok bemenete: objektum!
  - $_ vagy $PSItem változó.



Paraméterátadás: ByValue

String objektumok a csővezetékben

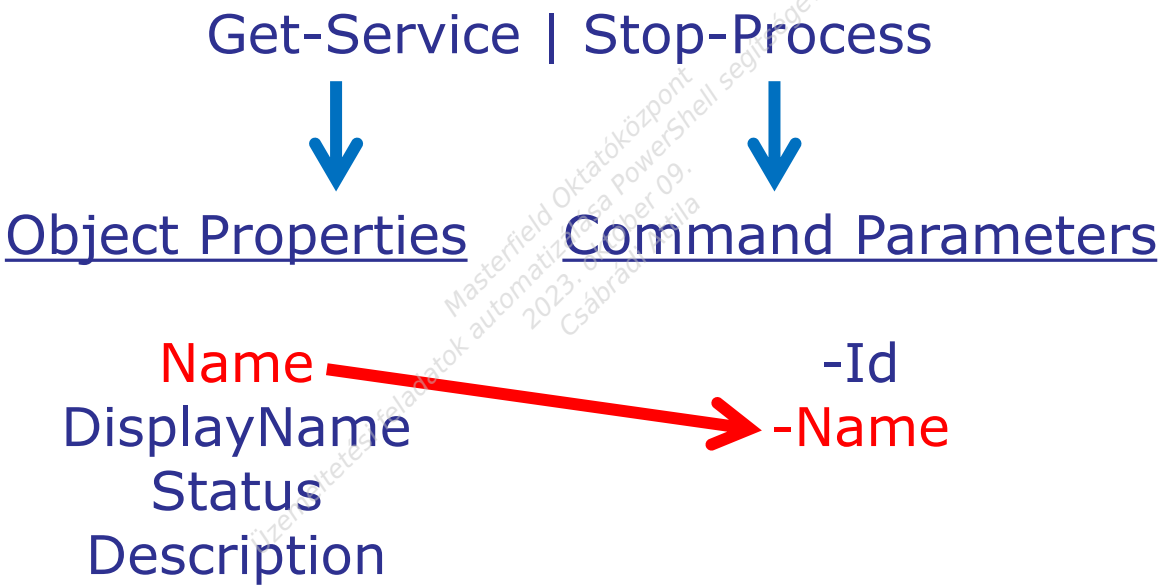
"BITS","WinRM" | Get-Service -Name

Átadva annak a paraméternek, ami fogad a csővezetékből „nyers” adatokat.

ByValue paraméterek

Required?	false
Position?	named
Default value	none
Accept pipeline input?	true (ByValue)
Accept wildcard characters?	false

Paraméterátadás: ByPropertyName



ByPropertyName paraméterek

Required?	false
Position?	named
Default value	none
Accept pipeline input?	true (ByPropertyName)
Accept wildcard characters?	false

Átirányítás

- Általánosan:
  - Get-Command > C:\Temp\parancsok.txt
    - Felülíró üzemmód.
  - Get-Process >> C:\Temp\folyamatok.txt
    - Hozzáfűző üzemmód.
- Specifikusan:
  - Kimenetek:
    - Success
    - Error
    - Warning
    - Verbose
    - Debug
    - Information
  - Speciális átirányítás:
    - Get-Childitem 'C:\Windows', 'C:\Nincsilyen' 2>&1 > .\dir.log

1	**Success** Stream	PowerShell 2.0
2	**Error** Stream	PowerShell 2.0
3	**Warning** Stream	PowerShell 3.0
4	**Verbose** Stream	PowerShell 3.0
5	**Debug** Stream	PowerShell 3.0
6	**Information** Stream	PowerShell 5.0
*	**All** Streams	PowerShell 3.0

Helyettesítő karakterek használata

- Konzolparancsok és cmdletek paramétereiben:
  - Dir *.txt
  - Get-Command *-Command
  - Get-Command *-Co*
  - Get-Command Get-Co?????
  - Get-Command Get-Co[mn]????
- -like; notlike (clike/cnotlike, ilike/inotlike) operátorok:
  - A like operátor **mintaillesztést** végez. Ez azt jelenti, hogy akkor ad igaz értéket vissza, ha a feldolgozandó szöveg egésze illeszkedik a mintára.
- Példák:
  - "Gőgős", "Gúnár", "Gedeon" -like "g*n*"
  - "Mosó", "Masa", "mosodája" -clike "[m-n]o?*"

Helyettesítő karakterek

Wildcard	Jelentés	Példa
*	Nulla vagy több karakter helyettesítése	*.txt
?	Egy karakter helyettesítése	*.tx?
[xyz]	A megadott karakterek közül bármelyik	[ABCDEFGH]*.txt
[x-z]	A megadott tartományon belüli karakterek közül bármelyik	[A-Z]*.txt

Reguláris kifejezések

- -match; notmatch (cmatch/cnotmach, imatch/inotmatch) operátorok:
  - A match operátor **mintakeresést** végez. Ez azt jelenti, hogy akkor ad igaz értéket vissza, ha a feldolgozandó szövegben megtalálható a minta általmeghatározott szövegrész.
  - Természetesen lehet olyan keresőmintát írni, amely csak az egész szövegre illeszkedik.
  - A match operátor az első illeszkedésnél befejezi a keresést.
- Példák:
  - "Gőgős", "Gúnár", "Gedeon" -match "g.*n.*"
  - "Mosó", "Masa", "mosodája" -cmatch "[m-n]o.*"

Reguláris kifejezések

RegEx	Jelentés	Példa
.	Bármely karakter egy előfordulása	.o.th
[xyz]	A megadott karakterek közül bármelyik	[abcdefgh]*\.txt
[x-z]	A megadott tartományon belüli karakterek közül bármelyik	[a-z]*\.txt
^	A szöveg kezdete	^Szöveg
$	A szöveg vége	szöveg.$
*	Nulla vagy több előfordulása a megelőző helyettesítő karakternek vagy mintának	W.*s
+	Egy vagy több előfordulása a megelőző helyettesítő karakternek vagy mintának	W.+s
?	Nulla vagy egy előfordulása a megelőző helyettesítő karakternek vagy mintának	W.?ndows
{n.m}	Legalább n, de nem több mint m előfordulása a megelőző helyettesítő karakternek vagy mintának	[a-z]{2,4}

Reguláris kifejezések

RegEx	Jelentés	Példa
\	A jelölést követő karakter speciálisan dolgozandó fel (escape)	Kérdés\?
\b	Szóhatároló karakter	\bHello\b
\s	Szóköz karakter	Hello\sworld!
\d	Bármely számjegy karakter	\d{3}
\w	Bármely szó karakter, a vesszőket és a whitespace karaktereket kivéve	\w*
\t	Tabulátor	1\t2
()	Csoportosítás	"123-456" -match "(\d{3})-(\d{3})"
?<csoportcímke>	Csoportosítás címkéjének (nevének) megadása	"123-456" -match "(?<a>\d{3})-(?<b>\d{3})"
|	VAGY logikai operátor	"Power-shell" -match "Power[\s|-]shell"

Reguláris kifejezések

- System.Text.RegularExpressions.Regex osztály:
  - A .NET saját reguláris kifejezés feldolgozó osztálya. Egynél több találatot is tud kezelni.
  - Metódusok:
    - Match
    - Matches
    - Split
    - Replace
  - Példa:
    - $pattern = [regex] "\b\w+-shell\b"
    - $pattern.matches("Power-shell is a great command-shell!")

Reguláris kifejezések

- Select-String:
  - A Select-String cmdlet feladata szöveg és szövegminták keresése a bemeneti karakterláncokban és fájlokban. Minták megadásánál reguláris kifejezéseket használhatunk.
- Szintaxis:
  - Select-String -Pattern <minta> -CaseSensitive -SimpleMatch -AllMatches
- Példa
  - "Power-shell is a great command-shell!" | Select-String -Pattern "\b\w+-shell\b" -AllMatches

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo .\00_01.txt

Változók

- $ karakterrel kezdődnek.
- Variant típusúak (objektumváltozók).
- Érték és referencia.
- Konverzió
  - Implicit
  - Explicit
- Get/Set-Variable

Változók

- Beépített (rendszer) változók:
  - $$: parancs - utolsó token.
  - $^: parancs - első token.
  - $?: utolsó művelet sikeressége.
  - $null: NULL érték.
  - $true: IGAZ érték.
  - $false: HAMIS érték.
  - $PSVersionTable: PowerShell verzióadatok.
  - $PSCulture: PowerShell által használt regionális beállítások.
  - $PWD: aktuális könyvtár.
  - $_ vagy $PSItem: a csőszakaszból kilépő objektum.
  - $input: belső tömbváltozó az inputértékek kezelésére.

Változótípusok

PowerShell rövid név	.NET típusnév
[int]	System.Int32
[long]	System.Int64
[string]	System.String
[char]	System.Char
[bool]	System.Boolean
[byte]	System.Byte
[double]	System.Double
[decimal]	System.Decimal
[float]	System.Single
[single]	System.Single
[regex]	System.Text.RegularExpressions.Regex
[array]	System.Array
[xml]	System.Xml.XmlDocument
[scriptblock]	System.Management.Automation.ScriptBlock
[switch]	System.Management.Automation.SwitchParameter
[hashtable]	System.Collections.Hashtable
[psobject]	System.Management.Automation.PSObject
[type]	System.Type
[datetime]	System.DateTime
[void]	System.Void

Tömbök

- Skalár változó vs. Tömbváltozó.
- Azonos elemeket tartalmazó tömbök:
  - [int[]] $ia = 1,2,3,4
- Nem azonos elemeket tartalmazó tömbök:
  - $files = Get-ChildItem C:\Windows
- Többdimenziós tömbök:
  - $table = (1,2,3,4,5),("a","b","c","d","e")
- Asszociatív tömbök:
  - $hash = @{ Név = "Gipsz Jakab";
    Cím = "Budapest";
    "E-mail" = "jgipsz@domain.local" }

Operátorok

- Aritmetikai:
  - +; -; ++; --; *; /; %
  - =; +=; -=; *=; /=
- Összehasonlító:
  - -(c)eq: egyenlő; -(c)ne: nem egyenlő
  - -(c)gt: nagyobb; -(c)ge: nagyobb egyenlő
  - -(c)lt: kisebb; -(c)le: kisebb egyenlő
- Logikai:
  - -or; -and; -xor; -not
- Bináris:
  - -bor; -band; -bxor; -bnot
- Típusvizsgálati
  - -is; -isnot

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo .\00_02.txt

Elágazás

- Egyágú:
  - if (feltétel) { művelet(ek)}
- Kétágú:
  - if (feltétel) { művelet(ek)} else { művelet(ek)}
- Egymásbaágyazott:
  - if (feltétel)
    { művelet(ek) }
    elseif (feltétel)
    { művelet(ek) } else { művelet(ek) }

Többirányú elágazás

- Switch:

```
switch( változó )
{
    { feltétel }
            { művelet(ek); break }
    { feltétel }
            { művelet(ek); break }
    default
            { művelet(ek) }
}
```

- Minden igaz ágat végrehajt!
  - Break használat.

Feltételes ciklusok

- While
  - while (feltétel) # amíg igaz
    { művelet(ek) }
- Do..While
  - do
    { művelet(ek) }
    while (feltétel) # amíg igaz
- Do..Until
  - do
    { művelet(ek) }
    until (feltétel) # amíg hamis

Növekményes ciklusok

- For:
  - for (inicializálás; feltétel; léptetés) { művelet(ek) }
- ForEach:
  - foreach ($elem in $tömb) { művelet(ek) }
  - parancs | foreach { művelet(ek) a $_ változóval }
    - Látszólagos (ld. Foreach-Object).
- ForEach-Object cmdlet:
  - parancs | Foreach-Object { művelet(ek) }

Ciklusvégrehajtás megszakítása

- Break:
  foreach ($elem in $tömb)
  { if (feltétel) { break } else { művelet(ek) } }
- Continue:
  foreach ($elem in $tömb)
  { if (feltétel) { continue } else { művelet(ek) } }

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo .\00_03.txt

Függvény

- Névvel rendelkezik.
- Visszatérési értéke van.
- Egyszerű definíció
  - Function függvény (paraméter(ek))
    {
     művelet(ek)
    }
  - Function függvény
    {
     param(paraméter(ek))

            művelet(ek)
    }

Függvényparaméterek

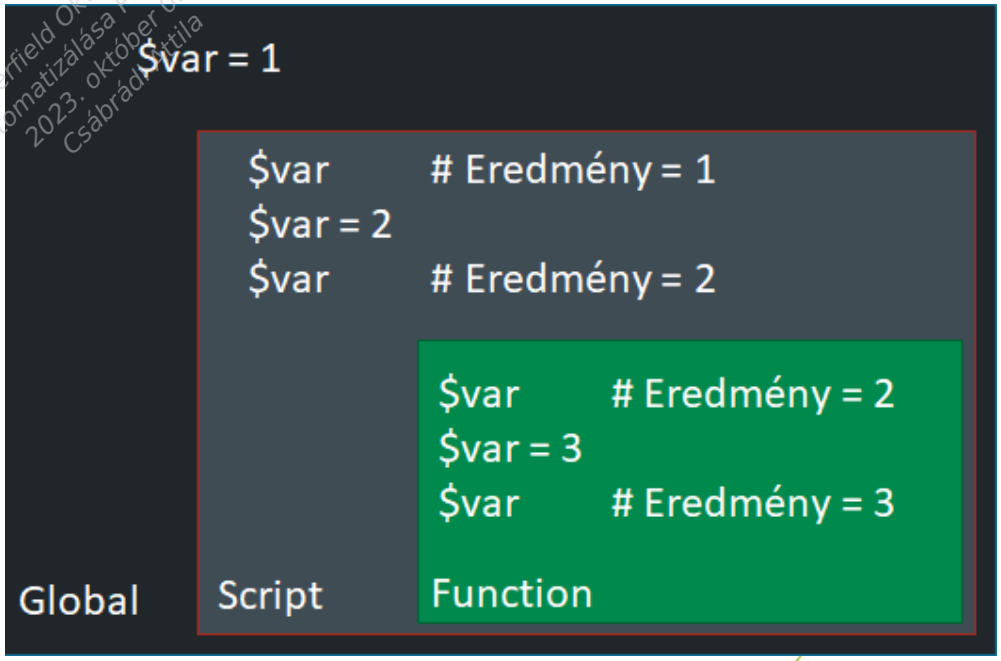
- Függvénydefiníció:
  - Function szorzat ($a,$b)
    {
        return $a*$b
    }
- Függvényhívás:
  - $c = szorzat 2 3
  - $c = szorzat -b 2 -a 3
  - $c = szorzat(2,3) # nem jó - tömbparaméter!
- Cím szerint paraméterátadás:
  - Function szorzat ([ref]$a, $b)
    {
        $a.value=$a.value*$b
    }
  - szorzat ([ref]$c) 2

Paraméterinicializálás

- Paraméterek alapértékének beállítása:
  - Function szorzat ($a = 2,$b=2)
    {
    return $a*$b
    }
  - szorzat # visszatérési érték: 4
  - szorzat 3 # visszatérési érték: 6
  - szorzat ”3” # visszatérési érték: 33
- Típusos paraméterek:
  - Function szorzat ([int]$a = 2, [int]$b = 2)
    {
    return $a*$b
    }
  - szorzat ”3” # visszatérési érték: 6

Láthatóság (Scope)

- Privát változó definiálása:
  - Get-Variable -Scope 1
  - Get-Variable -Private -Scope 1
  - $Private:változó
- Hatókör emelés:
  - Function global:első { "első" }
- Dot sourcing:
  - . C:\Scripts\script.ps1



Függvény vs. Filter

- A PowerShell-ben:
  - Function - tömb feldolgozás ($input).
  - Filter - elemenkénti feldolgozás ($_).
- Feldolgozási futószalag definiálása:
  - Function függvény (paraméter(ek))

```
{
        begin { művelet(ek) pl. inicializálás }
        process {művelet(ek) }
        end { művelet(ek) pl. visszatérési érték}
}
```

- Filter: mindhárom blokk, elemenkénti végrehajtás!

ScriptBlock

- A ScriptBlock egy műveletsor.
- Felfogható név nélküli függvénynek pl.:
  - 1,2,3 | &{process {$_*2}}
- ScriptBlock akár paraméter is lehet:
  - Function Execute-ScriptBlock ([scriptblock] $Process)

```
{
    Process {&($Process) }
}
```

  - 1,2,3 | Execute-ScriptBlock {$_*2}
- Saját paraméterrel rendelkezhet:
  - $script = {param($a) process {$_*$a}}
  - 1,2,3 | & $script 3

Scriptek

- Fájlba mentett műveletsor - technikailag egy fejrész nélküli függvény.
- Kiterjesztése .ps1 (PowerShell 1.0).
- Megjegyzés # karakter használatával.
- Soronkénti végrehajtás!
  - Ha nem kerül rá a vezérlés, nem ad hibajelzést!
- Engedélyezés:
  - Set-Executionpolicy

Scriptek futtatása

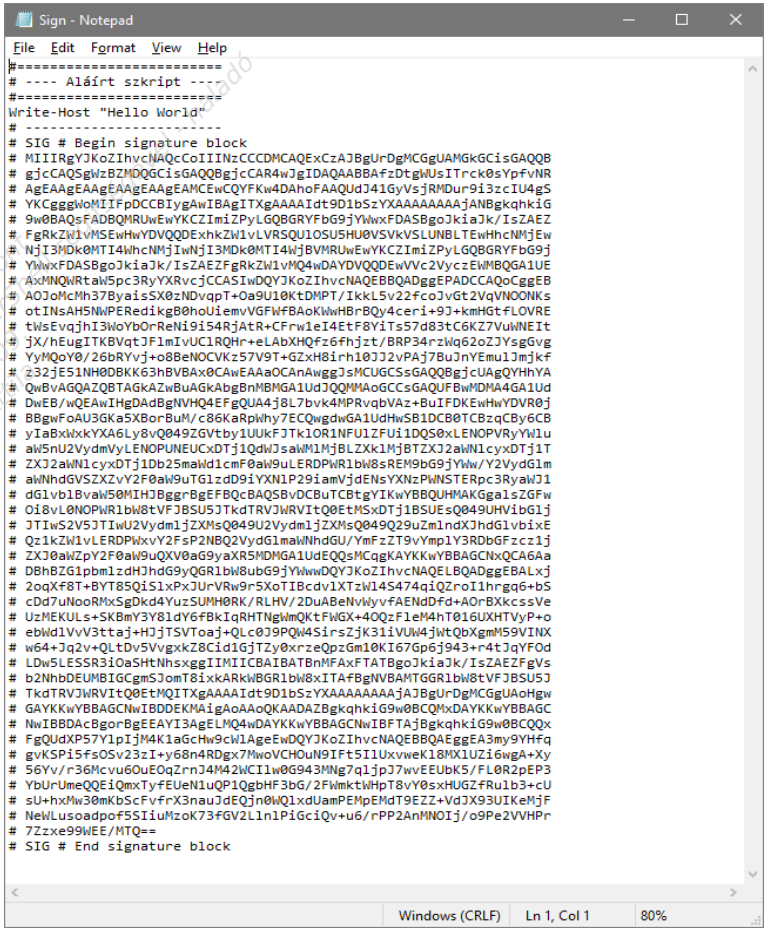
- Be kell állítanunk a megfelelő végrehajtási házirendet (execution policy).
- A szkript indításához adjuk meg annak teljes útvonalát, illetve ha a fájl az aktuális mappában van, használjuk a .\ jelölést.
- Ha az útvonal szóközöket tartalmaz, tegyük idézőjelek közé és írjuk elé a futtató karaktert (&).

Végrehajtási házirend

- Get/Set-ExecutionPolicy:
  - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell\ExecutionPolicy
- ExecutionPolicy paraméter:
  - Unrestricted
  - RemoteSigned
  - AllSigned (Unblock-File)
  - Restricted
  - Default
  - Bypass
  - Undefined
- Scope paraméter:
  - MachinePolicy
  - UserPolicy
  - Process (Powershell.exe -ExecutionPolicy Unrestricted)
  - CurrentUser
  - LocalMachine
- Get-ExecutionPolicy -List

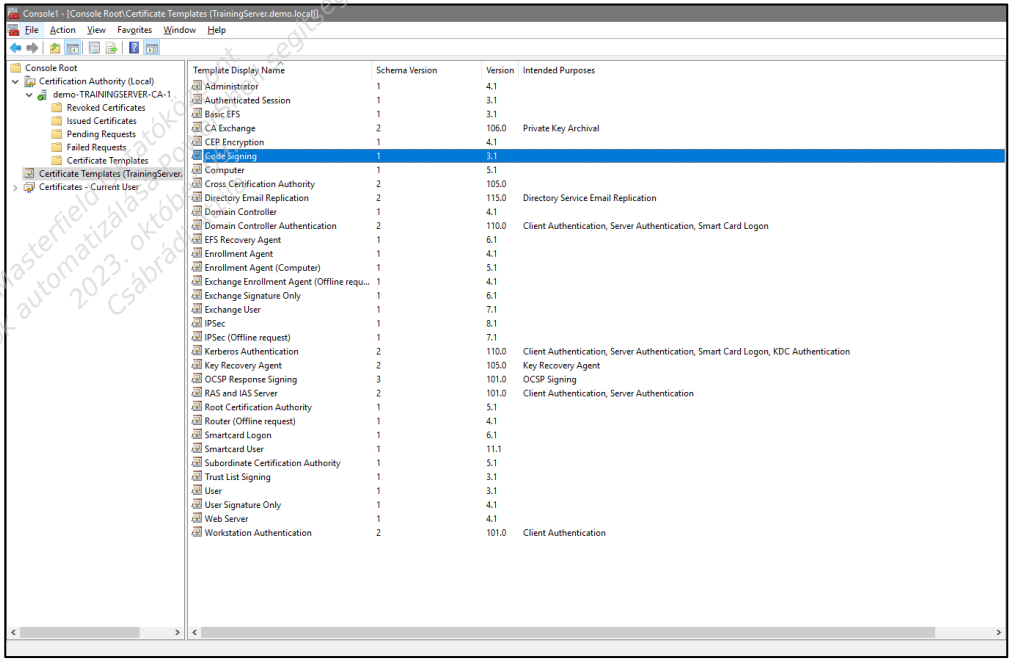
Digitális aláírás

- Célja
  - Felhasználó azonosítása, aki létrehozta / utoljára módosította a szkriptet.
  - Sértetlenségellenőrzés.
- Set-ExecutonPolicy:
  - AllSigned
  - Remotesigned



CodeSigning tanúsítvány

- Aláírni csak CodeSigning célú tanúsítvánnyal lehet.
- Lehet önaláírt, vagy tanúsítványszolgáltató által kibocsájtott.
- Tanúsítványszolgáltató:
  - Alapértelmezésben nincs közzétéve a tanúsítványsablon



Szkriptaláírás

- Szkriptek aláírása:
  - Set-AuthenticodeSignature -FilePath C:\Scripts\Sign.ps1 -Cerificate $signcert
- Szkriptaláírás lekérdezése:
  - Get-AuthenticodeSignature -FilePath C:\Scripts\Sign.ps1 |` Format-List -Property *

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo .\00_04.txt

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Haladó függvénykészítés

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

Kötelező paraméterek megadása

```
Function függvénynév
{
[CmdletBinding(SupportsShouldProcess)]
        Param  (
                [Parameter(
                Mandatory = $true,
                Position = 0
                ValueFromPipeline = $true,
                 ValueFromPipelineByPropertyName = $true
                )]
                [típus] $paraméternév
                )
   Process
        { ScriptBlock }
}
```

Kötelező paraméterek megadása

- [CmdletBinding(SupportsShouldProcess)]:
  - Cmdletbinding: csak megfelelő paraméterezés esetén működjön, ne rakja a nem definiált paramétereket az args tömbbe.
  - SupportShouldProcess: a –whatif, -confirm és –verbose paraméterek használatának lehetősége.
- Mandatory:
  - A paraméter kötelező.
- Position:
  - Pozicionális hely.
- ValueFromPipeline:
  - A paraméter fogadhat inputot a csőből.
- ValueFromPipelineByPropertyName:
  - A paraméter fogadhat inputot a csőből olyan objektumtól, melynek valamely tulajdonságának neve megegyezik a paraméterével

Paraméterek ellenőrzése

Ellenőrző címke	Jelentése
[AllowNull()]	Megengedi a paraméternek a $null értéket.
[AllowEmptyString()]	Megengedi az üres sztring használatát.
[AllowEmptyCollection()]	Megengedi az üres gyűjtemény (tömb) megadását.
[ValidateCount(1,5)]	Csak olyan tömböt fogad el, amelynek elemszáma 1 és 5 között van.
[ValidateLength(1,10)]	Csak olyan sztringet enged meg, amely hossza 1 és 10 között van.
[ValidatePattern("\d{2}-\d{3}")]	Csak az adott regex kifejezést kielégítő szöveget enged meg.
[ValidateSet("Vasárnap", "Hétfő", "Kedd")]	Csak a megadott értékek adhatók meg.
[ValidateScript($_ -ge 0})]	A scriptblock feltételnek megfelelő értéket enged meg.

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 01_01.txt
- Start-Demo 01_02.txt

Paraméterkészletek definiálása

- A paraméterkészletek lehetővé teszik, hogy bizonyos paraméterek csak akkor legyenek kötelezőek, ha más paramétert is megadunk, vagy hogy egy függvényt többféleképp is paraméterezhessünk.

```
Function függvénynév
{
        [CmdletBinding(DefaultParameterSetName = "ParameterSet1")]
        Param(
                [Parameter(
                        Mandatory = $true,
                        Position = 0,
                        ParameterSetName = "ParameterSet1",
                        HelpMessage = "HelpMessage1")]
                        [típus] [ValidateScript({$_ -ge 0})] $param1,
                [Parameter(
                        Mandatory = $true,
                        Position = 0,
                        ParameterSetName = "ParameterSet2")]
                        [típus] [ValidateScript({$_ -ge 0})] $param2
        )
  If ($pscmdlet.ParameterSetName -eq "ParameterSet1")
        { ScriptBlock }
  Else
        { ScriptBlock }
}
```

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 01_03.txt

Help készítése

```
<#
 .Synopsis
  Converts Bytes into the appropriate unit of measure.
 .Description
  The Get-OptimalSize2 function converts bytes into the appropriate unit of
  measure. It returns a string representation of the number.
 .Example
  Get-OptimalSize2 1024
  Converts 1024 bytes to 1.00 KiloBytes
  .Example
  Get-OptimalSize2 -sizeInBytes 1048576
  Converts 1048576 bytes to 1.00 MegaBytes
  .Example
  1048576,1024 |Get-OptimalSize2
  Converts 1048576 bytes to 1.00 MegaBytes and 1024 bytes to 1.00 KiloBytes
 .Parameter SizeInBytes
  The size in bytes to be converted
 .Inputs
  [int64]
 .OutPuts
  [string]
 .Notes
  Requires -Version 2.0
 .Link
  http://powershell.org
#>
```

Dinamikus és alapértelmezett paraméterek

- A dinamikus paraméterek olyan paraméterek, amelyek bizonyos feltételek teljesülésekor jelennek meg a cmdleteknél, függvényeknél. A leggyakoribb ilyen feltétel, hogy a parancs éppen melyik provider környezetében fut, de természetesen megadhatunk saját feltételt is.
- Az alapértelmezett paraméterek megadhatóak a v3.0-tól bevezetett DefaultParameterValues hash tömbben. A paraméterértékek feltöltése után a rendszer, amennyiben a parancs vagy függvény paraméter értékét nem adtuk meg - az eltároltat fogja használni.

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 01_04.txt
- Start-Demo 01_05.txt
- Start-Demo 01_06.txt

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

A Windows Management Infrastructure (WMI)

Masterfield Oktatóközpont
http://www.masterfield.hu

WMI

- A Microsoft a WMI segítségével valósítja meg a Web alapú vállalatirányítási rendszert (Web-Based Enterprise Management - WBEM).
- Célja, hogy egy vállalati hálózati környezetben az információkezelést egységes technológiákkal oldja meg.
- A WMI integrált támogatást biztosít a CIM modellhez (Common Information Model = Általános információs modell). A CIM írja le egy vállalati környezetben található objektumokat, tulajdonképpen egy hatalmas adatbázis.

WMI adatstruktúrák

WMI Adat struktúrák	Magyarázat
Névterek	Részekre osztott információs konténerek.
	Elsősorban termékekhez vagy gyártókhoz kötődnek.
Osztályok	Névterekben találhatóak meg.
	Kezelhető komponensek reprezentációja.
Példányok	Valódi előfordulásai egy osztálynak.
Statikus metódus	Hatását az osztályon nem a példányon fejti ki.

WMI osztályok

- System Classes: A CIM előre definiált osztályainak egy gyűjteménye. Olyan elemeket tartalmaz, mint események regisztrálása, rendszer biztonsági beállítások, figyelmeztető üzenetek generálása. stb).

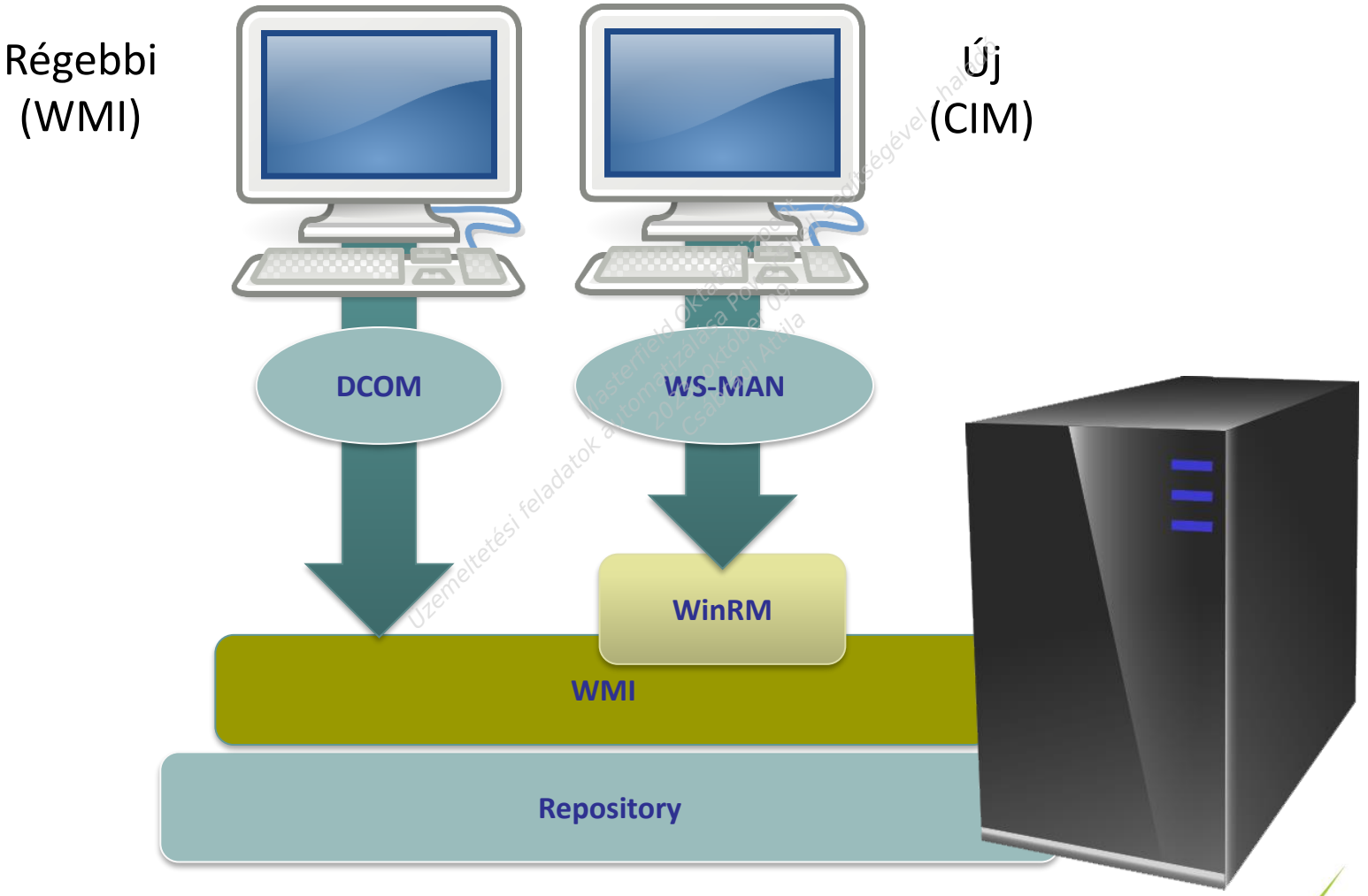
- Win32 Classes: A számítógép hardver elemeinek elérését biztosítja, még a processzor hűtőventillátor kezeléséhez (fordulatszám lekérdezés, szabályozás) is tartalmaz osztályokat.

- Standard Consumer Classes: Az egyéb kategóriába sorolható, bár cseppet sem mellékes: szkriptek által generált események kezelése, regisztrálása, naplózása, SMTP szolgáltatás és a parancssor által generált események kezelése.

WMI Provider-ek

- Win32
- SNMP
- Performance Counter
- Registry
- Windows Driver Model
- Directory Services
- Event Log
- Windows Installer
- Security

Cím vs. WMI



WMI vagy CIM?

	CIM	WMI
WMF 2.0 vagy későbbi szükséges	Igen	Nem
WRM bekapcsolása szükséges	Igen	Nem
Többplatformos megoldás	Igen	Nem
Egyportos tűzfalszabály	Igen	Nem
Session-based kapcsolat	Igen	Nem
Ad-hoc kapcsolat támogatása	Igen	Igen
Microsoft Windows XP támogatása	Igen	Igen
Microsoft Windows Server 2003 támogatása	Igen	Igen
Microsoft Windows NT 4.0 támogatása	Nem	Igen
Remote Administration tűzfalszabály megléte	Nem	Igen

WMI Explorer

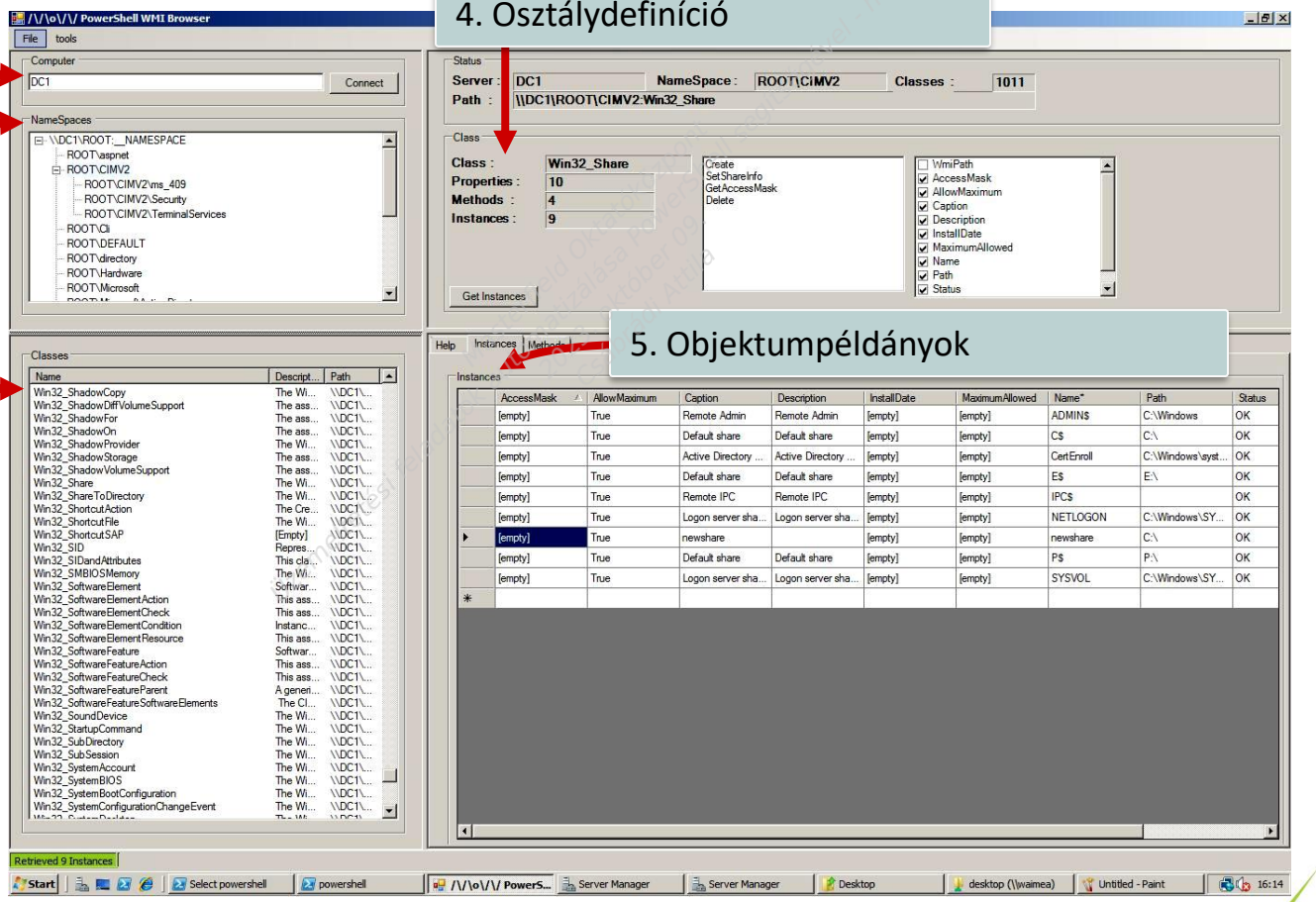
1. Kapcsolódás

2. Névterek

3. Osztályok

4. Osztálydefiníció

5. Objektumpéldányok



WMI cmdlet-ek

- Get-WmiObject
  - WMI osztály vagy objektumpéldány lekérdezése.
- Invoke-WmiMethod
  - WMI metódus hívása.
- Register-WmiEvent
  - Feliratkozás WMI eseményre.
- Remove-WmiObject
  - WMI osztály vagy objektumpéldány törlése.
- Set-WmiInstance
  - WMI objektumpéldány beállítása.

CIM cmdlet-ek

- Külön modul: CimCmdlets.
- Get-CimClass
  - WMI osztály lekérdezése.
- Get-CimInstance
  - WMI objektumpéldány lekérdezése.
- Invoke-CimMethod
  - WMI metódus hívása.
- Set-CimInstance
  - WMI objektumpéldány beállítása.
- New-CimInstance
  - WMI objektumpéldány létrehozása.
- Remove-CimInstance
  - WMI objektumpéldány törlése.

CIM cmdlet-ek

- Register-CimIndicationEvent
  - Feliratkozás WMI eseményre.
- Get-CimAssociatedInstance
  - Kapcsolódó WMI objektumpéldányok listázása.
- Get-CimSession
  - Hálózati kapcsolat (session) lekérdezése.
- New-CimSession
  - Hálózati kapcsolat (session) létrehozása.
- New-CimSessionOption
  - Hálózati kapcsolat (session) beállításainak létrehozása.
- Remove-CimSession
  - Hálózati kapcsolat (session) törlése.

WMI Where

- Egyszerű Where:
  - $query = "SELECT * FROM Win32_Service WHERE Name='AudioSrv'"
  - Get-WMIObject -Query $query
- Like:
  - $query = "SELECT * FROM Win32_Service WHERE Name LIKE '%Audio%'"
  - Get-WMIObject -Query $query
- Null:
  - $query = "SELECT * FROM Win32_LogicalDisk WHERE FileSystem IS NULL"
  - Get-WMIObject -Query $query
- Logikai operátorok:
  - $query = "SELECT * FROM Win32_Service WHERE (State='Running' OR State='Paused') AND Name LIKE '[af]%'"
  - Get-WMIObject -Query $query

WMI objektumpéldányok létrehozása

- New-CimInstance:
  - Csak olyan osztályok esetében, ahol nincs külön Create metódus.
- Create metódus hívásával:
  - $share = [wmiclass] "Win32_Share"
  - $share.Create("C:\Share","Temp",0)
  - $shareparams = @{Name = "Temp";
    Path = "C:\Share";
    Type = [uint32] 0}
  - Invoke-CimMethod -ClassName Win32_Share
    -MethodName Create
    -Arguments $shareparams

WMI objektumpéldányok módosítása

- Tulajdonság szerkesztésével:
  - Csak a nem Read-Only tulajdonságok esetében.
  - $share.MaximumAllowed = 25

- Metódushívással:
  - $share.SetShareInfo(25,$null,$null)

  - Invoke-CimMethod
    -Query "SELECT * FROM Win32_Share WHERE Name='Temp'"
    -MethodName SetShareInfo
    -Arguments @{MaximumAllowed = [uint32] 10;
    Description = $null;
    Access =$null}

WMI objektumpéldányok törlése

- Remove-WmiObject
  - $proc = Get-Wmiobject -Query "SELECT *
    FROM Win32_Process
    WHERE name LIKE 'notepad'"
  - Remove-WmiObject –InputObject $proc
- Remove-CimInstance
  - $proc = Get-Wmiobject -Query "SELECT *
    FROM Win32_Process
    WHERE name LIKE 'notepad'"
  - Remove-CimInstance -InputObject $proc
- Metódushívással
  - $proc.Kill()
  - $share.Delete()

WMI osztály és objektumkapcsolatok

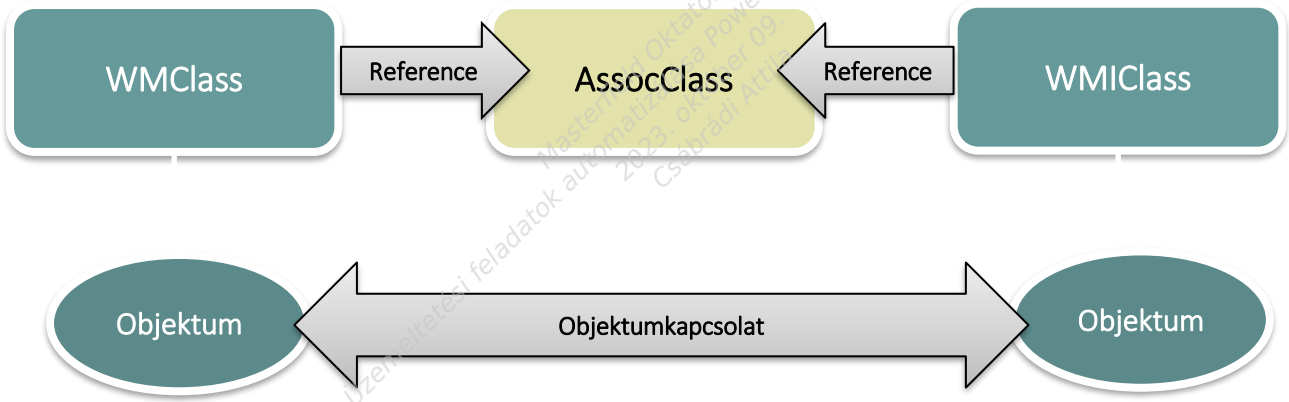
- Osztálykapcsolat:
  - $query = "ASSOCIATORS OF{Win32_Service.Name='NetLogon'}
    WHERE ClassDefsOnly"
  - Get-WMIObject -Query $query
- Kapcsoló osztályok (AssocClass) lekérdezése:
  - $query = "REFERENCES OF {Win32_Service.Name='NetLogon'}
    WHERE ClassDefsOnly"
  - Get-WMIObject -Query $query

WMI osztály és objektumkapcsolatok

- Objektumkapcsolat:
  - $query = "ASSOCIATORS OF {Win32_Service.Name='NetLogon'}
    WHERE ResultClass=Win32_Service"
  - Get-WMIObject -Query $query

  - $query = "ASSOCIATORS OF {Win32_Service.Name='NetLogon'}
    WHERE AssocClass=Win32_DependentService"
  - Get-WMIObject -Query $query

WMI osztály és objektumkapcsolatok



Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 02_01.txt

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Haladó technikák

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

Type accelerators

- $a = New-Object System.Int32
- [System.Int32] $a = 5
- [int] $a = 5
- $a = [int] 5

- [adsi] ""
- $tomb = [array] 1
- $ip=[ipaddress] "192.168.1.1"
- $sb = [scriptblock] {$_.Name -eq "PowerShell"}

Statikus metódusok

- Nem példányosítható, az osztályon fut.
- [system.math] | Get-Member MemberType Methods –Static
- [math]::pi
- [math]::Pow(3,2)

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 03_01.txt

Objektum típusok

- PS (.NET) objektum
  - Get-Item C:\PowerShell | Get-Member
- WMI objektum
  - Get-WmiObject Win32_Directory -Filter "Name='C:\\Powershell'" | Get-Member
- COM objektum
  - $Filesystem = New-Object -ComObject "Scripting.FilesystemObject"
  - $Filesystem.GetFolder("C:\PowerShell")
  - $Filesystem.GetFolder("C:\PowerShell") | Get-Member

Saját PS Objektum

- Objektum létrehozása:
  - $object = New-Object –TypeName PSObject
- Objektum tulajdonságainak megadása:
  - $object | Add-Member –MemberType NoteProperty –Name Name –Value (Get-Item C:\PowerShell).FullName
  - $object | Add-Member –MemberType NoteProperty –Name Mode –Value (Get-Item C:\PowerShell).Mode
  - $object | Add-Member –MemberType NoteProperty –Name Compressed –Value (Get-WmiObject Win32_Directory -Filter "Name='C:\\Powershell'").Compressed
  - $object | Add-Member –MemberType NoteProperty –Name Encrypted –Value (Get-WmiObject Win32_Directory -Filter "Name='C:\\Powershell'").Encrypted
  - $object | Add-Member –MemberType NoteProperty –Name Size –Value($Filesystem.GetFolder("C:\PowerShell")).Size
- Objektum kiírása:
  - Write-Output $object
- Objektum tulajdonságainak felderítése:
  - $object | Get-Member

Update-List

- Objektum tulajdonságaként szereplő tömb módosítására, manipulálására való.
- Update-List

  [-Add <Object[]>]
  [-Remove <Object[]>]
  [-Replace <Object[]>]
  [-InputObject <PSObject>]
  [[-Property] <String>] [<CommonParameters>]

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 03_02.txt

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Formátumbeállítás

Kimeneti formátum módosítása

- Kimeneti cmdletek:
  - Format-List
  - Format-Table
  - Format-Wide
  - Format-Custom
- Alapértelmezett kimeneti formátum:
  - C:\WINDOWS\System32\WindowsPowershell\v1.0
    - types.ps1xml
- Egyéni kimeneti tulajdonság definiálása:
  - Format-Table @{Label=címke;
    Expression={scriptblock};
    Width=szélesség}
- Egyéni objektum tulajdonság definiálása:
  - Select-Object @{Name=tulajdonságnév;
    Expression={scriptblock}}

Tulajdonságok bővítése

- Update-TypeData:
  - .ps1xml kiterjesztésű fájlt vár.
- Remove-TypeData
- Új típus létrehozása:
  - $user = New-Object -TypeName PSObject -Property property,property,property…
  - A tulajdonsághalmazt hash tömbként kell definiálni.
- Formázás beállítása:
  - Update-FormatData
  - format.ps1xml fájl.

Osztálydefiníció

PowerShellben:

- class Test {
  [int] $a
  [int] $b

  Test ([int] $x, [int] $y) # konstruktor
  {$this.a = $x; $this.b = $y; }

  [int] Add()
  {return ($this.a + $this.b);}

  static [int] Multiply([int] $n, [int] $m) # statikus metódus
  {return ($n * $m);}
  }

Osztálydefiníció

C#-ban:

```
$source = @"
public class Test{
        public int a {get; set;}
        public int b {get; set;}

        public Test (int x, int y)                  # konstruktor
                { this.a = x; this.b = y; }

        public int Add()
                {return (this.a + this.b);}

        public static int Multiply(int x, int y)    # statikus metódus
                {return (x * y);}
}
"@
```

Saját osztály használata

- $object = New-Object Test 1,2 # Objektum létrehozása.
- $object.Add() # Metódushívás.
- $object.Multiply() # Hibára fut a statikus tag miatt.
- [Test]::Multiply(3,2) # Statikus metódus hívása.

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 04_01.txt

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

HTML és XML adatok kezelése PowerShellből

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

HTML jelentések készítése

- ConvertTo-Html:
  - CssUri: Stíluslap megadása.
  - Property: Jelentésben szerepeltetendő tulajdonságok.
  - Body: Törzs.
  - Head: Fejléc.
  - Title: Cím.
  - PostContent: A táblázat után szerepeltetendő szöveg.
  - PreContent: A táblázat előttszerepeltetendő szöveg.
  - Meta: meta-tag-ek megadása.
  - Charset: Karakterkészlet.
  - Transitional: sima HTML helyett XHTML.

Kimenet mentése XML-be

- ConvertTo-Xml
- Export-Clixml
- Import-Clixml

XML osztály és objektum-metódusok

- Get-Content C:\PowerShell\Products.xml
- [xml] $products = Get-Content C:\PowerShell\Products.xml
- $products.GetType().Fullname
- $products | Get-Member

XML osztály és objektum-metódusok

Administrator: Windows PowerShell

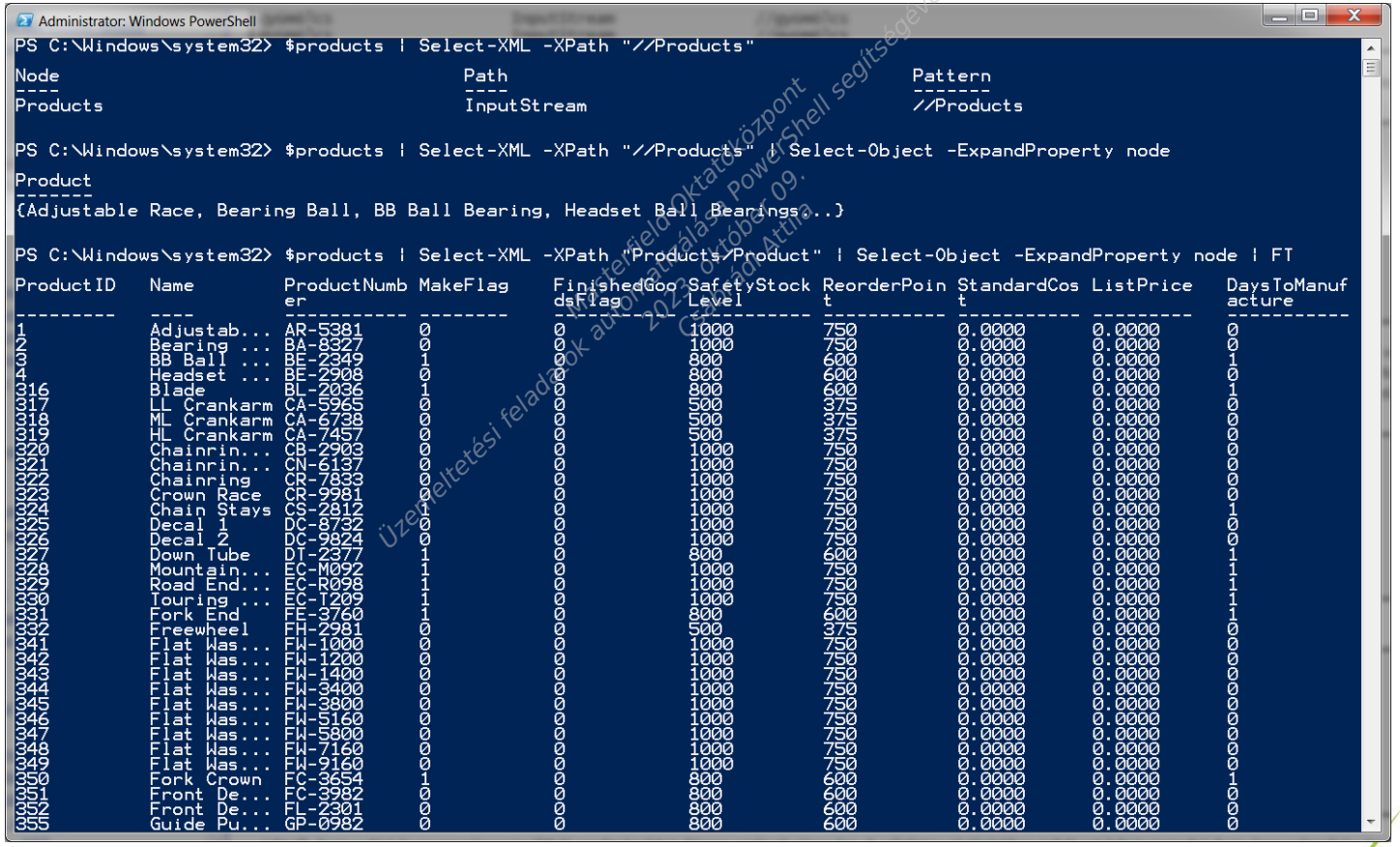
```
   TypeName: System.Xml.XmlDocument

Name                        MemberType            Definition
----                        ----------            ----------
ToString                    CodeMethod            static string XmlNode(psobject instance)
AppendChild                 Method                System.Xml.XmlNode AppendChild(System.Xml.XmlNode newChild)
Clone                       Method                System.Xml.XmlNode Clone()
CloneNode                   Method                System.Xml.XmlNode CloneNode(bool deep)
CreateAttribute             Method                System.Xml.XmlAttribute CreateAttribute(string name), System.Xml.X...
CreateCDataSection          Method                System.Xml.XmlCDataSection CreateCDataSection(string data)
CreateComment               Method                System.Xml.XmlComment CreateComment(string data)
CreateDocumentFragment      Method                System.Xml.XmlDocumentFragment CreateDocumentFragment()
CreateDocumentType          Method                System.Xml.XmlDocumentType CreateDocumentType(string name, string ...
CreateElement               Method                System.Xml.XmlElement CreateElement(string name), System.Xml.XmlEl...
CreateEntityReference       Method                System.Xml.XmlEntityReference CreateEntityReference(string name)
CreateNavigator             Method                System.Xml.XPath.XPathNavigator CreateNavigator()
CreateNode                  Method                System.Xml.XmlNode CreateNode(System.Xml.XmlNodeType type, string ...
CreateProcessingInstruction Method                System.Xml.XmlProcessingInstruction CreateProcessingInstruction(st...
CreateSignificantWhitespace Method                System.Xml.XmlSignificantWhitespace CreateSignificantWhitespace(st...
CreateTextNode              Method                System.Xml.XmlText CreateTextNode(string text)
CreateWhitespace            Method                System.Xml.XmlWhitespace CreateWhitespace(string text)
CreateXmlDeclaration        Method                System.Xml.XmlDeclaration CreateXmlDeclaration(string version, str...
Equals                      Method                bool Equals(System.Object obj)
GetElementById              Method                System.Xml.XmlElement GetElementById(string elementId)
GetElementsByTagName        Method                System.Xml.XmlNodeList GetElementsByTagName(string name), System.X...
GetEnumerator               Method                System.Collections.IEnumerator GetEnumerator()
GetHashCode                 Method                int GetHashCode()
GetNamespaceOfPrefix        Method                string GetNamespaceOfPrefix(string prefix)
GetPrefixOfNamespace        Method                string GetPrefixOfNamespace(string namespaceURI)
GetType                     Method                type GetType()
ImportNode                  Method                System.Xml.XmlNode ImportNode(System.Xml.XmlNode node, bool deep)
InsertAfter                 Method                System.Xml.XmlNode InsertAfter(System.Xml.XmlNode newChild, System...
InsertBefore                Method                System.Xml.XmlNode InsertBefore(System.Xml.XmlNode newChild, Syste...
Load                        Method                System.Void Load(string filename), System.Void Load(System.IO.Stre...
LoadXml                     Method                System.Void LoadXml(string xml)
Normalize                   Method                System.Void Normalize()
PrependChild                Method                System.Xml.XmlNode PrependChild(System.Xml.XmlNode newChild)
ReadNode                    Method                System.Xml.XmlNode ReadNode(System.Xml.XmlReader reader)
RemoveAll                   Method                System.Void RemoveAll()
RemoveChild                 Method                System.Xml.XmlNode RemoveChild(System.Xml.XmlNode oldChild)
ReplaceChild                Method                System.Xml.XmlNode ReplaceChild(System.Xml.XmlNode newChild, Syste...
Save                        Method                System.Void Save(string filename), System.Void Save(System.IO.Stre...
SelectNodes                 Method                System.Xml.XmlNodeList SelectNodes(string xpath), System.Xml.XmlNo...
SelectSingleNode            Method                System.Xml.XmlNode SelectSingleNode(string xpath), System.Xml.XmlN...
Supports                    Method                bool Supports(string feature, string version)
Validate                    Method                System.Void Validate(System.Xml.Schema.ValidationEventHandler vali...
WriteContentTo              Method                System.Void WriteContentTo(System.Xml.XmlWriter xw)
WriteTo                     Method                System.Void WriteTo(System.Xml.XmlWriter w)
Item                        ParameterizedProperty System.Xml.XmlElement Item(string name) {get;}, System.Xml.XmlElem...
Products                    Property              System.Xml.XmlElement Products {get;}


PS C:\Windows\system32> _
```

Keresés és mozgás az XML-ben

- Select-XML
  - Xpath: A keresőkifejezés megadása



Keresés és mozgás az XML-ben

- Saját metódusokkal
  - $products.Products
  - $products.Products.Product
  - $products.Products.Product | Format-Table –AutoSize
  - $products.SelectNodes("//Products")
  - $products.SelectNodes("//Products/Product[2]")
- Elem módosítás / hozzáadás:
  - .SetAttribute(…)
  - .AppendChild(…)

JSON kezelése

- Két cmdlet:
  - ConvertTo-JSON
    - Bármilyen parancs által létrehozott objektumhalmaz adatainak visszaadása JSON formátumban.
  - ConvertFrom-JSON
    - Pl. Invoke-WebRequest által visszaadott JSON formátumú adatok feldolgozása.

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 05_01.txt

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Hibakezelés és debugging

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

Hibatípusok a PowerShellben

- Terminating error - Megszakító hibák:
  - Rendszerszintű (pl. erőforrás) hibák.
- Non-terminating error - Nem megszakító hibák:
  - Felhasználói hibák: paraméterezés, értékadás stb.
- $error tömb:
  - Ide kerülnek a hibák, verem szerkezetű, FIFO.

Globális hibakezelési változók

Változó	Magyarázat
$Error	A korábban már látott hibajelzések tömbje.
$ErrorActionPreference	Globális hibakezelési mód: • Continue [default] - folytat, • Stop - megáll, • SilentlyContinue - figyelmeztetés nélkül továbbmegy, de az $Error tömbbe bekerül a hiba, • Inquire - rákérdez, • Ignore: - figyelmeztetés nélkül továbbmegy, az $Error tömbbe sem kerül be a hiba.
$MaximumErrorCount	Az $error tömb maximális mérete. Az ennél régebbi (nagyobb sorszámú) hibajelzések kihullanak a tömbből.
$ErrorView	A hibajelzések nézete: Normal vagy CategoryView.

CommonParameters

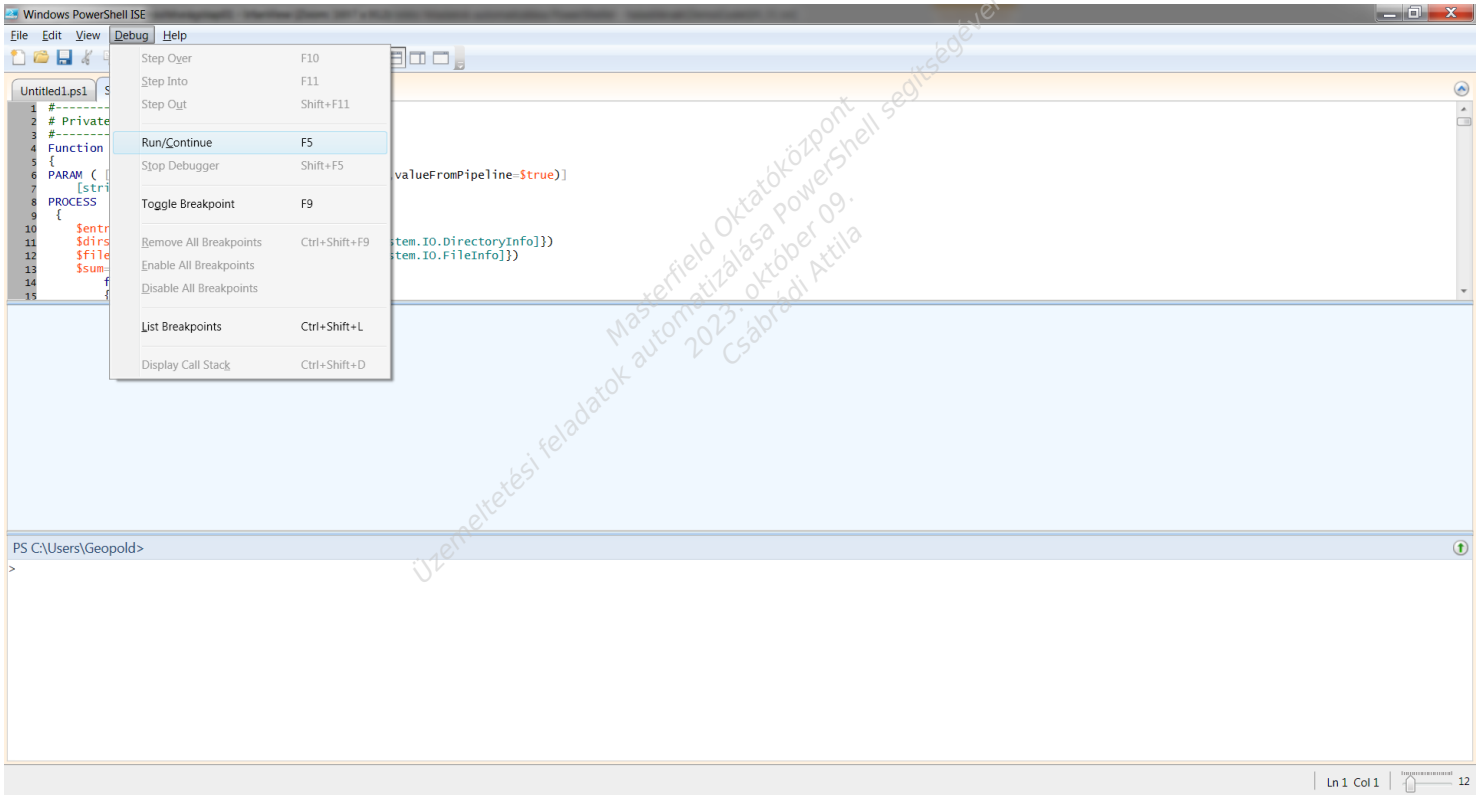
Paraméter	Magyarázat
Verbose	Bőbeszédes kimenetet ad a művelet lefolyásáról.
Debug	Hibakereső információkat ad, és interaktív módon lekezelhetők a hibák.
ErrorAction	Az előzőhöz hasonló, de nem csak interaktívan, hanem fixen beállítható hibakezelési mód: • Continue [default] - folytat, • Stop - megáll, • SilentlyContinue - figyelmeztetés nélkül továbbmegy, de az $Error tömbbe bekerül a hiba, • Inquire - rákérdez, • Ignore: - figyelmeztetés nélkül továbbmegy, az $Error tömbbe sem kerül be a hiba.
ErrorVariable	Saját hibaváltozónk neve ($ jel nélkül!). A $error tömb mellett ide is betöltődik a hibát leíró objektum.
OutVariable	A kimenetet ide tölti be.
OutBuffer	Az objektum-puffer mérete.

Debugging

- -Debug paraméter használata.
- -ErrorAction változó beállítása.
- Set-PSDebug –Trace:
  - Háttérben indítja a hibakereső motort, részletes lépésenkénti információt ír ki a program futásáról.
- Set-PSBreakpoint használata:
  - Beállít egy töréspontot:
    - Változóra.
    - Sorra.
- Debug-Job:
  - Leállítja a jobot és elindítja a hibakereső motort.

Debugging

- PowerShell ISE



Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 06_01.txt

Hibakezelés függvényben vagy scriptben

- Csapda (Trap)
  - Példa:
    - trap [ExceptionType]

```
#pl.: [System.Management.Automation.CommandNotFoundException]

      {
            művelet(ek)
      }
```

- Dobni és elkapni (Throw..Trap)
  - function fuggveny ($param = $(throw "Hiba")))

```
      {
            $param
      }
 trap { "Hibajelenség: $_ " }
```

Hibakezelés függvényben vagy scriptben

- Try… Catch... Finally.
- Példa:

```
Try
  {Művelet(ek)}
Catch [ExceptionType]
  {Specifikus hibakezelési művelet(ek)}
Catch
  {Általános hibakezelési művelet(ek)}
Finally
  {Művelet(ek)}
```

Hibák és információs üzenetek kiírása a kimenetre

- Write-Warning:
  - Figyelmeztető üzenet kiírása.
- Write-Error:
  - Hibaüzenet kiírása.
- Write-Verbose:
  - Részletes futási adatok kiírása.
  - $VerbosePreference = "Continue"
    - Alapértelmezés = SilentlyContinue.
- Write-Information:
  - Információs adatok kiírása:
    - ExecutionContext információk.
    - $InformationPreference = "Continue"
      - Alapértelmezés = SilentlyContinue.

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 06_02.txt

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

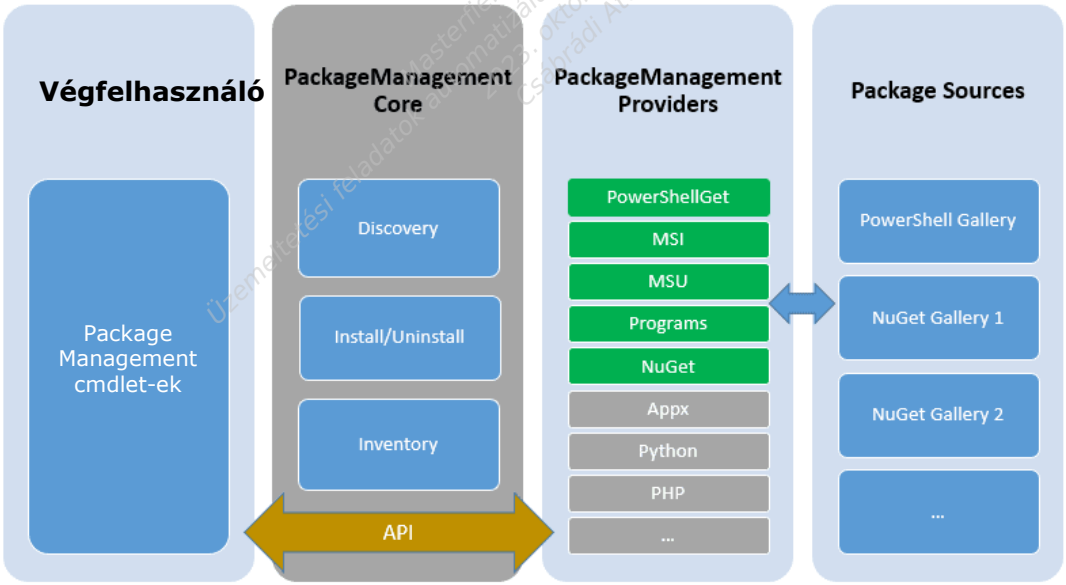
Függvénytár és scriptmodul készítés

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

PackageManagement modul

- Package management:
  - A csomagkezelési alrendszer lehetővé teszi, hogy szoftvertárolókból (PS Gallery, NuGet, stb.) telepítsünk fel szoftvercsomagokat.



Parancsok csomagkezeléshez

- PackageManagement parancsok:
  - Find-Package/PackageProvider
  - Get-Package/PackageProvider
  - Get/Set-PackageSource
  - Import-PackageProvider
  - Install/Uninstall-Package
  - Install-PackageProvider
  - Register/Unregister-PackageSource
  - Save-Package

Parancsok csomagkezeléshez

- PowerShellGet parancsok:
  - Find-Command/DscResource/Module/Script
  - Get-InstalledModule
  - Get-InstalledScript
  - Get/Set-PSRepository
  - Install/Uninstall-Module
  - Install/Uninstall-Script
  - Publish-Module/Script
  - Register/Unregister-PSRepository
  - Save-Module/Script
  - Update-Module/Script

Csomagszolgáltatók

- PowerShellGet
- nuGet
- Chocolatey
- Docker
- GitHub
- WinGet
- AppXGet

Gyakran használt modulok

- NTFSSecurity:
  - Jogosultságkezelési modul.
- PSExcel:
  - Modul az Excel fájlok kezeléséhez.
- PowerShell Universal:
  - Web alapú eszközök (pl. irányítópultok) fejlesztéséhez.
- PSWindowsUpdate:
  - Frissítések kezelése PowerShellből.
- Carbon:
  - Alkalmazás és konfigurációkezelés.
- PSCX:
  - Általános kiegészítések PowerShellhez.
- ShowUI:
  - Egyszerű felület (front-end) készítő PowerShell szkriptekhez.

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 07_01.txt

Függvénytár

- A függvénytár egy olyan script (.ps1), mely fejlett függvényeket, globális változókat, aliasokat tartalmaz.
- Függvénytár hívása (dot-sourcing)
  - . .\függvénytár.ps1

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 07_02.txt

Scriptmodul

- A scriptmodul egy **psm1** kiterjesztésű fájl, ami PowerShell scriptet tartalmaz. Ezt tudjuk legegyszerűbben létrehozni, hiszen akár egy már meglevő, bevált scriptünket egyszerű fájlátnevezéssel modullá tehetjük.
- Mivel tud többet egy scriptmodul, mint egy függvénytár ? Elsősorban a scriptben található függvények, változók és egyéb elemek láthatóságát, hozzáférhetőségét tudjuk kényelmesebben szabályozni a csak modulokban alkalmazható **Export-ModuleMember** cmdlet segítségével. A másik előny, hogy nem kell „dotsourcing” segítségével átemelni a script függvényeit és egyéb elemeit.

Scriptmodul

- Export-ModuleMember:
  - Megadható vele, hogy a modul a környezetbe történő importálásakor mely funkciókat tegye elérhetővé (publikussá) a felhasználók számára
- New-ModuleManifest:
  - Moduljegyzék készítése, ami a modul metaadatait és az exportálandó funkciókat tartalmazza.
- Import-Module modul.psm1 -AsCustomObject –Force:
  - Modul objektumként történő importálása
- About help készítése modulhoz:

```
PS C:\> Get-Tree -path C:\PowerShell\Modul
Modul
├────EN-us
│       about_modul.help.txt
└────HU-hu
        about_modul.help.txt
   modul.psm1
```

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 07_03.txt

Jegyzékmodul

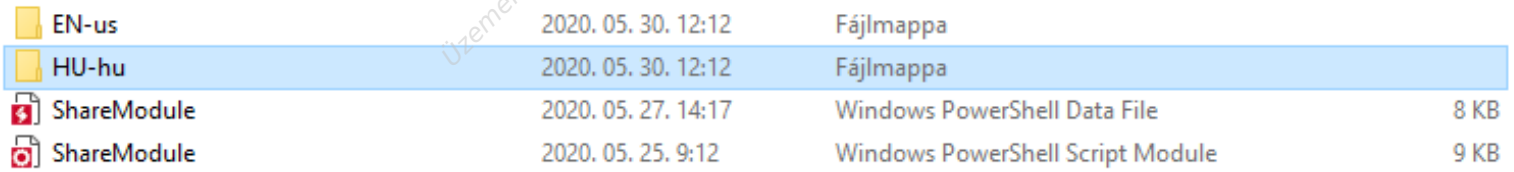
- Lehetőség van olyan modul létrehozására is, amiben nincsen modulfájl, csak jegyzékfájl és esetleg .NET építőelemek.
  - RootModule = 'ScriptModule.psm1'
  - NestedModules = ’module1’,’module2’
- Ezzel a lehetőséggel egy már létező modulhoz akár többfajta jegyzékfájllal más és más exportált változókat, függvényeket és cmdleteket lehet definiálni.

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 07_04.txt

About súgó modulhoz

- A modul általános leírását célját tartalmazhatja.
- Lekérdezni a Get-Help about_<Modulnév> paranccsal lehet.
- Minden támogatandó nyelven elkészíthető.



Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 07_05.txt

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

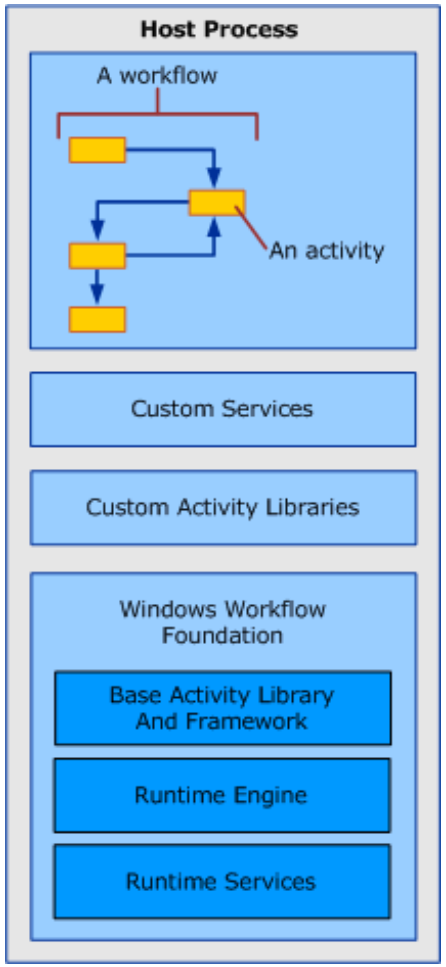
Munkafolyamatok

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

Munkafolyamatok

- A PowerShell 3.0 verziótól új lehetőség jelent meg szkriptek futtatására: a workflow (munkafolyamat).
- Hasonlít a függvényhez.
- Speciális tulajdonságok:
  - hosszú futásúak,
  - ismételhetők,
  - párhuzamosíthatók,
  - megszakíthatók és folytathatók,
  - megállíthatók és újraindíthatók.
- Valójában a .NET-hez tartozó Workflow Foundation hajtja végre:
  - Activity-k, nem parancsok.
  - XAML leírófájl.



Munkafolyamatok

- XAMLDefinition tulajdonság:
  - Valójában ez az XML rész hajtódik végre a Windows Workflow Foundationben található workflow motor segítségével. Azaz fontos megjegyezni, hogy workflow-k esetében PowerShell szintaxissal valójában workflow-t írunk.
- Scriptblock:
  - Beépített paraméterekkel rendelkezik, amelyek lehetővé teszik, hogy automatikusan rendelkezzen számos olyan speciális lehetőséggel, amit nem nekünk kell leprogramozni, hanem gyárilag tudja a workflow. Ez a bonyolult szkript fogja valójában megszólítani a Windows Workflow Foundationt és a StartWokflowApplication metódus hívásával átadja neki a XAML workflow definíciót.

Függvény vs. munkafolyamat

Függvény	Munkafolyamat
Powershell által végrehajtva.	Munkafolyamat-motor által végrehajtva.
Naplózás és újraindítás bonyolult kódolással .	Naplózás és újraindítás a munkafolyamat-motor része.
Sorfolytonos művelet feldolgozás.	Támogatja a párhuzamosságot.
Egyben fut le.	Futtatható / szüneteltethető / újraindítható.
Adatvesztés lehetséges hálózati problémák esetén.	Az adatok fennmaradhatnak hálózati problémák esetén.
Teljes nyelvkészlet és szintaxis.	Korlátozott nyelvkészlet és szintaxis.
Cmdletek futtatása.	Tevékenységek futtatása.

Párhuzamosítás

- Párhuzamos végrehajtás munkafolyamatban:

```
Workflow ForeachParallelTest
{
   param([string[]]$computers)
   foreach –parallel ($computer in $computers)
      {
         Get-WmiObject –Class Win32_OperatingSystem
         –PSComputerName $computer
      }
}
```

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 08_01.txt

Ellenőrzőpontok

- CheckPoint-WorkFlow cmdlet.
- Naplózás.
- Eredmények mentése:
  - $Env:UserProfile\AppData\Local\Microsoft\Windows \PowerShell\WF\PS\default
- Felfüggeszthető / újraindítható.
- Hálózati kimaradás/újraindítás esetén folytatódik az ellenőrzőponttól.

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 08_02.txt

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

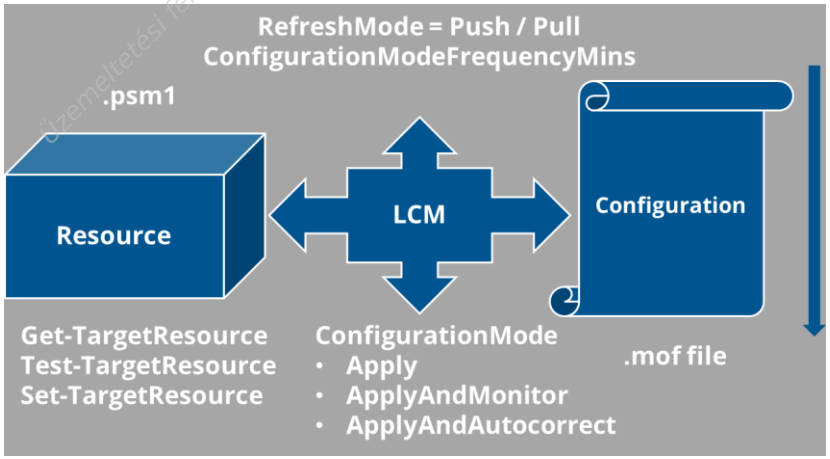
Desired State Configuration

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

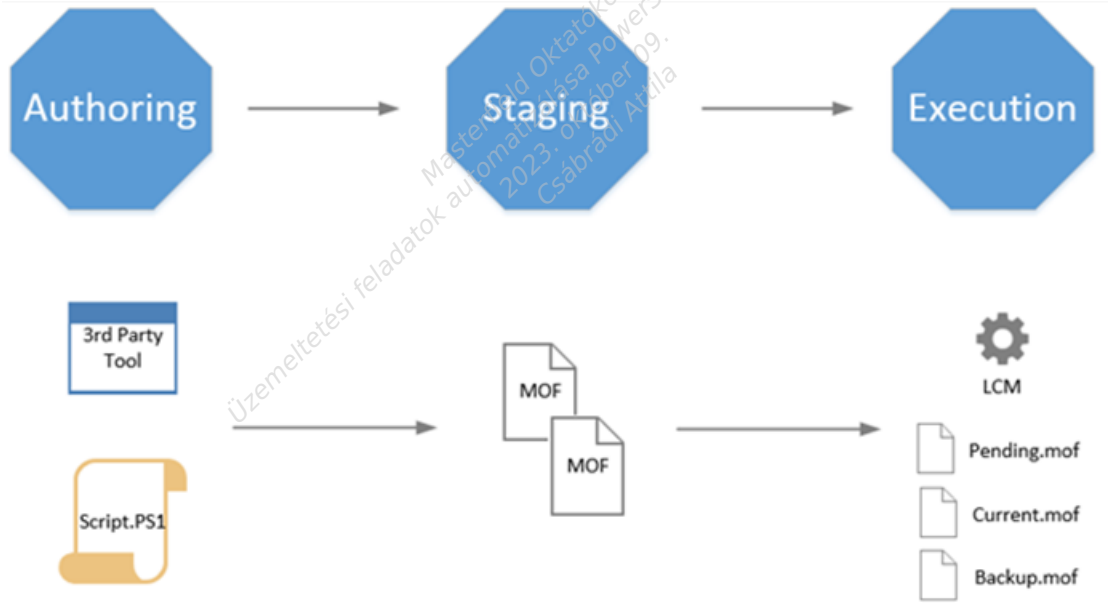
Desired State Configuration

- A DSC egy újabb automatizálási lehetőség, mellyel megfogalmazhatjuk az általunk menedzselt gépek kívánt állapotát és azt automatikusan fenn is tarthatjuk.
- A DSC három fő komponensből áll:
  - Konfigurációk.
  - Erőforrások.
  - Helyi konfigurációs motor (LCM).



Desired State Configuration

C:\Windows\System32\Configuration



Desired State Configuration - komponensek

- PSDesiredStateConfiguration modul:
  - Get-Command -Module PSDesiredStateConfiguration
- Windows PowerShell Desired State Configuration Service:
  - „Pull” üzemmódú konfigurációs telepítő.
- Erőforrások:
  - Get-DscResource
- Local Configuration Manager:
  - MOF (Management Object Format) fájlok.
  - Get-DSCLocalConfigurationManager

Desired State Configuration - erőforrások

- Erőforrások szintaxisa:
  - Get-DscResource -Name Resource –Syntax
- Erőforrásmodulok:
  - Modules\PSDesiredStateConfiguration\DSCResources
    - Get-TargetResource
    - Set-TargetResource
    - Test-TargetResource

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 09_01.txt

Desired State Configuration

- Konfiguráció:
  - Get-DscConfiguration
  - Test-DscConfiguration
  - Start-DSCConfiguration
- Konfiguráció visszavonása:
  - Ensure = ”Present” / ”Absent”

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 09_02.txt

Desired State Configuration

- Konfiguráció

```
Configuration Webconfig
{
   Node localhost
   {
                  WindowsFeature IIS
                  {
                        Ensure =
"Present"/"Absent"
                        Name =
"Web-Server"
                  }
            }
}
```

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 09_03.txt
- Start-Demo 09_04.txt

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Eseménykezelés

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

WMI események

- A WMI objektumokon regisztrálhatóak eseménykezelők.
  - A megadott esemény kiváltja a scriptblok lefutását.
    - Register-WmiEvent
    - Unregister-WmiEvent
    - Get-EventSubscriber
- Perzisztens WMI eseménykezelők:
  - EventFilter
  - EventConsumer
  - EventBinding

Egyéb események

- PowerShell események:
  - New -Event
  - Register-EngineEvent
- .NET események:
  - Register-ObjectEvent
  - Pl.: Timer, FileSystemWatcher stb.

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 10_01.txt

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Adatbázisok elérése PowerShellből

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

Excel adatok elérése

- COM objektumon keresztül:
  - $excel = New-Object -COMObject Excel.Application
- Láthatóság:
  - $excel.Visible = $true
- Új munkafüzet megnyitása:
  - $wb=$excel.Workbooks.Add()
- Adatok kiírása Excelbe:
  - $worksheet = $workbook.Worksheets.Item(1)
  - $range = $worksheet.Cells.Item(1,1)
  - $row = 1
  - $s = Get-Process | Select-Object name
  - $s | foreach -process {$range = $worksheet.Cells.Item($row,1);$range.value2 = $_.Name;$row++ }
  - $excel.DisplayAlerts = $False
  - $workbook.SaveAs("C:\Temp\Get_Process.xls")
  - $excel.Quit()

Excel adatok elérése

- PSExcel modulon keresztül:
  - Nem kell hogy telepítve legyen az Excel.
  - OfficeOpenXml.ExcelPackage objektum.
- Cmdletek:
  - New-Excel
  - Save-Excel
  - Close-Excel
  - Export-XLSX
  - Import-XLSX
  - Format-Cell
  - Get-CellValue
  - Get-Workbook
  - Get-Worksheet

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 11_01.txt
- Start-Demo 11_02.txt

Acess adatok elérése

- Kapcsolat felépítése
  - $connection = New-Object System.Data.OleDb.OleDbConnection("Provider=Microsoft.ACE.OLEDB.1x.0; Data Source=$path")
  - $connection.Open()
- Adatok olvasása
  - $sql = "SELECT * FROM table"
  - $cmd = New-Object System.Data.OleDb.OleDbCommand($sql, $connection)

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 11_03.txt

SQL adatok elérése

- WMI-n,
- .NET osztályokon,
  - ODBC-n,
  - SQL Server Management Objects (SMO)-en,
- SQL PS Provideren keresztül.

SQLPS

- Beépített PS mini-shell: sqlps.exe.
- SMO-ra épül (SMO objektumokat kapunk eredményül).
- Cmdlet-ek:
  - Encode-SqlName
  - Decode-SqlName
  - Invoke-Sqlcmd
  - Invoke-PolicyEvaluation
  - Convert-UrnToPath
- SQLSERVER Provider:
  - Set-Location SQLSERVER:\SQL\localhost\DEFAULT\Databases
  - Kontextust átadja a hívó scriptnek!

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 11_04.txt

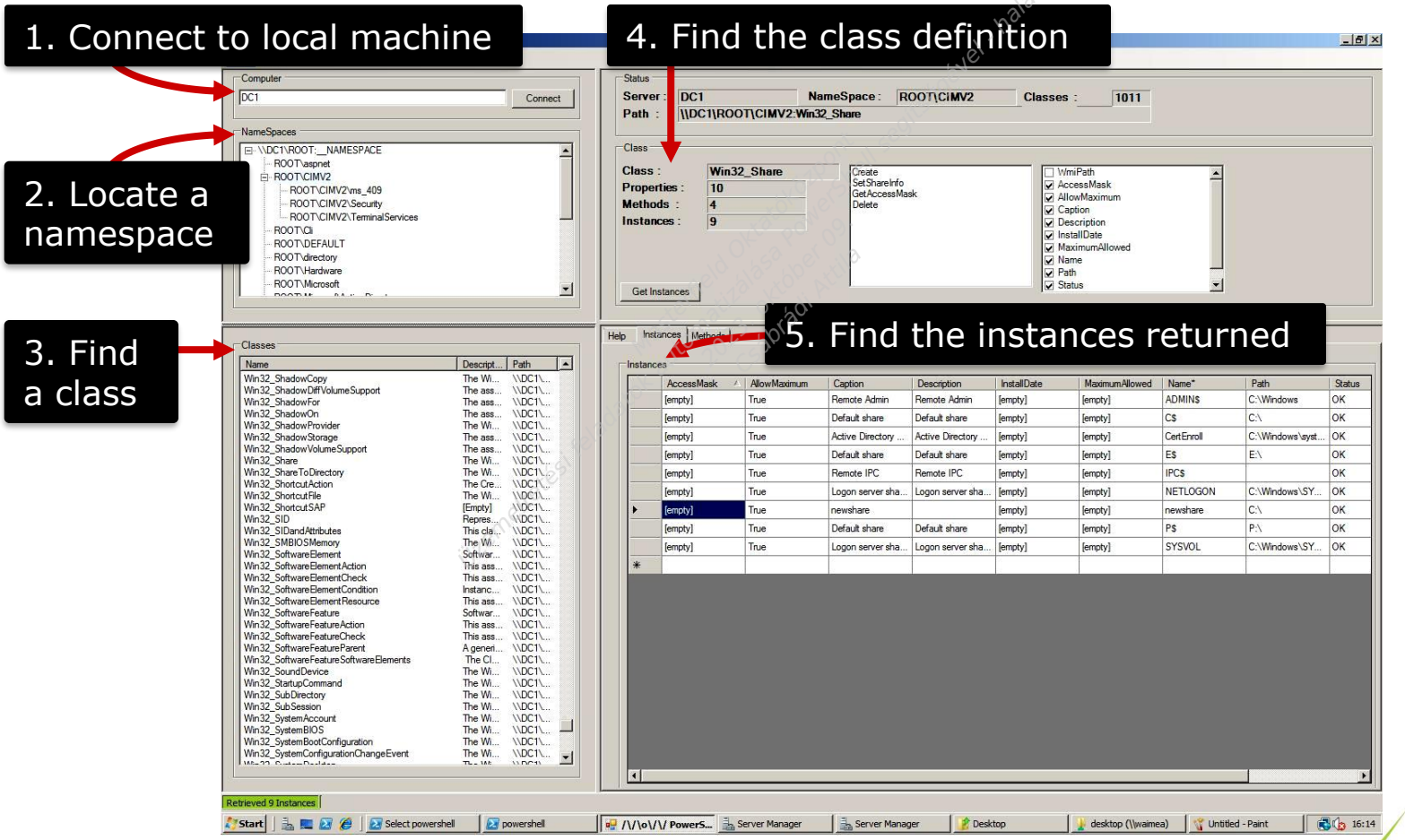
Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Grafikus felület tervezése

Masterfield Oktatóközpont
http://www.masterfield.hu

Példa: a WMI Explorer



GUI elemek - WinForms

- [System.Reflection.Assembly]::LoadWithPartialName ("System.Windows.Forms")
  - Add-Type -Assembly System.Windows.Forms
- [System.Reflection.Assembly]::LoadWithPartialName ("System.Drawing")
- $frmMain = new-object Windows.Forms.Form
- $frmMain.Size = new-object System.Drawing.Size @(800,600)
- $frmMain.text = "PowerShell WMI Explorer"

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 12_01.txt
- Start-Demo 12_02.txt

GUI elemek - WPF

- XAML leírás az űrlaphoz és a komponensekhez:
  - Add-Type –Assembly PresentationFramework
- XAML formátum:

```
<?xml version="1.0"?>
- <Window x:Name="Window" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  - <Grid x:Name="Grid">
    - <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
      </Grid.RowDefinitions>
    - <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="Auto"/>
      </Grid.ColumnDefinitions>
      <TextBox x:Name="PathTextBox" Width="150" Grid.Row="0" Grid.Column="0"/>
      <Button x:Name="ValidateButton" Grid.Row="0" Grid.Column="1" Content="Validate"/>
      <Button x:Name="RemoveButton" Grid.Row="1" Grid.Column="0" Content="Remove"/>
    </Grid>
</Window>
```

- Scriptből az XML felolvasása értelmezése:
  - $reader = (New-Object System.Xml.XmlNodeReader $xaml)
  - $window = [Windows.Markup.XamlReader]::Load($reader)

Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 12_03.txt

Diagram-jelentések készítése

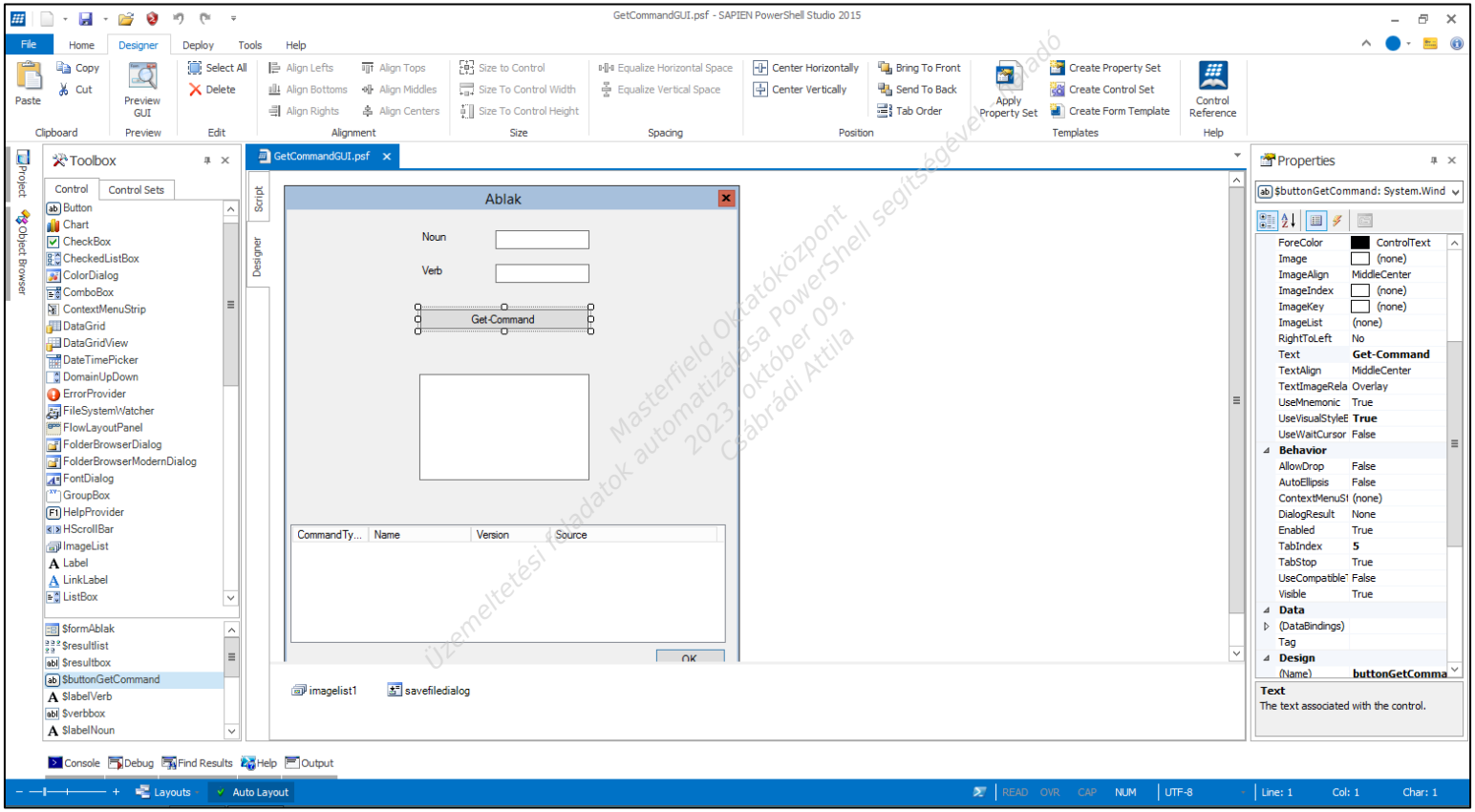
- System.Windows.Forms
- System.Windows.Forms.DataVisualization
- $Chart = New-object System.Windows.Forms.DataVisualization.Charting.Chart
- $Chart | Get-Member

Gyakorlat

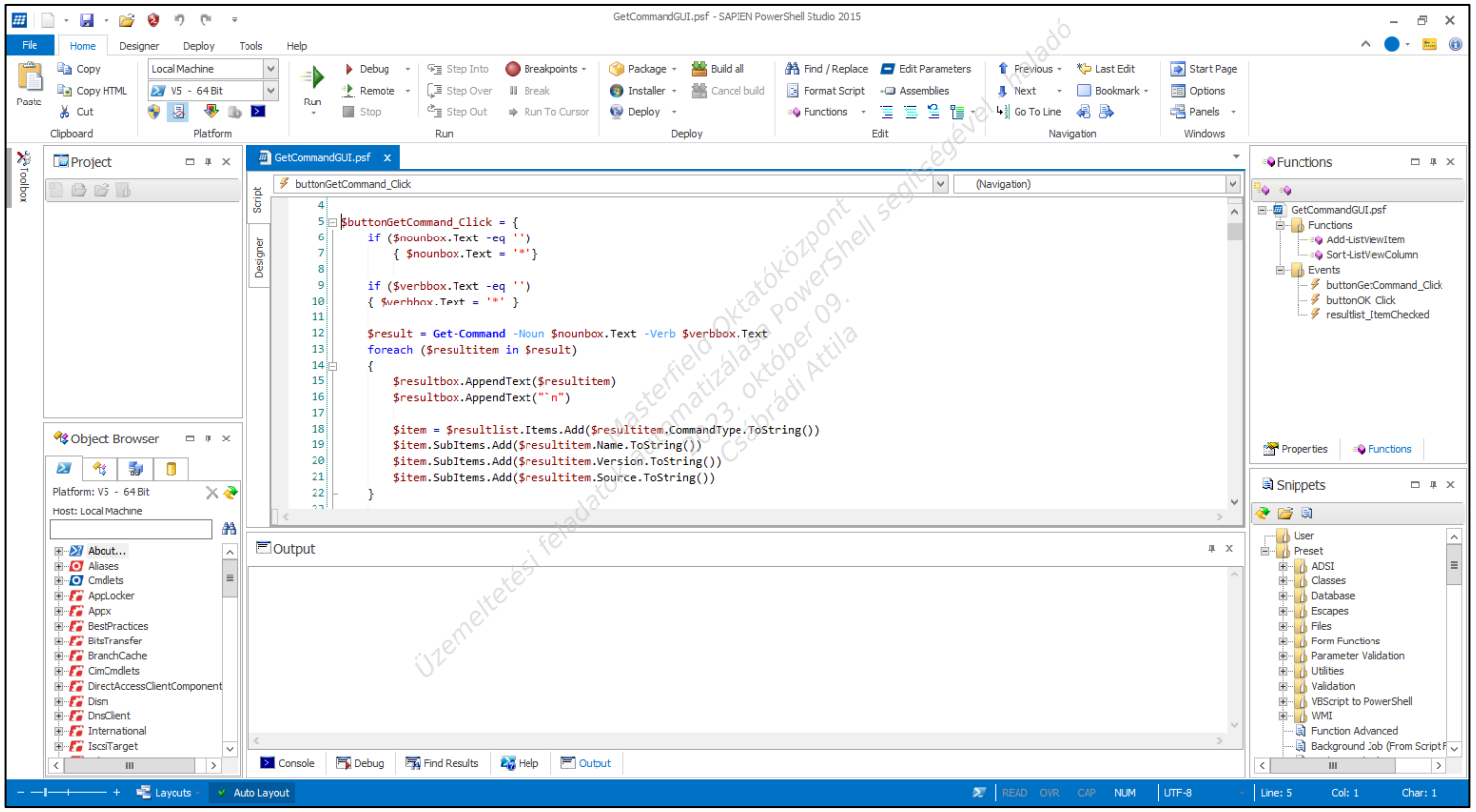
- . .\StartDemo.ps1
- Start-Demo 12_04.txt

PoshGui.com

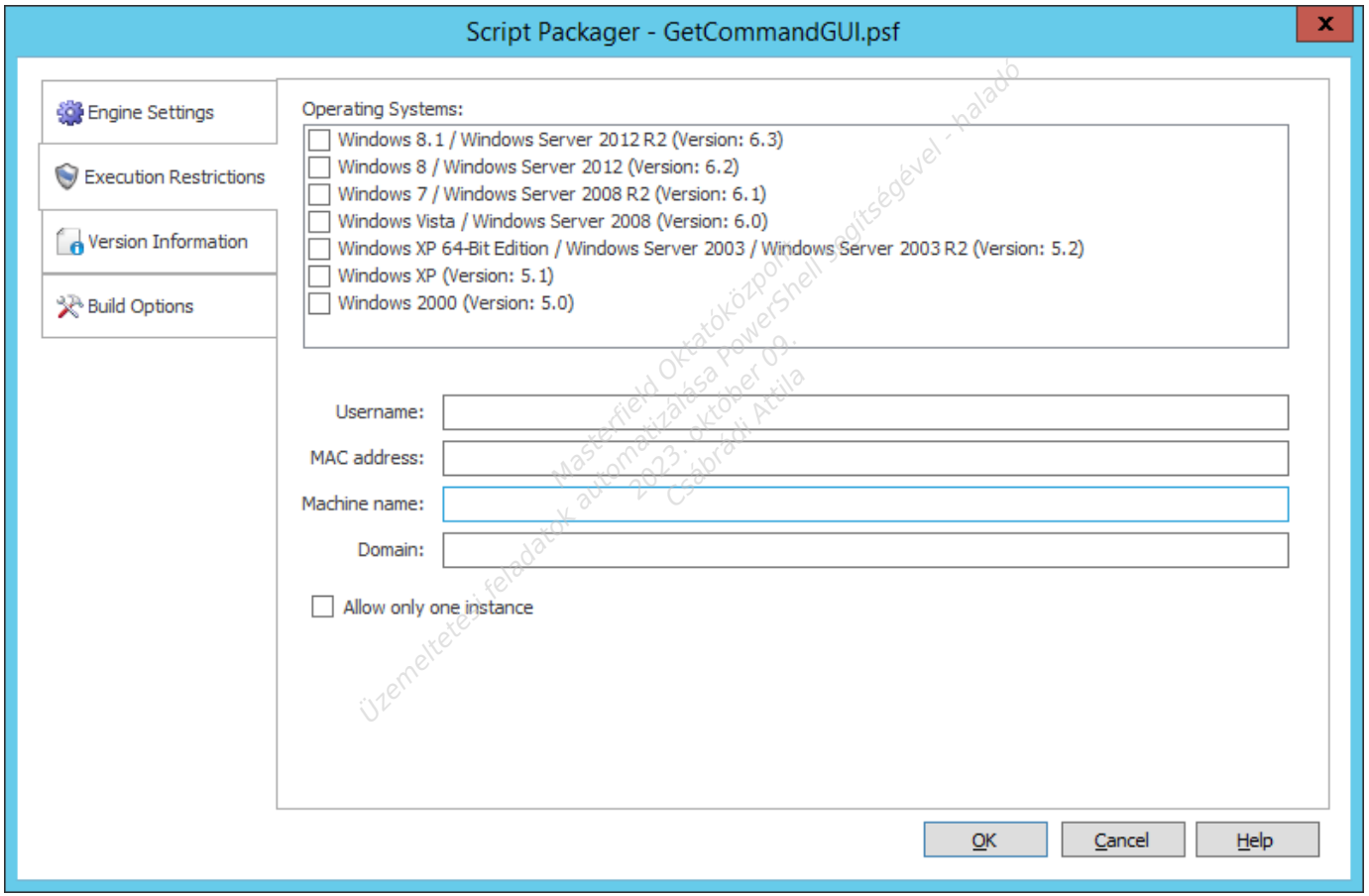
Powershell Studio



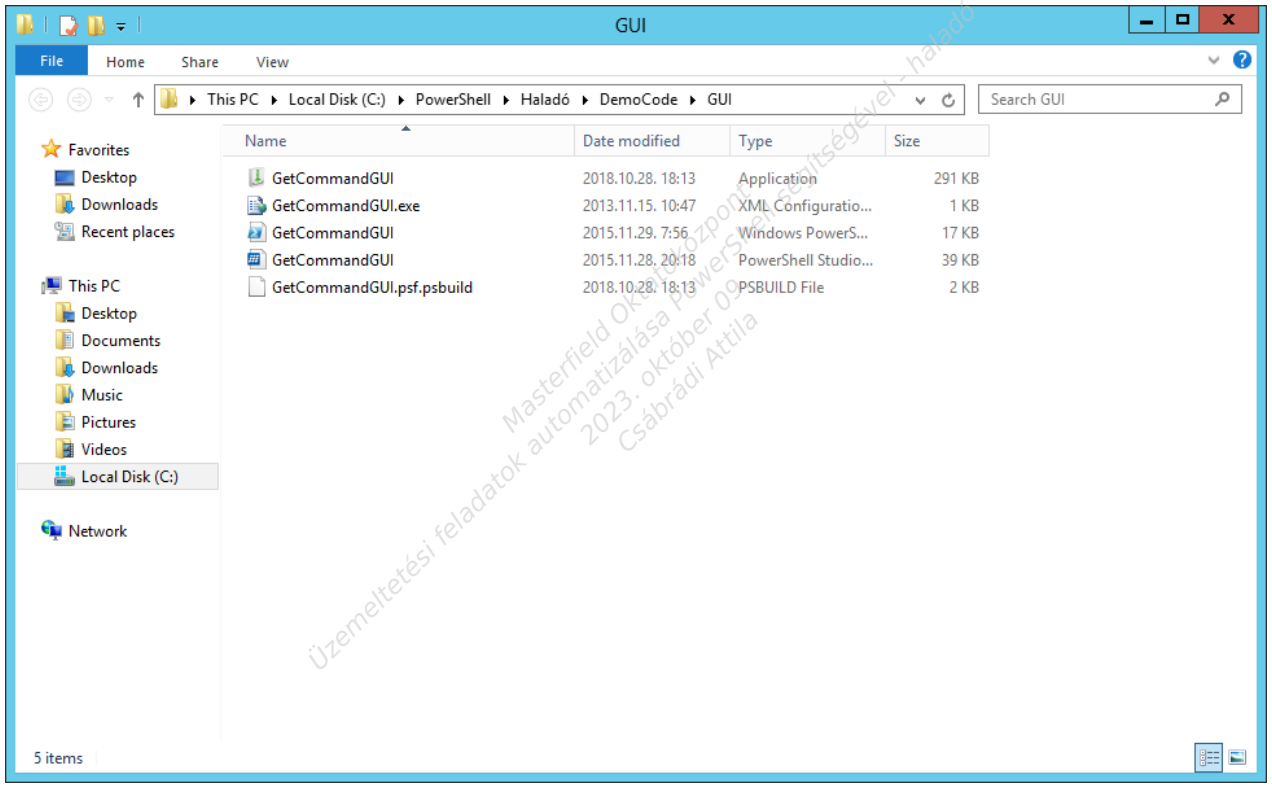
Powershell Studio



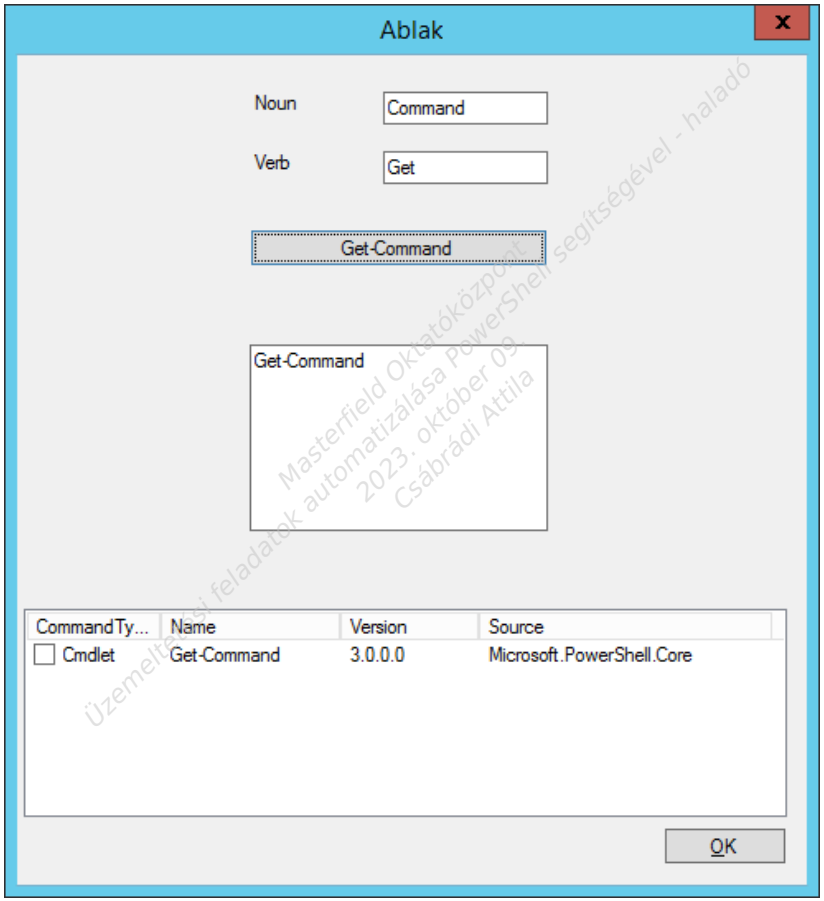
Powershell Studio



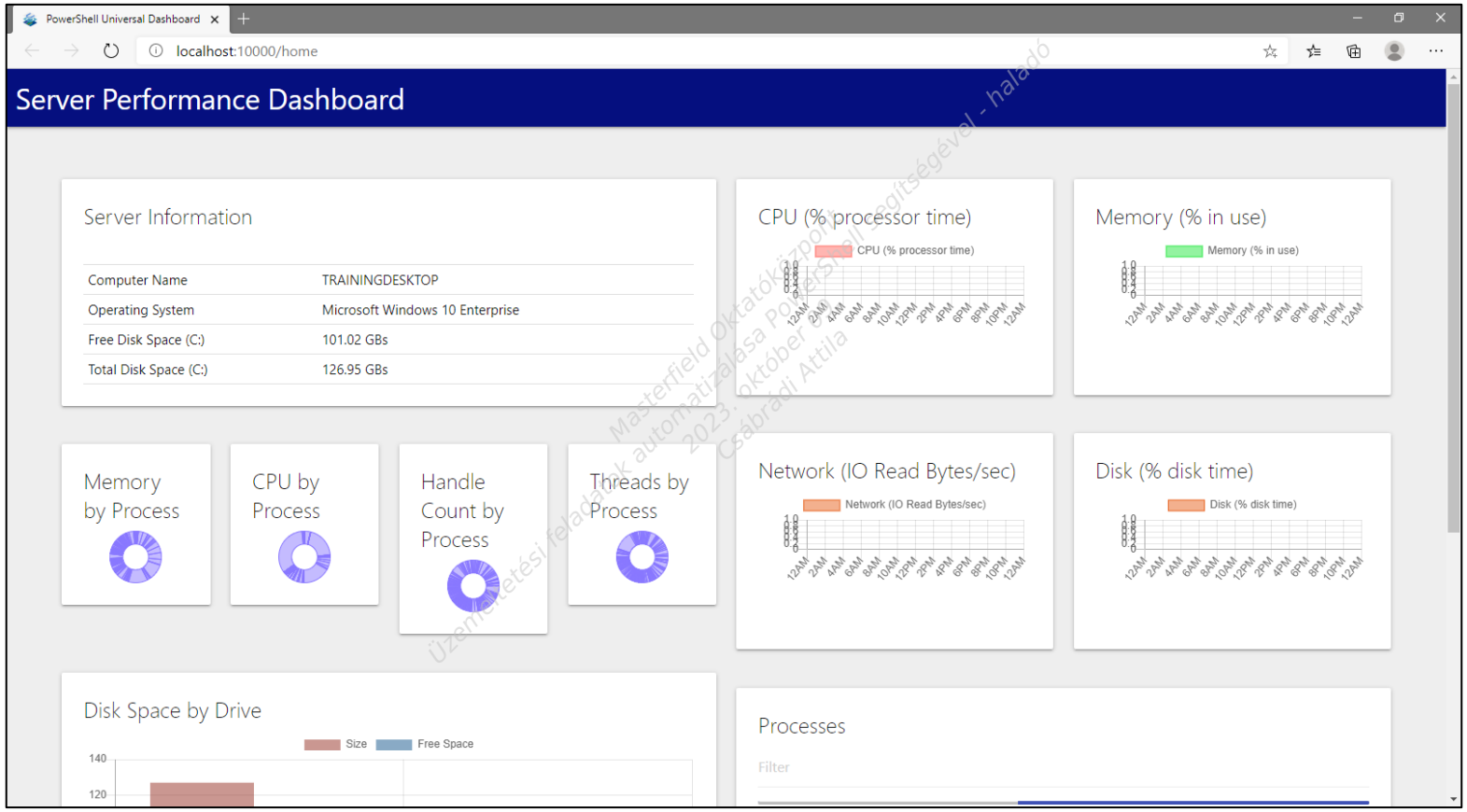
Powershell Studio



Powershell Studio



Universal DashBoard



Gyakorlat

- . .\StartDemo.ps1
- Start-Demo 12_05.txt

És végül...

... kérem, tegyék fel kérdéseiket!

Köszönjük, hogy meghallgatták előadásunkat!

További tanfolyamok és információ:

www.masterfield.hu