Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

PowerShell haladó - tartalomjegyzék

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

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	Alapvetően a példányokkal és a statikus metódusokkal dolgozunk
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

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

Csővezeték (pipe)

- Parancs | parancs
  - Parancsok kimenete: objektum!
  - Parancsok bemenete: objektum!
- Formázás
  - Objektumok alapértelmezett kimenete
  - Format-List; Format-Table

PSDrive-ok

- Különböző típusú adatok egységes elérésére
- Providerek
  - FileSystem
  - Registry
  - CertificateStore
  - Alias
  - Environment
  - Function
  - Variable
- Get-Command –noun PSDrive

Változók

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

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

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:\>. .\start-demo.ps1).

Tömbök

- Skalár változó vs. Tömbváltozó
- Azonos elemeket tartalmazó tömbök
  - [int32[]] $IA = 1500,2230,3350,4000
- Nem azonos elemeket tartalmazó tömbök
  - $FSA = Get-ChildItem
- Többdimenziós tömbök
  - $table = (1,2,3,4),("a","b","c","d")
- Asszociatív tömbök
  - $hash = @{ Név = "Gipsz Jakab"; Cím = "Budapest"; "e-mail"="jgipsz@domain.local"}

Dátumok

- DateTime osztály
  - Metódusok:
    - AddDays .. AddTicks
    - CompareTo
    - Parse
    - ToLongDateString .. ToShortTimeString
- (Get-Date).AddYears(10).DayOfWeek

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
- Típusvizsgálati
  - -is

Reguláris kifejezések

- -like; notlike operátorok
  - "ablak","abrosz","alma","auto" –like "[a-d]b?*„
- -match; notmatch operátorok
  - "ab" -match "[^a]|[^b]"
- System.Text.RegularExpressions.Regex osztály
  - Metódusok:
    - Match
    - Matches
    - Split
    - Replace

Reguláris kifejezések

RegEx	Matches	Example
.	One instance of any character	.o.th
[xyz]	One instance of the set	[CMRS]andy
[x-z]	One instance of the range	[A-Z]eramy
^	The beginning of the string	^Subject:
$	The end of the string	meeting$
*	Zero or more of the preceding	W.*s
+	One or more of the preceding	[MZ]+any
?	Zero or one of the preceding	[MZ]?any
\	Escapes a special character	Try\$

Függvény

- Névvel rendelkezik
- Visszatérési értéke van
- A PowerShell-ben:
  - Function
  - Filter

Függvénydefiníció

- Egyszerű definíció
  - Function függvény (paraméter(ek)) { művelet(ek) }
- 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!

Függvényparaméterek

- Függvénydefiníció
  - Function terület ($a,$b){ return $a*$b }
- Függvényhívás
  - $c = terület 2 3
  - $c = terület –b 2 –a 3
  - $c = terület(2,3) #nem jó!
- Cím szerint paraméterátadás
  - Function dupláz ([ref]$a) { $a=$a*2}
  - dupláz ([ref]$c)

Paraméterinicializálás

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

Változó számú paraméter

- Paraméterlista
  - $args tömbváltozó
  - Function kiír
  - {
  - If ($args)
  - {
  - foreach ($arg in $args)
  - {Write-Host $arg }
  - }
  - }

Láthatóság (Scope)

- Private: $változó
- Get-Variable –Scope 1
- Get-Variable –Private –Scope 1

- function global:első { "első" }
- . C:\Temp\Scripts\script.ps1

ScriptBlock

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

Scriptek

- Fájlba mentett műveletsor
- Kiterjesztése .ps1 (PowerShell 1.0)
- Megjegyzés # karakter használatával
- Soronkénti végrehajtás!
  - Ha nem kerül rá a vezérlés, nem ad hibajelzést!
- Engedélyezés
  - Get-ExecutionPolicy
  - Set-Executionpolicy Restricted
  - AllSigned
  - RemoteSigned
  - Unrestricted

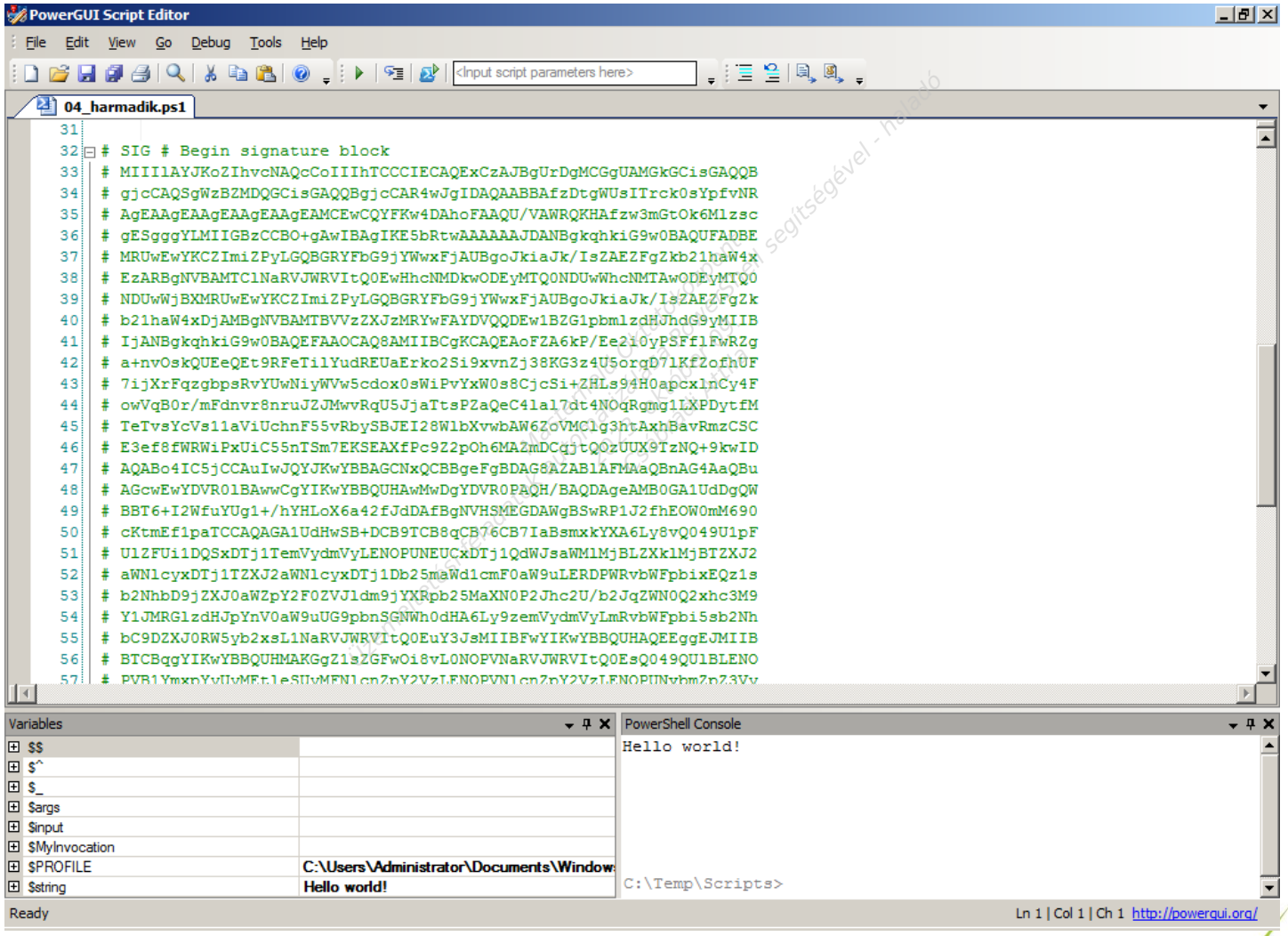
Scriptek futtatása

- 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 (&).

Script paraméterek

- $args tömb használata
  - if ($args.Length -ne 3)
    - {
    - Write-Error "A szkript csak 3 paraméterrel indítható!"
    - return ”Hibás futás!”
    - }
- param blokk használata
  - param ($a, $b)
  - $a / $b

Scriptek digitális aláírása



Hibatípusok a PowerShellben

- Terminating error
  - Megszakító hibák
    - Nullával való osztás
    - Szintaktikai hibák
- Non-terminating error
  - Nem megszakító hibák
    - Cmdlet-ek paraméterezési hibái
- $error tömb!

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, Inquire - rákérdez.
ErrorVariable	Saját hibaváltozónk neve ($ jel nélkül!). A $error változó mellett ide is betöltődik a hibát leíró objektum.
OutVariable	Kimenetet ide tölti be.
OutBuffer	Az objektum-puffer mérete, ennyi elemet „magában” tart, mielőtt továbbítja az outputot a következő csőszakasznak

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, Inquire - rákérdez.
$MaximumErrorCount	Az $error tömb maximális mérete. Az ennél régebbi (nagyobb sorszámú) hibajelzések kihullanak a tömbből.
$ErrorView	A hibajelzések nézete: Normal vagy CategoryView

Hibakezelés függvényben vagy scriptben

- Csapda (Trap)
  - Példa:
    - trap [ExceptionType]
      - # pl.: System.DivideByZeroException
    - { művelet(ek) }
- Dobni és elkapni (Throw..Trap)
  - trap { "Hibajelenség: $_ " }
  - function dupla ($v = $(throw (New-Object System.ArgumentException -arg "Adjál meg valamit, amit duplázni lehet!")))
  - {
  - $v*2
  - }

Hibakezelés függvényben vagy scriptben

- Try..Catch..Finally
- Példa:

```
Try
{
    $AuthorizedUsers = Get-Content \\FileServer\HRShare\UserList.txt -ErrorAction
    Stop
}
Catch [System.OutOfMemoryException]
{
    Restart-Computer localhost
}
Catch
{
    $ErrorMessage = $_.Exception.Message
    $FailedItem = $_.Exception.ItemName
    Break
}
Finally
{
    $Time=Get-Date
    "This script made a read attempt at $Time" | out-file c:\logs\ExpensesScript.log -
    append
}
```

És végül...

... kérem, tegyék fel kérdéseiket!

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Haladó technikák

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

PowerShell haladó - tartalomjegyzék

Snapinek

- PowerShell 1.0
  - Providerek
  - Cmdlet-ek
- .NET dll-ekben
  - HKEY_LOCAL_MACHINE
    - Software\Microsoft\PowerShell\1
      - PowerShellSnapIns
- Get-PSSnapin
- Add-PSSnapin
- Remove-PSSnapin

Modulok

- PowerShell 2.0
  - Providerek
  - Cmdlet-ek
  - Aliasok
  - Függvények
  - Formátumok
- Modulkönyvtárakban
  - %windir%\system32\WindowsPowerShell\v1.0\Modules
  - %home%\Documents\WindowsPowerShell\Modules
- Get-Module
- Import-Module
- Remove-Module

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)

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

És végül...

... kérem, tegyék fel kérdéseiket!

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Haladó WMI technikák

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

PowerShell haladó - tartalomjegyzék

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	Alapvetően a példányokkal és a statikus metódusokkal dolgoozunk
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

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 osztály és objektumkapcsolatok

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

WMI események

- A WMI objektumokon regisztrálhatóak eseménykezelők
- A megadott esemény kiváltja a scriptblok lefutását
- Register-WmiEvent
- Unregister-WmiEvent
- Get-EventSubscriber

És végül...

... kérem, tegyék fel kérdéseiket!

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Formátumbeállítás

PowerShell haladó - tartalomjegyzék

Kimeneti formátum módosítása

- Alapértelmezett kimeneti formátum
  - C:\WINDOWS\system32\windowspowershell
    - v1.0\types.ps1xml
- Saját tulajdonság hozzáfűzése
  - Get-ChildItem C:\Windows| Format-Table name,@{Expression={if($_.psiscontainer){"Könyvtár"}else{"Fájl"}};Label="Típus";width=10}
  - Get-ChildItem C:\Windows | Select-Object name, @{Name="Típus";Expression={if ($_.Extension -eq ".log"){"Naplófájl"}else{"Reguláris fájl"}}}

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

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

Diagram-jelentések készítése

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

És végül...

... kérem, tegyék fel kérdéseiket!

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

XML adatok kezelése PowerShellből

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

PowerShell haladó - tartalomjegyzék

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

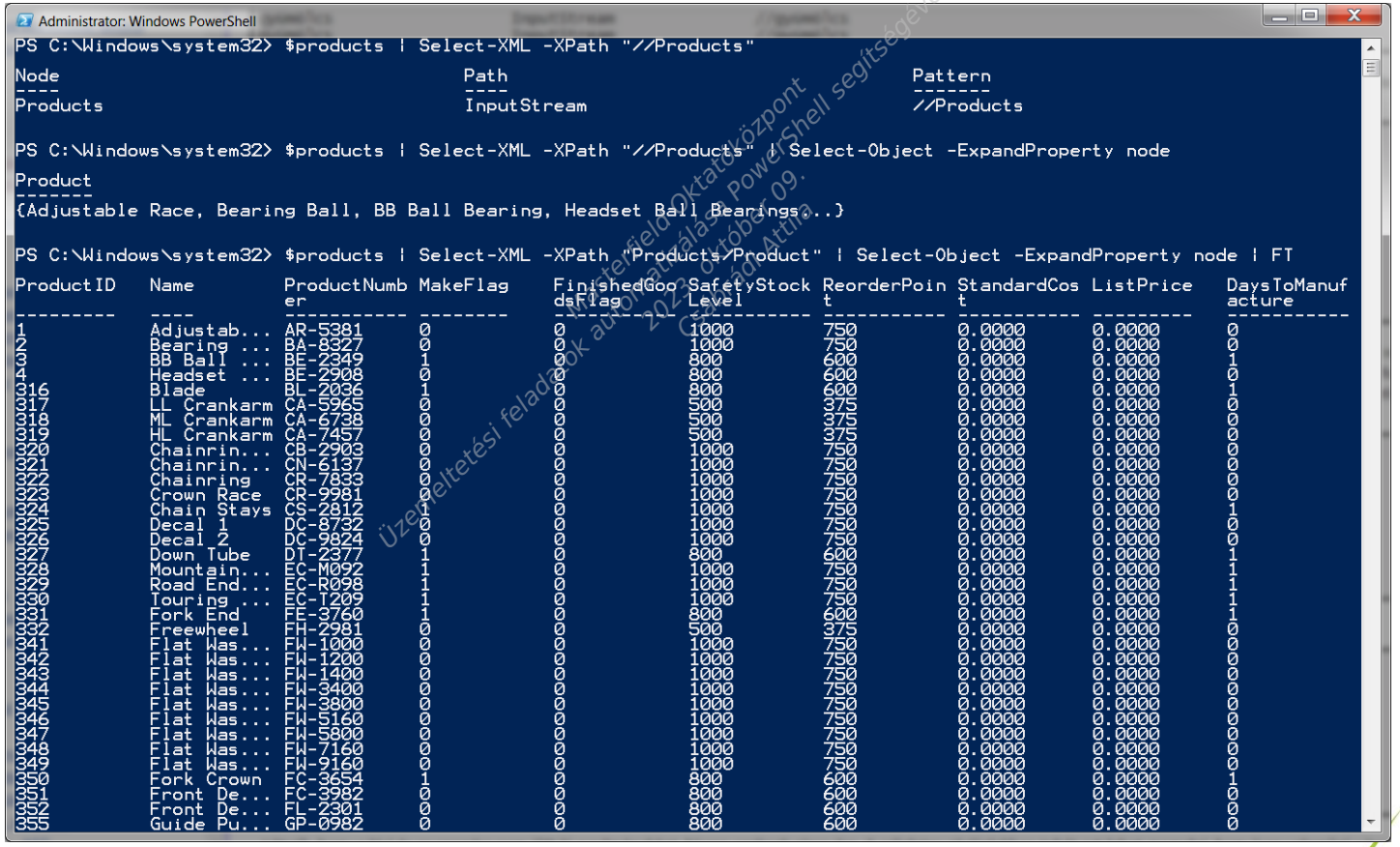
```
   TypeName: System.Xml.XmlDocument

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]")

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(…)

És végül...

... kérem, tegyék fel kérdéseiket!

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Haladó függvénykészítés

Masterfield Oktatóközpont
http://www.masterfield.hu

PowerShell haladó - tartalomjegyzék

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 aparaméterével

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 }
}
```

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
#>
```

Proxy függvény

- A Proxy függvény egy már létező cmdlet „klónozásával” és kiegészítésével létrejövő függvény.
  - $metadata=New-Object System.Management.Automation.CommandMetaData (Get-Command cmdlet)
  - [Management.Automation.ProxyCommand]::Create ($metadata)

És végül...

... kérem, tegyék fel kérdéseiket!

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Függvénytár és scriptmodul készítés

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

PowerShell haladó - tartalomjegyzék

Függvénytár

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

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
```

Jegyzékmodul

- Lehetőség van olyan modul létrehozására is, amiben nincsen modulfájl, csak jegyzékfájl és esetleg .NET építőelemek.
  - # Modules to import as nested modules of the module specified in ModuleToProcess
  - NestedModules = 'module1','module2'
- Ezzel a lehetőséggel egy „igazi" modulhoz akár többfajta jegyzékfájllal más és más exportált változókat, függvényeket és cmdleteket lehet definiálni.

És végül...

... kérem, tegyék fel kérdéseiket!

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Hibakezelés és debugging

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

PowerShell haladó - tartalomjegyzék

Hibatípusok a PowerShellben

- Terminating error
  - Megszakító hibák
    - Nullával való osztás
    - Szintaktikai hibák
- Non-terminating error
  - Nem megszakító hibák
    - Cmdlet-ek paraméterezési hibái
- $error tömb!

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, Inquire - rákérdez.
ErrorVariable	Saját hibaváltozónk neve ($ jel nélkül!). A $error változó mellett ide is betöltődik a hibát leíró objektum.
OutVariable	Kimenetet ide tölti be.
OutBuffer	Az objektum-puffer mérete, ennyi elemet „magában” tart, mielőtt továbbítja az outputot a következő csőszakasznak

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, Inquire - rákérdez.
$MaximumErrorCount	Az $error tömb maximális mérete. Az ennél régebbi (nagyobb sorszámú) hibajelzések kihullanak a tömbből.
$ErrorView	A hibajelzések nézete: Normal vagy CategoryView

Hibakezelés függvényben vagy scriptben

- Csapda (Trap)
  - Példa:
    - trap [ExceptionType]
      - # pl.: System.DivideByZeroException
    - { művelet(ek) }
- Dobni és elkapni (Throw..Trap)
  - trap { "Hibajelenség: $_ " }
  - function dupla ($v = $(throw (New-Object System.ArgumentException -arg "Adjál meg valamit, amit duplázni lehet!")))
  - {
  - $v*2
  - }

Hibakezelés függvényben vagy scriptben

- Try..Catch..Finally
- Példa:

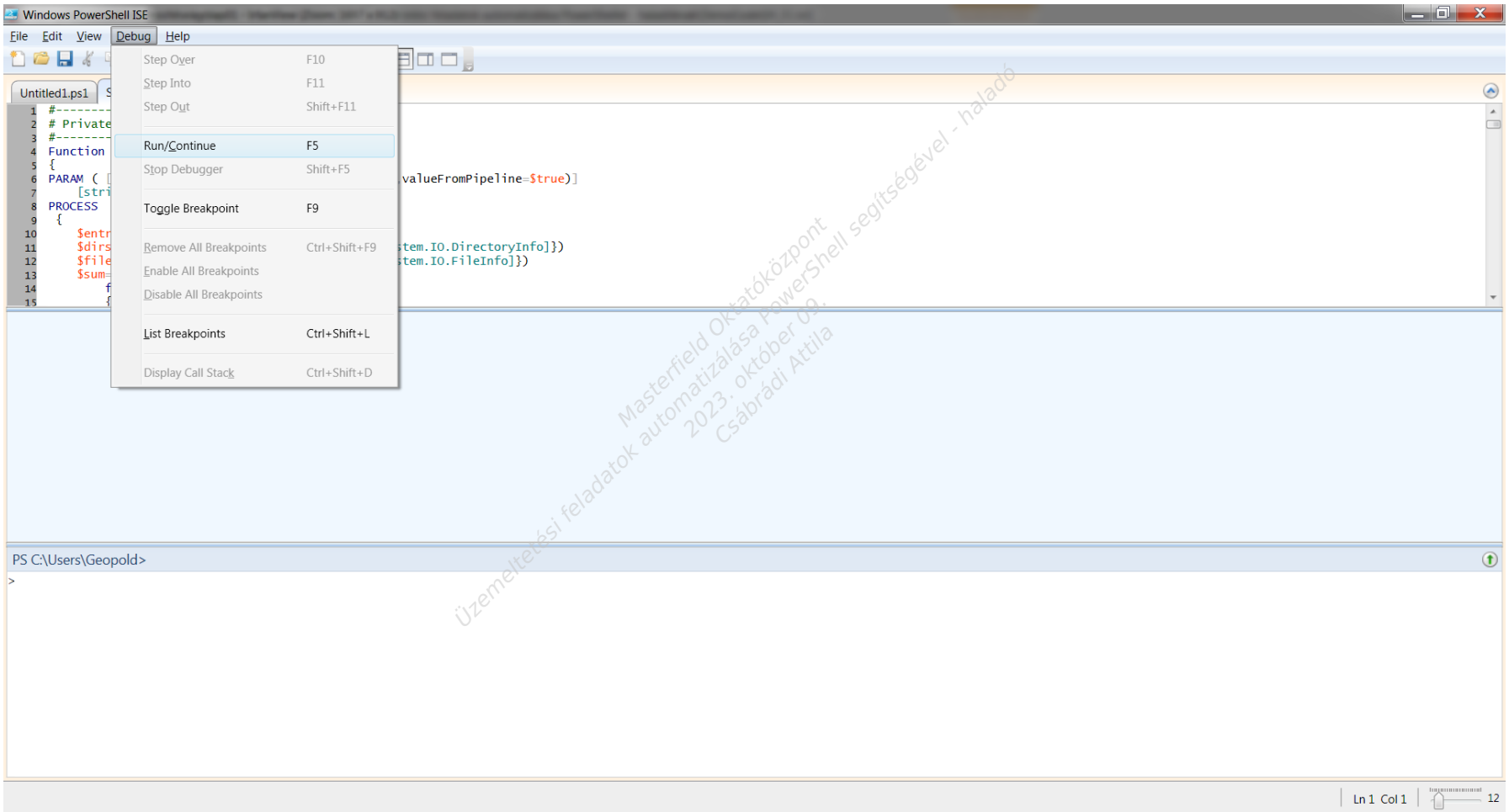
```
Try
{
    $AuthorizedUsers = Get-Content \\FileServer\HRShare\UserList.txt -ErrorAction
    Stop
}
Catch [System.OutOfMemoryException]
{
    Restart-Computer localhost
}
Catch
{
    $ErrorMessage = $_.Exception.Message
    $FailedItem = $_.Exception.ItemName
    Break
}
Finally
{
    $Time=Get-Date
    "This script made a read attempt at $Time" | out-file c:\logs\ExpensesScript.log -
    append
}
```

Debugging

- -debug paraméter használata
- -erroraction változó beállítása
- Set-PSDebug -Trace használata
- Set-PSBreakpoint használata

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

Debugging



És végül...

... kérem, tegyék fel kérdéseiket!

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Rendszerszolgáltatások

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

PowerShell haladó - tartalomjegyzék

Munkafolyamatok

A PowerShell 3.0 verziótól új lehetőség jelent meg szkriptek futtatására: a workflow (munkafolyamat). A workflow nagyon hasonlít a függvényhez, azonban a munkafolyamatok kialakításuk szerint lehetnek hosszú futású, ismételhető, gyakori, párhuzamosítható, megszakítható, megállítható és újraindítható folyamatok. Szüneteltethetőek és folytathatóak; váratlan megszakadás (pl.: hálózati kimaradás vagy számítógép-újraindítás) után folytathatóak.

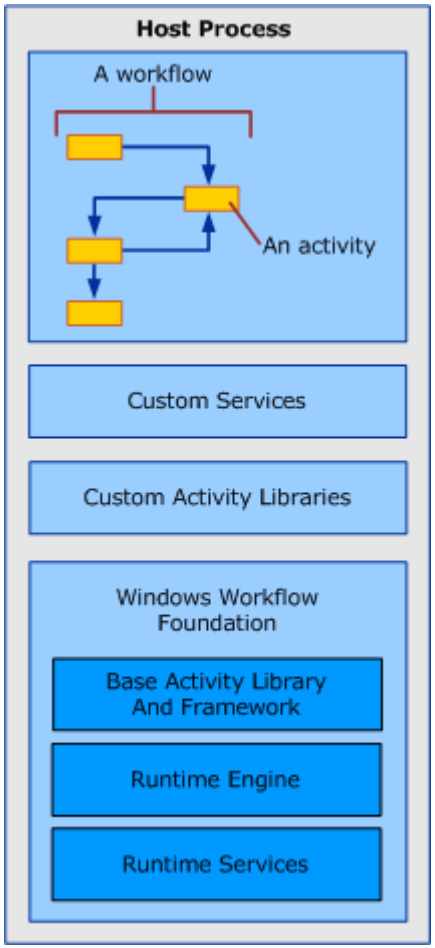
Munkafolyamatok

- 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.

Munkafolyamatok

A workflow-kban nem parancsokat, hanem tevékenységeket (*activity*) hajtunk végre. A tevékenységek eredményét a workflow elmentheti, így a folyamat megszakadása esetén, ha azt újraindítjuk, akkor folytatható a folyamat a megszakadási ponttól. Mi határozhatjuk meg, hogy mikor iktatunk be olyan ellenőrzési pontokat (*checkpoint*), amelyekhez újraindítás esetén szeretnénk visszatérni. Kérhetjük azt is, hogy minden tevékenység után a WWF automatikusan készítsen ellenőrzési pontot, illetve bizonyos tevékenységek esetében a workflow maga illeszt be ilyen ellenőrzési pontot, mint például a gép újraindításakor.

A tevékenységeket végezhetjük sorban és párhuzamosan is. A soros végrehajtás teljesen hasonló a függvényeknél látott végrehajtási módhoz. A párhuzamos végrehajtás natívan nem érhető el a PowerShell függvényekben, csak ha héttérbeli folyamatokat (*job*) készítünk, vagy *Runspace* objektumokkal mi magunk valósítjuk meg a többszálú működést.



Munkafolyamatok

- Párhuzamos végrehajtás munkafolyamatban

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

Desired State Configuration

- 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

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

Desired State Configuration

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

Desired State Configuration

- Konfiguráció

```
Configuration DSCConfig {
    Node localhost {
        File Fájl {
            Ensure = 'Present'
            Type = 'File'
            SourcePath = 'C:\Source\file.txt'
            DestinationPath = 'C:\Destination\file.txt'
            MatchSource = $true
        }
        Registry Reg {
            Key = "HKLM:\SOFTWARE\DSCTest"
            ValueName = "Value"
            Ensure = 'Present'
            ValueData = "DSCValue"
            ValueType = "String"
        }
    }
}
```

És végül...

... kérem, tegyék fel kérdéseiket!

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

Győri György

Adatbázisok elérése PowerShellből

Masterfield Oktatóközpont
http://www.masterfield.hu

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

PowerShell haladó - tartalomjegyzék

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()

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)

SQL adatok elérése

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

OleDB

- [$connectionString = "Provider=microsoft.jet.oledb.4.0; "
- + "Data Source=C:\scripts\emberek.mdb; "
- $sqlCommand = "select * from személyek"
- $connection = New-Object System.Data.OleDb.OleDbConnection $connectionString
- $command = New-Object System.Data.OleDb.OleDbCommand $sqlCommand,$connection
- $connection.Open()
- $adapter = New-Object System.Data.OleDb.OleDbDataAdapter $command
- $dataset = New-Object System.Data.DataSet
- [void] $adapter.Fill($dataSet)
- $connection.Close()
- $records = $dataSet.Tables | Select-Object -Expand Rows | Select-Object -Expand Rows

SQL Server PowerShell Extensions

- Közösségi fejlesztés
  - Mint a PSCX
  - SMO-ra épül
  - Providerek és cmdlet-ek
  - Scriptek
  - Folyamatos fejlesztés

SQLPS

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

És végül...

... kérem, tegyék fel kérdéseiket!

Üzemeltetési feladatok automatizálása PowerShell segítségével - haladó

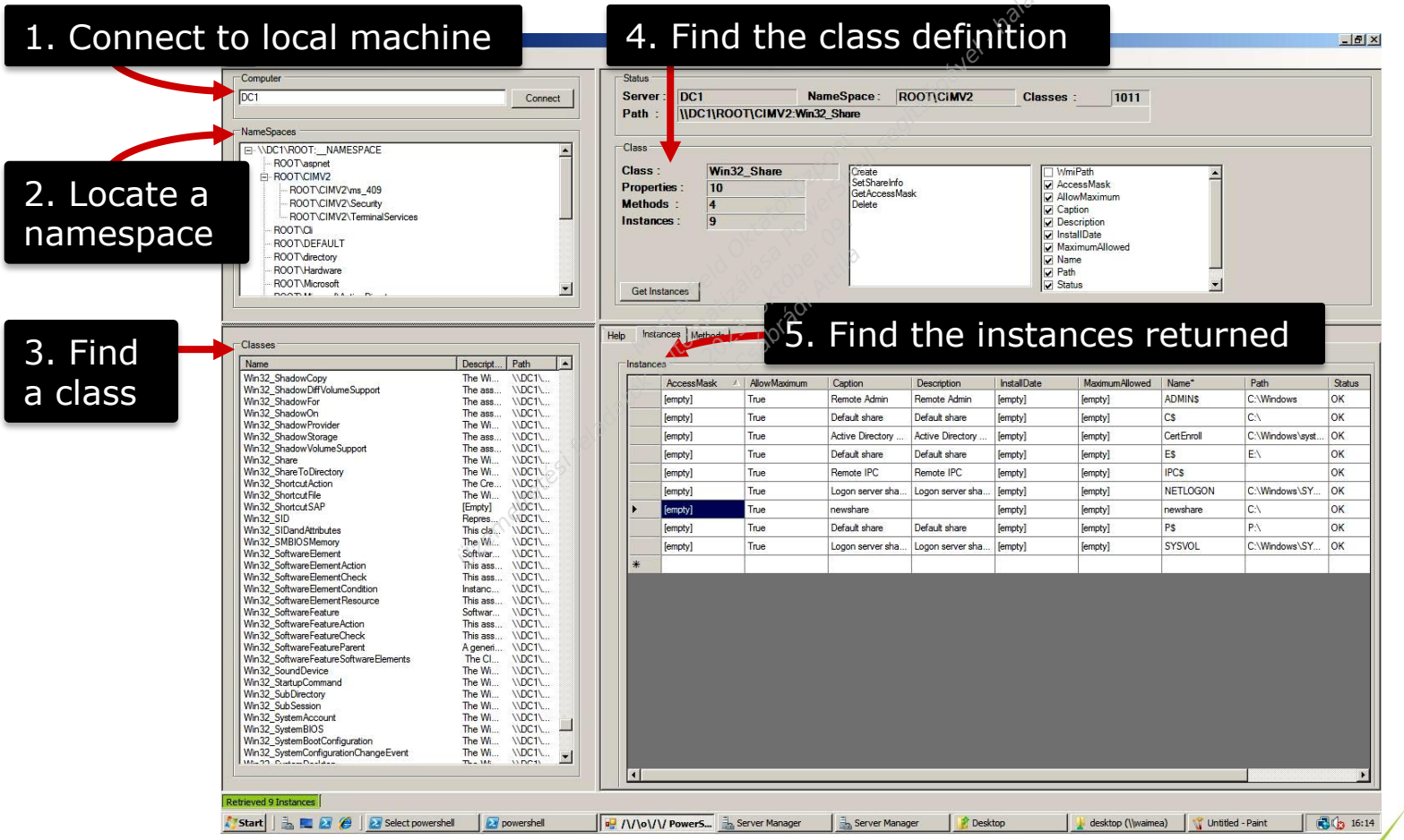
Győri György

Grafikus felület tervezése

Masterfield Oktatóközpont
http://www.masterfield.hu

PowerShell haladó - tartalomjegyzék

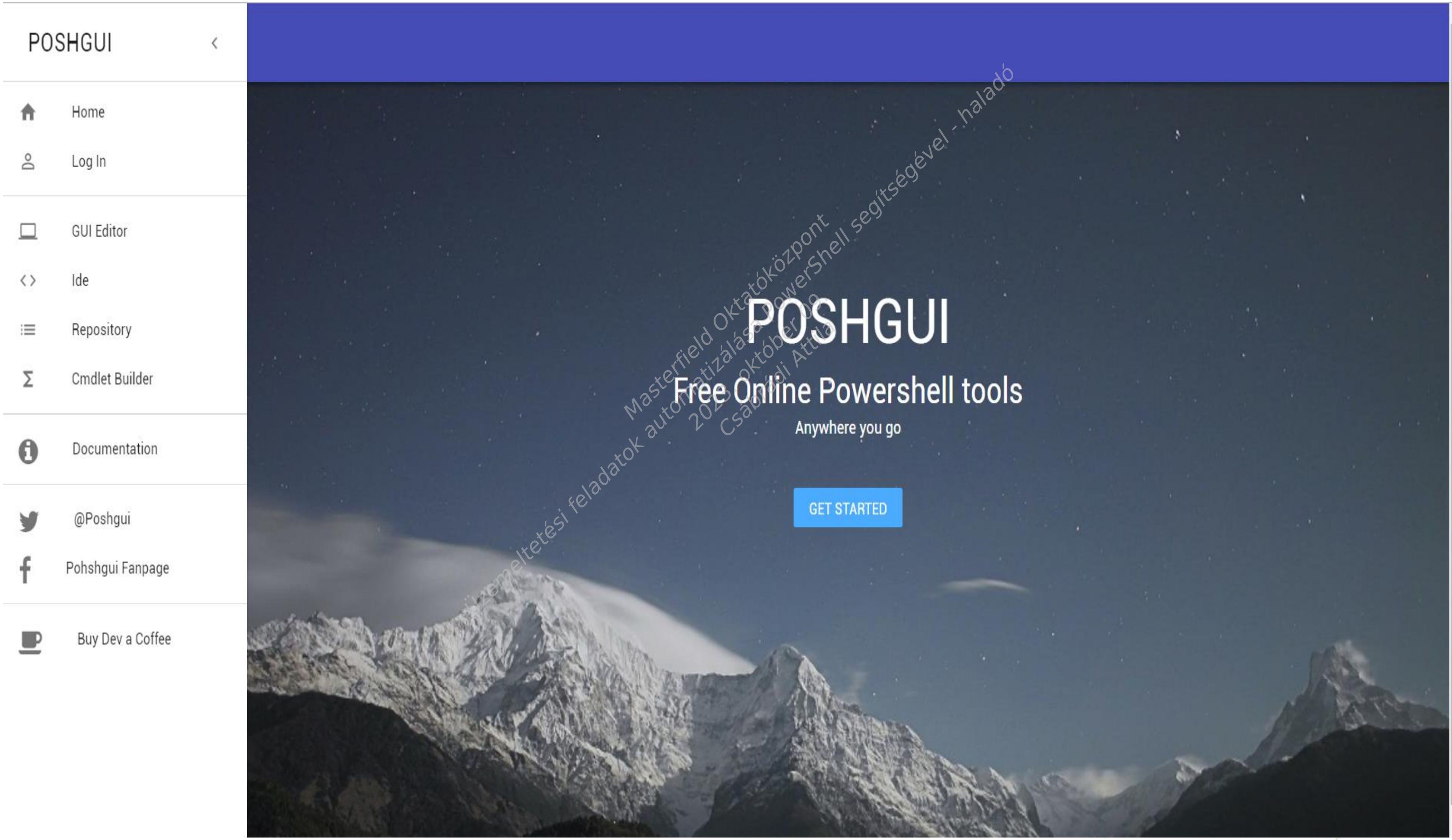
WMI Explorer



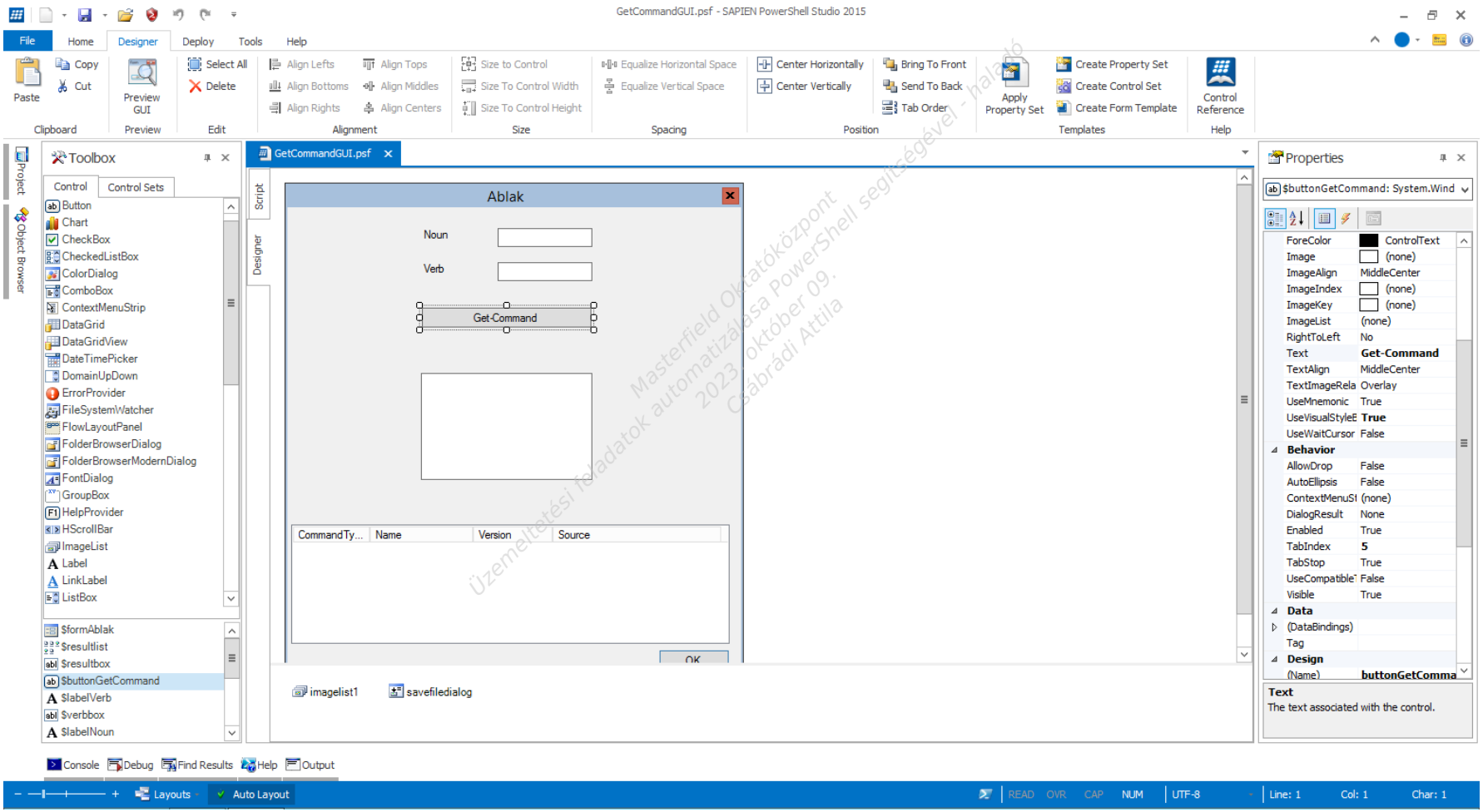
GUI elemek

- [void][System.Reflection.Assembly]:: LoadWithPartialName("System.Windows. Forms")
  - Add-Type -Assembly System.Windows.Forms
- [void] [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"

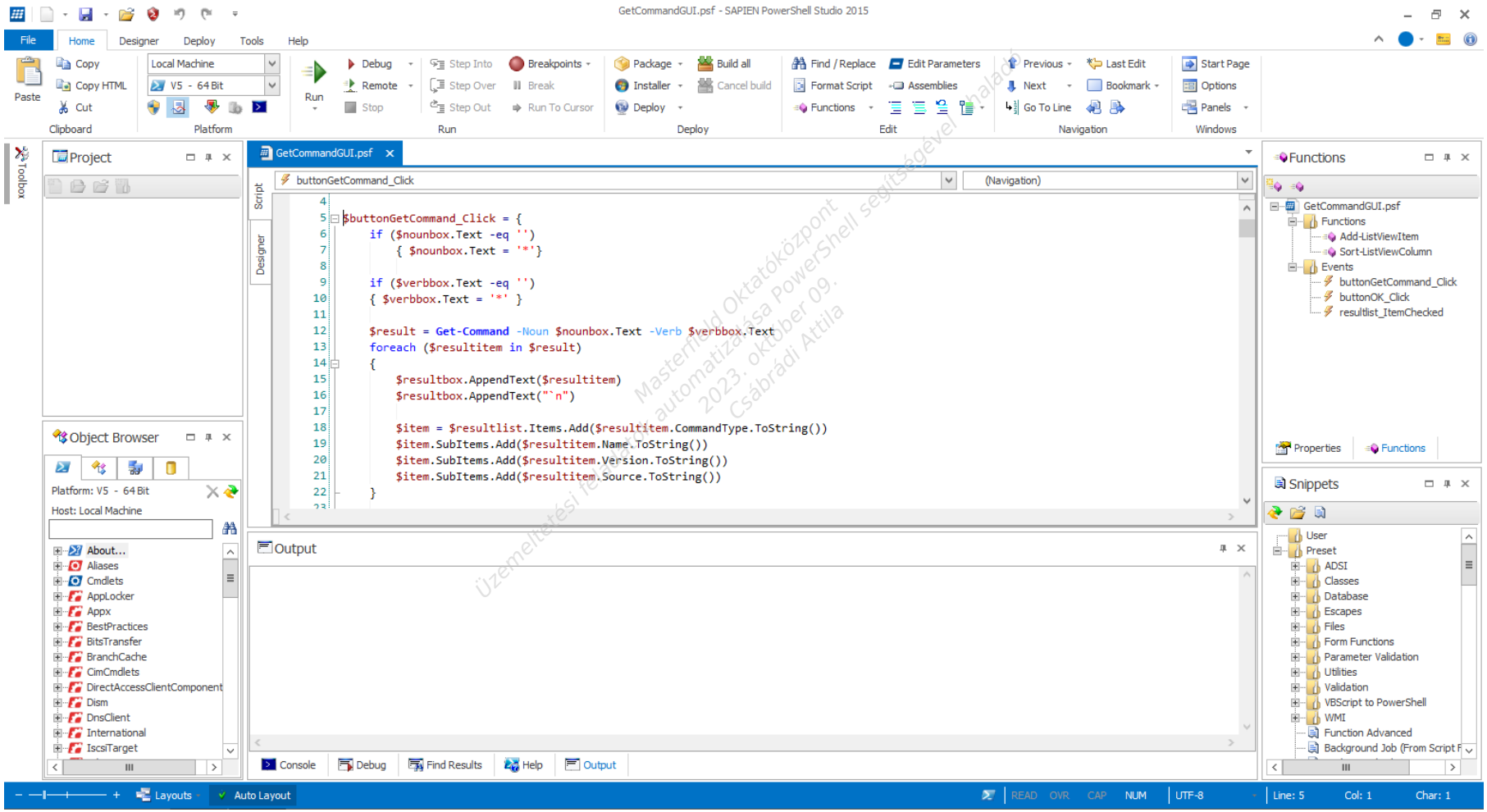
PoshGui.com


POSHGUI
Home
Log In
GUI Editor
Ide
Repository
Cmdlet Builder
Documentation
@Poshgui
Pohshgui Fanpage
Buy Dev a Coffee
POSHGUI
Free Online Powershell tools
Anywhere you go
GET STARTED

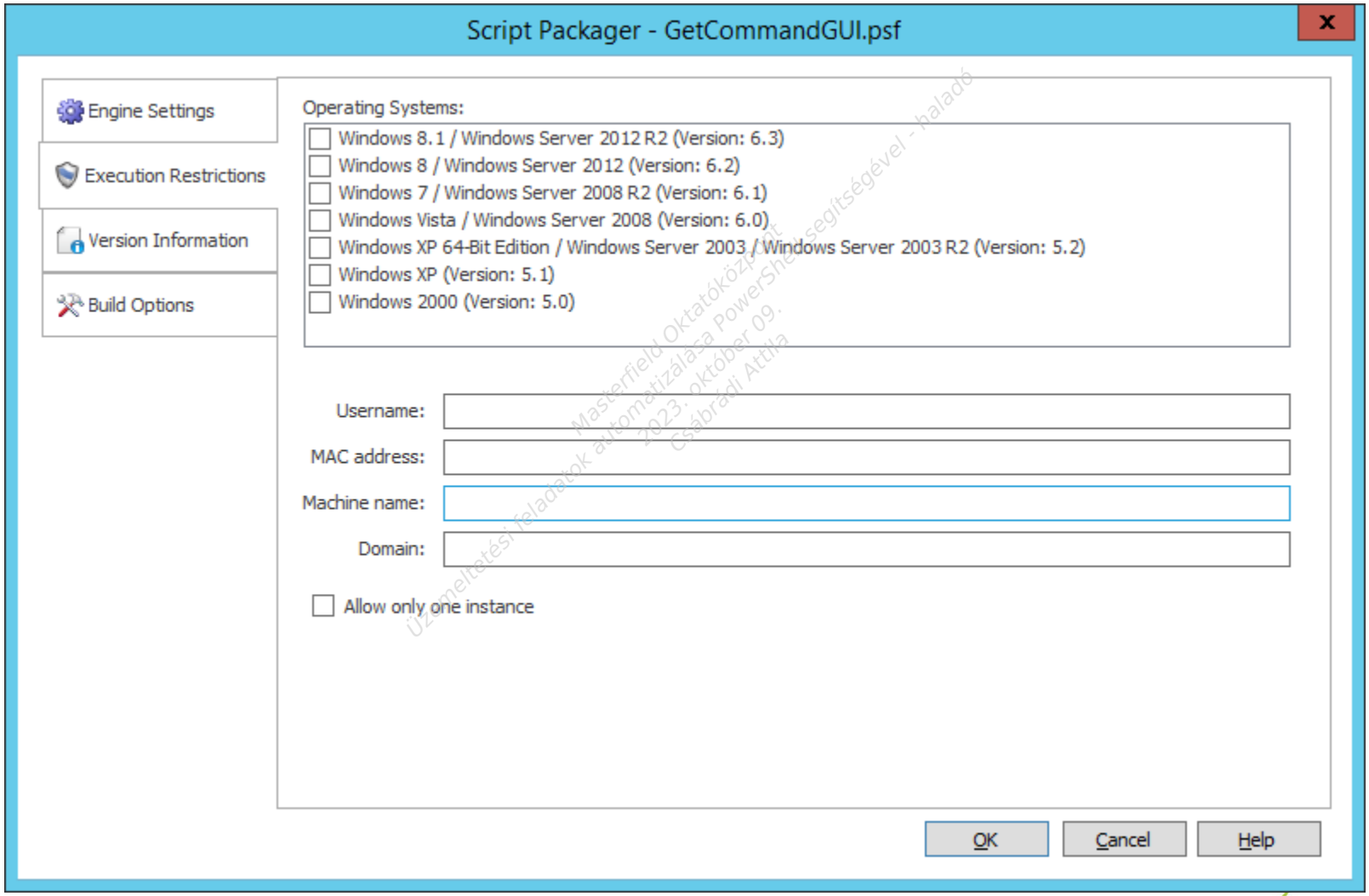
Powershell Studio



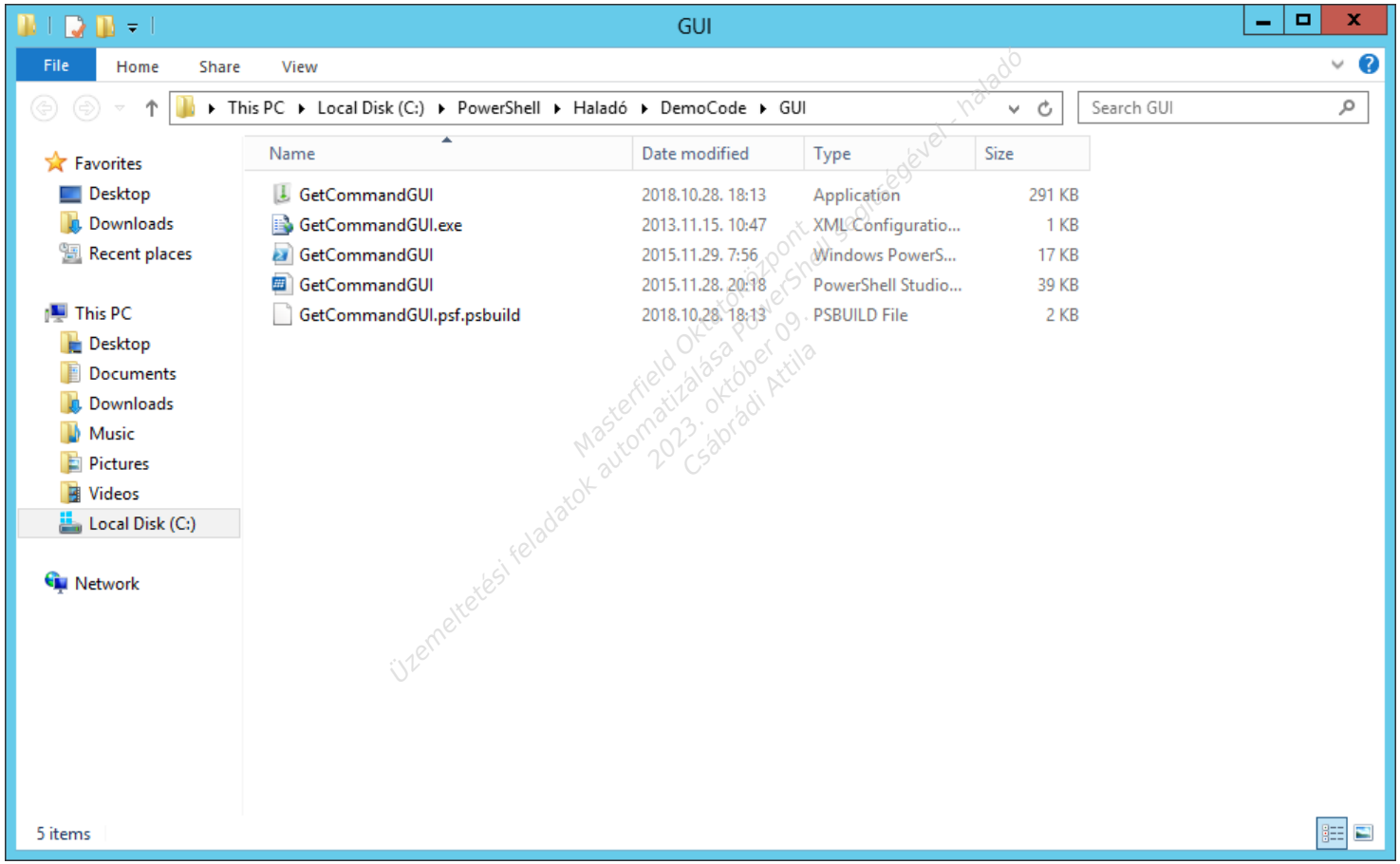
Powershell Studio



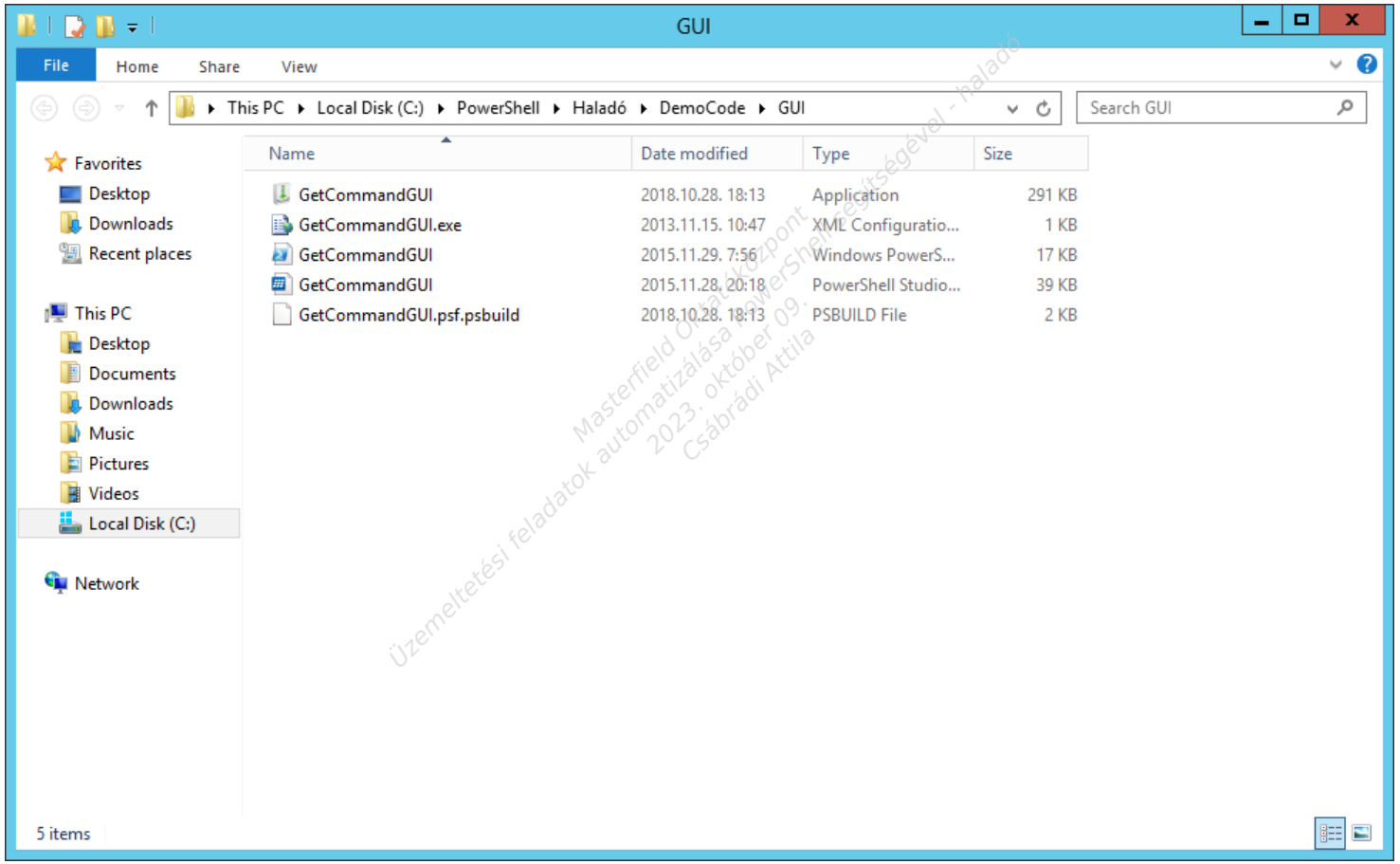
Powershell Studio



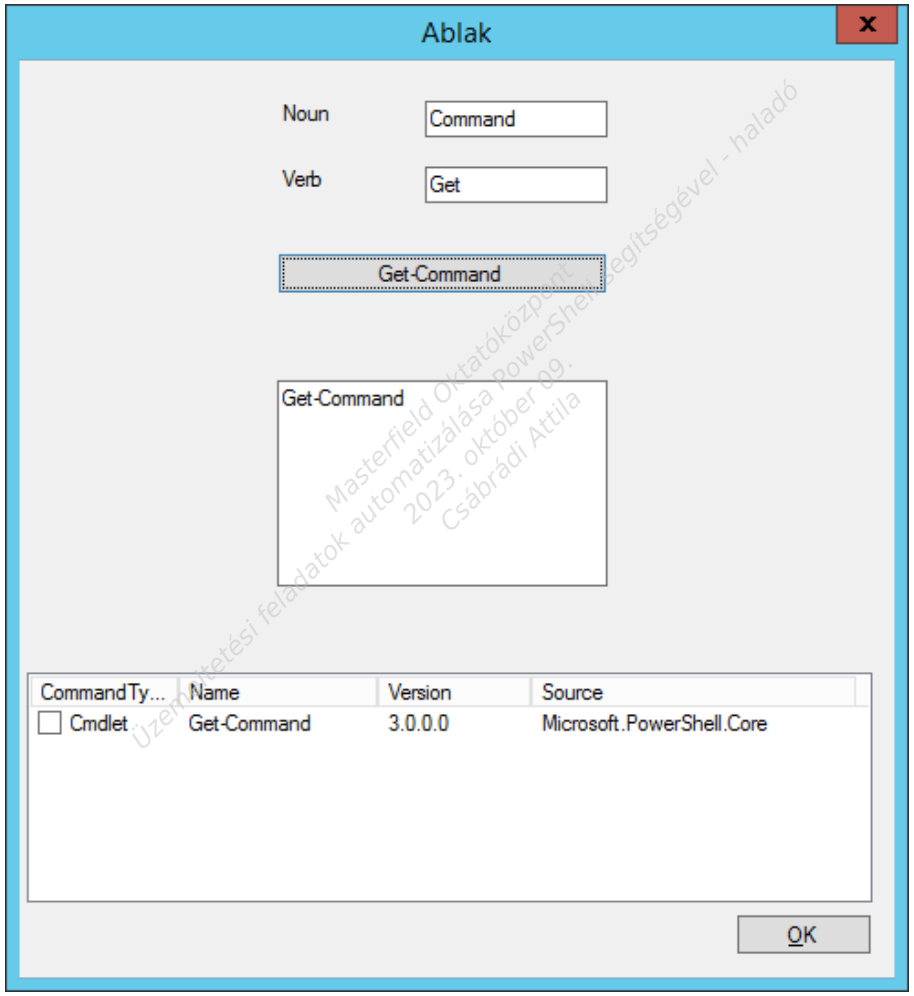
Powershell Studio



Powershell Studio



Powershell Studio



És végül...

... kérem, tegyék fel kérdéseiket!

Köszönjük, hogy meghallgatták előadásunkat!

További tanfolyamok és információ:

www.masterfield.hu