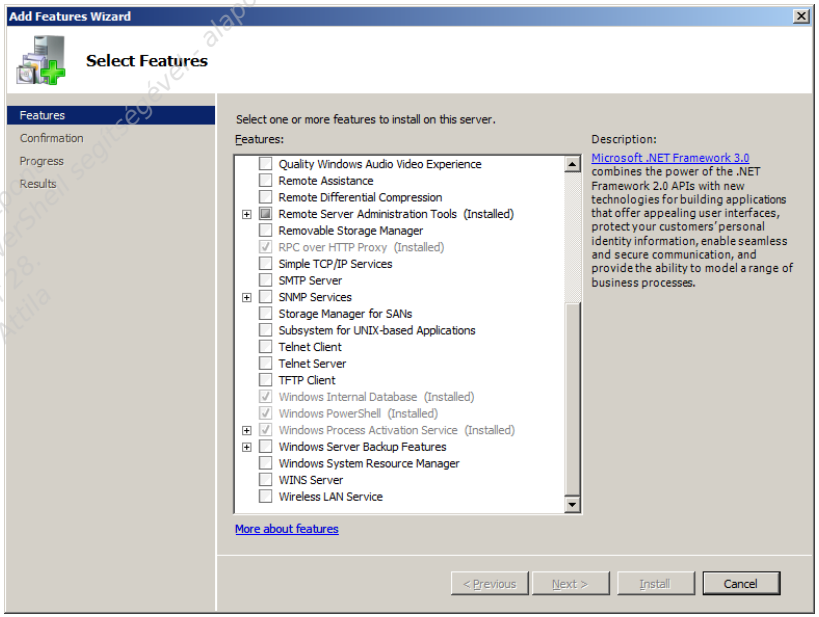
Üzemeltetési feladatok automatizálása PowerShell segítségével - alapok

Masterfield TRAINING
IT COURSES FOR PROFESSIONALS

Agenda

Telepítés, feltételek

- Kezdetek
  - 2006:
    - .NET FrameWork 2.0
    - Windows XP/2003/Vista/2008
  - Windows 2008:
    - Feature-ként volt telepíthető



Windows PowerShell Verziók

	2.0	3.0	4.0	5.0
Windows XP	Nem elérhető	Nem elérhető	Nem elérhető	Nem elérhető
Windows Server 2003	Elérhető	Nem elérhető	Nem elérhető	Nem elérhető
Windows Vista	Elérhető	Nem elérhető	Nem elérhető	Nem elérhető
Windows Server 2008	Elérhető	Elérhető	Nem elérhető	Nem elérhető
Windows 7	Telepítve	Elérhető	Elérhető	Elérhető
Windows Server 2008 R2	Telepítve	Elérhető	Elérhető	Elérhető
Windows 8	Nem elérhető	Telepítve	Elérhető	Elérhető
Windows Server 2012	Nem elérhető	Telepítve	Elérhető	Elérhető
Windows 8.1 és Windows Server 2012 R2	Nem elérhető	Nem elérhető	Telepítve	Elérhető
Windows Server 2016/2019 és Windows 10	Nem elérhető	Nem elérhető	Nem elérhető	Telepítve

Windows PowerShell 1.0 and 2.0 = .NET Framework 2.0
Windows PowerShell 3.0 = .NET Framework 4.0
Windows PowerShell 4.0/5.0 = .NET Framework 4.5

PowerShell 6/7-es Verziók

A PowerShell Core 6.0 volt az első kiadás mely az új többplatformos .Net Core keretrendszerre épült. A Windows PowerShell változataival való kompatibilitása hiányos.

A PowerShell 7.0 a PowerShell nyílt forráskódú, platformfüggetlen (Windows, macOS és Linux) kiadása, amely a heterogén környezetek és a hibrid felhő kezelésére készült.

A PowerShell 7.0 a .NET Core 3.1-es verzióra váltott, ami jelentősen nagyobb kompatibilitást tesz lehetővé a meglévő Windows PowerShell-modulokkal.

Verziómeghatározás

- $PSVersionTable használata (PowerShell 1.0 esetén nem létezik ez a rendszerváltozó)
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\PowerShell\3\PowerShellEngine\PowerShellVersion
  - Itt tárolódnak a beépített (rendszerrel kapott) PowerShell verzió beállításai
- (Get-Host).Version
  - A Host a Powershellben az a program amin belül a PowerShell motor fut, vagyis a konzol, az ISE (az integrált szkriptelő környezet), valamilyen IDE (integrált fejlesztőkörnyezet) pl a Visual Studio, vagy a Visual Studio Code, amiben van beépített terminál. Ennek a hostnak a Version tulajdonsága adja meg, hogy a Powershell motor milyen verziószámú.
- Futtassuk a PowerShell-t a –version 2.0 kapcsolóval ha kompatibilitási üzemmódot akarunk használni.

Segédprogramok

- ISE (Integrated Scripting Environment)
- Visual Studio Code
- PowerGUI, PowerGUI Script Editor
- PSScriptPad
- PowerShell Plus
- PowerShell Studio
- Kiegészítő segédprogramok:
  - RegexBuddy
  - Reflector, ILSpy

Kiegészítések

- PowerShell Community Extension (PSCX)
- Quest – ActiveRoles Management Shell for Active Directory
- Exchange Management Shell (EMS)
- PowerShell Extension for SQL (SQLPS)
- SharePoint Management Shell
- PowerShellGallery

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

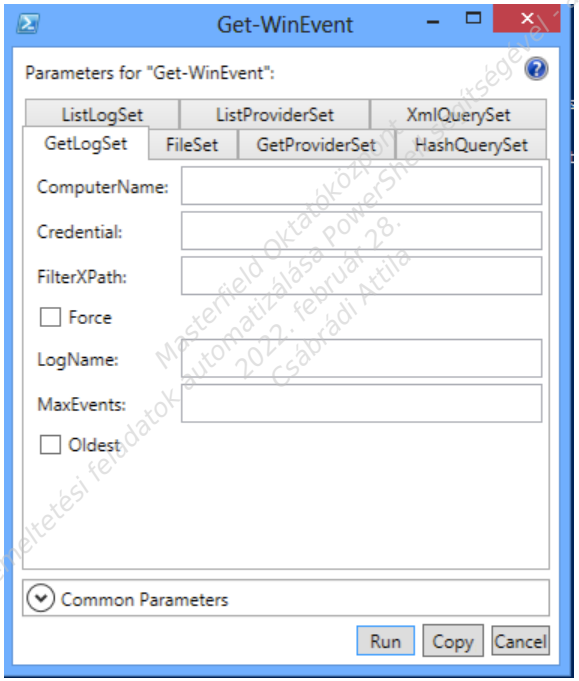
A .NET FrameWork

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

PowerShell alapelemek

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

Show-Command



Súgó használata

- Get-Help
  - Detailed
  - Full
  - Examples
  - Online
- Update-Help
  - LiteralPath
- Save-Help
  - LiteralPath
- Get-Help <parancs> -Parameter <parameter>
- Get-Help about_*

TAB kiegészítés

- PS C:\> Set-<tab>
- PS C:\> Get-ChildItem -<tab>
- PS C:\> New-Alias # bekéri a kötelező paramétereket
- PS C:\> $ezegynagyonhosszunevuvaltozo = 1
- PS C:\> $ez<tab>
- PS C:\> $s = "Helló világ!"
- PS C:\> $s.<tab>

Parancsok egy és több sorban

- PS C:\> "Ez egy
>> enterrel megtört mondat"
>>
- PS C:\> $s `
>> = "Hello world!"
>>
- PS C:\> $s ; 1+2; $ez<tab>

Parancstörténet

- Get-History
- Invoke-History #
- Clear-History
- (Get-PSReadlineOption).HistorySavePath
  - C:\Users\<User>\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
- Get-History | Export-Clixml -Path C:\commands.xml
  - Add-History -Path C:\commands.xml

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

Gyakorlat

- . .\start-demo.ps1
- Get-ExecutionPolicy
- Set-ExecutionPolicy ”RemoteSigned”
- . .\start-demo.ps1
- Start-Demo 01_01.txt

Alias-ok

- Becenév, álnév
- Alias hozzárendelhető
  - PowerShell függvények
  - PowerShell szkriptek
  - Bármilyen végrehajtható állomány (exe, com, cmd, vbs, stb.)
- Get/Set-Alias
- New-Alias
- Export/Import-Alias

PSDrive-ok

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

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

Get- Member paraméterei

- -MemberType
  - AliasProperty
  - CodeProperty
  - Property
  - NoteProperty
  - ScriptProperty
  - Properties
  - PropertySet
  - Method
  - CodeMethod
  - ScriptMethod
  - Methods
  - ParameterizedProperty
  - MemberSet
  - Event
  - Dynamic
  - All

- -View
  - Extended
  - Adapted
  - Base
  - All

Csővezeték (pipe)

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

Paraméterátadás: ByValue

String objektumok a csővezetékben

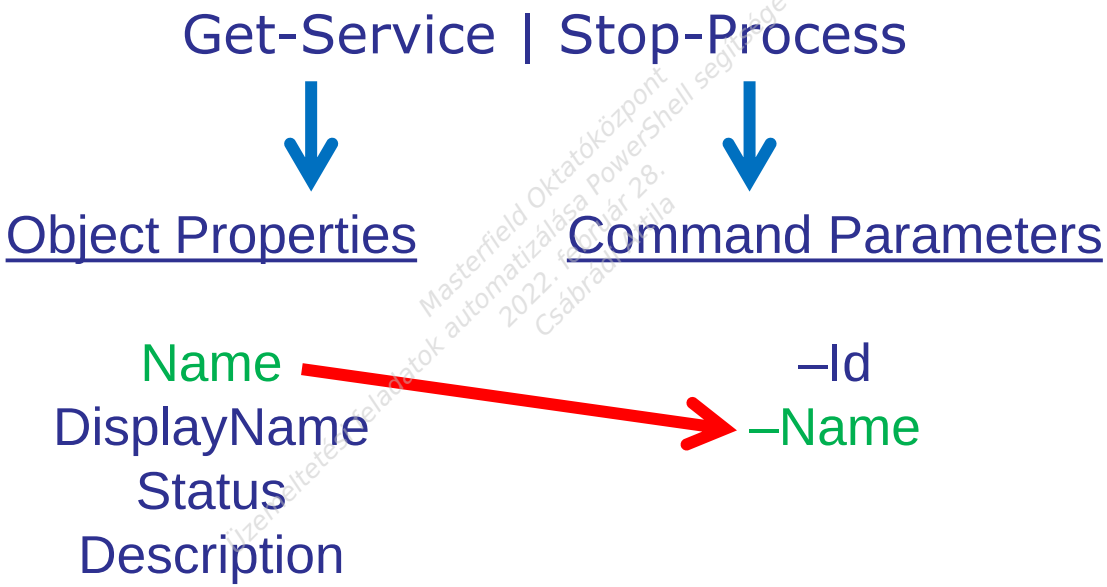
```
"BITS","WinRM" | Get-Service -Name
```

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

Snapinek és modulok

- Powershell 1.0: Snapinek
  - .NET programozói eszközökkel készíthető
  - Tartalmazhat
    - Cmdletek
    - Függvények
    - Változók
    - Aliasok
    - PSProviderek
    - PSDrive definíciók
    - További szolgáltatások (.dll fájlok)

Snapinek és modulok

- Powershell 2.0: Modulok
  - .NET programozói eszközökkel készíthető
  - Tartalmazhat
    - Cmdletek
    - Függvények
    - Változók
    - Aliasok
    - PSProviderek
    - PSDrive definíciók
    - További szolgáltatások (.dll fájlok)
  - PowerShell nyelven készíthető (szkriptmodulok)
  - Tartalmazhat
    - Függvények (mint cmdletek)
    - Változók
    - Aliasok

Snapinek és modulok

- Modulkezelés
  - Get-Module
    - ListAvailable paraméter
  - Import-Module
    - PS 3.0 és Windows 8/2012 : AutoLoading
    - PrefIx paraméter
  - Remove-Module
  - Install-Module
    - PowerShellGet: Package Management
- Modul cmdlet-re hivatkozás
  - Ha nincs ilyen nevű cmdlet, függvény, vagy szkript: Get-NetIpAddress
  - Ha van duplikált parancsnév a rendszerben: NetTCPIP\Get-NetIpAddress
  - Ha definiáltunk prefixet: Get-<prefix>NetIpAddress

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 01_02.txt

Agenda

Változók

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

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 02_01.txt

Idézőjelek használata

- ” ” idézőjel:
  - Speciális karakterek értelmezése
  - Változóértékek behelyettesítése
- ‘ ’ idézőjel:
  - Speciális karakterek figyelmen kívül hagyása
  - Változónevek behelyettesítése
- PS C:\> $string = "szöveg"
- PS C:\> "Ez itt egy $string"
- PS C:\> 'Ez itt egy $string'

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

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 02_02.txt

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	Jelentés	Példa
.	Bármely karakter egy előfordulása	.o.th
[xyz]	A megadott karakterek közül bármelyik	[CMRS]andy
[x-z]	A megadott tartományon belüli karakterek közül bármelyik	[A-Z]eramy
^	A szöveg kezdete	^Subject:
$	A szöveg vége	meeting$
*	Nulla vagy több előfordulása a megelőző helyettesítő karakternek vagy mintának	W.*s
+	Egy vagy több előfordulása a megelőző helyettesítő karakternek vagy mintának	[MZ]+any
?	Nulla vagy egy előfordulása a megelőző helyettesítő karakternek vagy mintának	[MZ]?any
\	A jelölést követő karakter speciálisan dolgozandó fel	Try\$

Kimenet formázása

- Az –f .NET formátumparaméter használata

"1:{0} – {1}" -f 10, 20

– Beállítása

{0,10:C}

Paraméterindex Formátum

Karakterhely
Pozitív szám: jobbra
Negatív szám: balra

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
    - Dir 'C:\', 'fakepath' 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

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 02_03.txt

Agenda

1. Bevezetés
2. Nyelvi elemek
3. Vezérlési szerkezetek
4. Függvények és scriptek
5. A cmdlet-ek
6. Hibakezelés
7. Rendszerszolgáltatások
8. Változások verzióról-verzióra

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

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 03_01.txt

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

- . .\start-demo.ps1
- Start-Demo 03_02.txt
- Start-Demo 03_03.txt

Agenda

Függvény

- Névvel rendelkezik
- Visszatérési értéke van
- A PowerShell-ben:
  - Function – tomb feldolgozás
  - Filter – elemenkénti feldolgozás

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
  - $c = terület(2,3) #nem jó - tömbparaméter!
- Cím szerint paraméterátadás
  - Function dupláz ([ref]$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

Hibakezelés

- Throw kulcsszó
  - Function dupla ($a = 2)

```
{
    if ($a -lt 0)
    {
    throw "Pozitív számot kérek!"
    }
   return $a*2
}
```

  - Function dupla ($a=$(throw "Kötelező paraméter hiányzik!")

```
{
    return $a*2
}
```

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}

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 04_01.txt

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

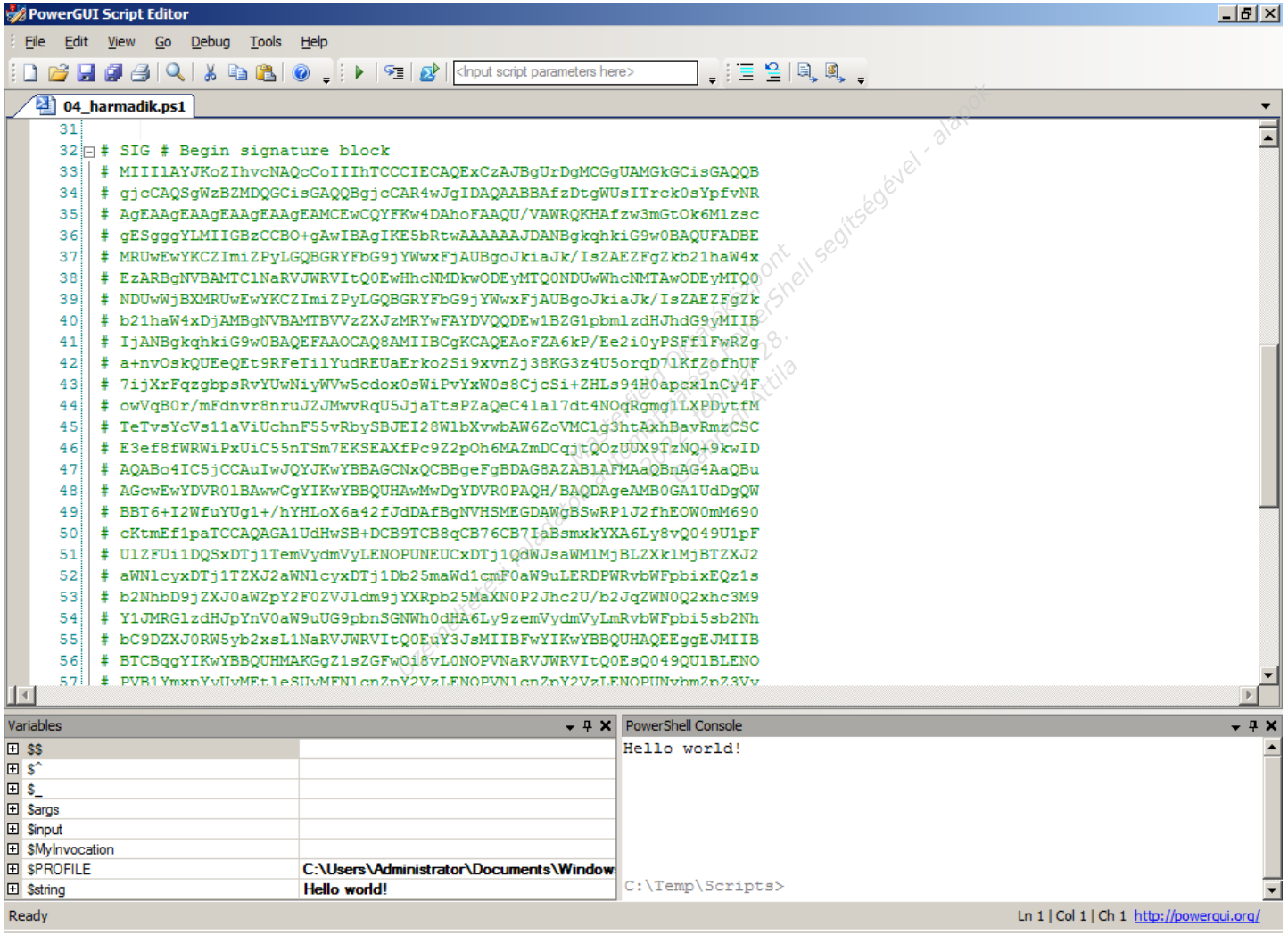
Scriptek futtatása

- 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 (&).

Script paraméterek

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

Scriptek digitális aláírása



Gyakorlat

- . .\start-demo.ps1
- Start-Demo 04_02.txt

Agenda

Fontosabb cmdlet-ek

- Object cmdletek
  - Foreach-Object
  - Where-Object
  - Tee-Object
  - Group-Object
  - Select-Object
  - Sort-Object
  - Compare-Object
  - Get-Unique
  - Measure-Object
- Export cmdletek
  - Out-File,
  - Export/Import-Csv/CliXML
- Item cmdletek
  - Get/Set-Item
  - Get-ChildItem
  - Copy/Move-Item
  - New/Remove-Item
  - Rename-Item

ForEach-Object

- A ForEach-Object cmdlet csővezetékből érkező elemeket dolgoz fel.
- Kétféle szintaxis
  - <parancs> | ForEach-Object -MemberName <tag>
    - Példa
      - Get-ChildItem | ForEach-Object -MemberName Name
      - Get-ChildItem | ForEach-Object -MemberName GetType
  - <parancs> | ForEach-Object {$_.Tulajdonság}
    - Példa
      - Get-ChildItem | ForEach-Object {$_.Name, $_GetType()}
- Alias: %

Where-Object

- A Where-Object cmdlet csővezetékből érkező elemeket szűri egy megadott feltétel szerint.
- Kétféle szintaxis
  - <parancs> | Where-Object Tulajdonság <reláció> érték
    - Példa
      - Get-Process | Where-Object WorkingSet -gt 100Mb
  - <parancs> | Where-Object {$_.Tulajdonság <reláció> érték}
    - Példa
      - Get-Process | Where-Object {$_.WorkingSet -gt 100Mb}

- Alias: ?

Tee-Object

- A Tee-Object cmdlet tetszés szerinti pontra illesztve az objektumhalmazt egy változóba vagy egy megadott fájlba írja, de eközben változatlan formában továbbküldi a csövön is.
- Példa:
  - Get-ChildItem | Tee-Object -FilePath c:\dir.txt | Where-Object {$_.Name -eq "Windows"}

Group-Object

- Group-Object cmdlet segítségével egy (vagy több) megadott tulajdonság értéke szerint csoportosíthatunk objektumokat.
- Példa:
  - Get-Service | Group-Object -Property Status

Select-Object

- A Select-Object cmdlet a kapott objektumok megcsonkítását képes elvégezni, a kimenetként kapott objektumokban már csak a paraméterlistában megadott tulajdonságok fognak szerepelni. Az új objektum PSObject típusú!
- Példa:
  - Get-Process | Select-Object Name, Company, Description
- A másik működési mód a tömbök számosságát csökkenti. A megmaradt objektumok megtartják típusukat.
  - First #
  - Last #
  - Skip #
  - SkipLast #
  - Unique
- Példa
  - Get-Process | Select-Object -First 10

Sort-Object

- A Sort-Object tetszőleges tulajdonság értékei szerint tudja sorba rendezni a kimenetet.
- Példa:
  - Get-Process | Sort-Object -Property WorkingSet –Descending
- Amennyiben nem adjuk meg a rendezés alapját képező tulajdonságot, úgy az objektumon definiált DefaultSortProperty tulajdonságban megadott tag lesz a rendezés alapja.

Compare-Object

- A Compare-Object cmdlet segítségével két tetszőleges gyűjteményt hasonlíthatunk össze, kimenetül a gyűjtemények közötti különbséget leíró objektumokat kapunk.
- Példa:
  - $a = Get-Process
  - &(Read-Host "Folyamat neve: ")
  - $b = Get-Process
  - Compare-Object $a $b

Get-Unique

- A Get-Unique eltávolítja az eredményhalmazból a duplikátumokat.
- Példa:
  - Get-Process | Get-Unique | Sort-Object

Measure-Object

- Tetszőleges objektumcsoport elemeivel kapcsolatos összegzést átlagolást, stb. végezhetünk el a Measure-Object cmdlet segítségével.
- Két üzemmód:
  - Szöveges (példa):
    - Get-Content .\start-demo.ps1 | Measure-Object -IgnoreWhiteSpace -Line -Word -Char
  - Objektumorientált (példa):
    - Get-Process | Measure-Object -Property WorkingSet -Sum

Eredménykonverzió

- ConvertTo/ConvertFrom
  - ConverTo-HTML: HTML táblázattá alakítja az eredményt
    - Get-ChildItem | ConvertTo-HTML | Out-File C:\Temp\dir.txt
  - ConvertTo-Csv: Vesszővel elválasztott értékekké alakítja az eredményt
    - Get-ChildItem | ConvertTo-Csv | Out-File C:\Temp\dir.csv
  - ConvertTo-Xml: XML listává alakítja az eredményt
    - Get-ChildItem | ConvertTo-XML | Out-File C:\Temp\dir.xml
  - ConvertTo-Json: JavaScript objektumtömbökké (kulcs-érték párok) alakítja az eredményt
    - Get-ChildItem | ConvertTo-Json | Out-File C:\Temp\dir.js
  - ConvertFrom-*: átalakított értékhalmaz visszaalakítása objektumtömbbé

Export / Import

- Export
  - Out-File
    - Get-ChildItem | Out-File C:\Temp\dir.txt
  - Export-Csv (-Delimiter)
    - Get-ChildItem | Export-Csv C:\Temp\dir.csv
  - Export-CliXml (-Encoding)
    - Get-ChildItem | Export-CliXML C:\Temp\dir.xml
- Import
  - Get-Content
    - $a = Get-Content C:\Temp\dir.txt
    - $a = Get-Content C:\Temp*.txt
  - Import-Csv (-Delimiter –Header -UseCulture)
    - $a = Import-Csv C:\Temp\dir.csv
  - Import-CliXml (-Encoding)
    - $a = Import-CliXML C:\Temp\dir.xml

Általános elem parancsok

- Item parancsok
  - Get-Item: adott elem lekérdezése
  - Set-Item: adott elem beállítása
  - Get-ChildItem: adott elem gyermek elemeinek lekérdezése
  - Copy-Item: elem másolása
  - Move-Item: elem mozgatása
  - New-Item: új elem létrehozása
  - Remove-Item: elem eltávolítása (törlése)
  - Rename-Item. Elem átnevezése
- Hogy milyen műveleteket és pontosan hogyan végezhetünk el egy elemen, az az elem típusától és a PSProvider által meghatározott lehetőségektől függ.

Kimenet megjelenítésének beállításai

- Kimeneti cmdletek
  - Format-List
  - Format-Table
  - Format-Wide
  - Format-Custom
- Egyéni kimeneti tulajdonság definiálása
  - Format-Table @{Label=címke;
    Expression={scriptblock};
    Width=szélesség}
- Egyéni objektum tulajdonság definiálása
  - Select-Object @{Name=tulajdonságnév;
    Expression={scriptblock}}

Profilok

Elérési út	Magyarázat
%windir%\system32\WindowsPowerShell\v1.0\ profile.ps1	Az adott gép összes felhasználójának, az összes Powershell műveletét befolyásolja.
%windir%\system32\WindowsPowerShell\v1.0\ Microsoft.PowerShell_profile.ps1	Az adott gép összes felhasználójának, az összes a PowerShell konzolban (CLI) végzett műveletét befolyásolja.
%UserProfile%\My Documents\ WindowsPowerShell\profile.ps1	Az adott gép aktuális felhasználójának, az összes Powershell műveletét befolyásolja.
%UserProfile%\My Documents\ WindowsPowerShell\ Microsoft.PowerShell_profile.ps1	Az adott gép aktuális felhasználójának, az összes a PowerShell konzolban (CLI) végzett műveletét befolyásolja.

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 05_01.txt

Agenda

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)
  - Példa:

```
trap { "Hibajelenség: $_" }
function dupla ($v = $(throw
(New-Object System.ArgumentException
-arg "Adjál meg valamit, amit duplázni lehet!")))
  {
        $v*2
  }
```

  - Try … Catch … Finally esetén az Error objektum továbbadására!

Try … Catch … Finally

```
Try
        {művelet(ek)}
        …
Catch [<exception>]
        {művelete(ek)}
        …
Catch
        {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

Agenda

Fájlkezelés

- New-Item
  - Új elem létrehozására
  - Könyvtár és file típus is
  - Példa:
    - New-Item -Path .
      -Name szöveg.txt
      -Type File
      -Value "Ez egy szöveg"
    - New-Item -Path .
      -Name "Alkönyvtár"
      -Type Directory

Szövegfájlok feldolgozása

- Get-Content
  - Szövegállományok tartalmának beolvasása
  - Példa:
    - Get-Content .\start-demo.ps1
- Select-String
  - Az átadott stringből válogatja le a feltételnek megfelelő sorokat
  - Példa:
    - Get-Content $env:windir*.log | Select-String -pattern "error"

Fájlok hozzáférési listáinak kezelése

- Get-Acl
  - FileSecurity objektum
  - Get-Acl .\start-demo.ps1
- Set-Acl
  - Hozzáfűz a listához egy FileSystemAccessRule objektumot
  - Példa:
    - $Acl = Get-Acl C:\PowerShell
    - $entry = New-Object System.Security.AccessControl.FileSystemAccessRule ("Szkriptelők","Read","Allow")
    - $Acl.AddAccessRule($entry)
    - Set-Acl C:\PowerShell $Acl

Alternatív adatfolyamok kezelése

- Get-Item -Path C:\Temp\szoveg.txt -Stream *
- -Stream Paraméter
  - Get-Item
  - Get-Content
  - Remove Item
  - Hozzáférést biztosít a fájlok alternatív adatfolyamihoz

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 06_01.txt

Eseménynapló feldolgozás

- Get-EventLog
  - Példa:
    - Get-EventLog System -Newest 10
      - A rendszernapló utolsó 10 bejegyzését jeleníti meg
    - Get-EventLog -List
      - Az eseménynaplók listáját adja vissza
- EventLogEntry objektum
  - EntryType: pl. Error vagy Warning
  - EventID: eseményazonosító
  - TimeGenerated: esemény bekövetkezésének időpontja
  - Message: esemény üzenetszövege

Registry kezelés

- Registry PSProvider
- Nem könyvtár-fájl analógia
- Kulcsok és tulajdonságok
  - New-Item # kulcs létrehozása
  - NewItemProperty # bejegyzés létrehozása

Property típus	Leírás
Binary	bináris adat
DWord	UInt32 egész
ExpandString	Környezeti változókat kifejtő szöveg
MultiString	Többsoros szöveg
String	Szöveg
QWord	8 bájtos bináris adat

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).

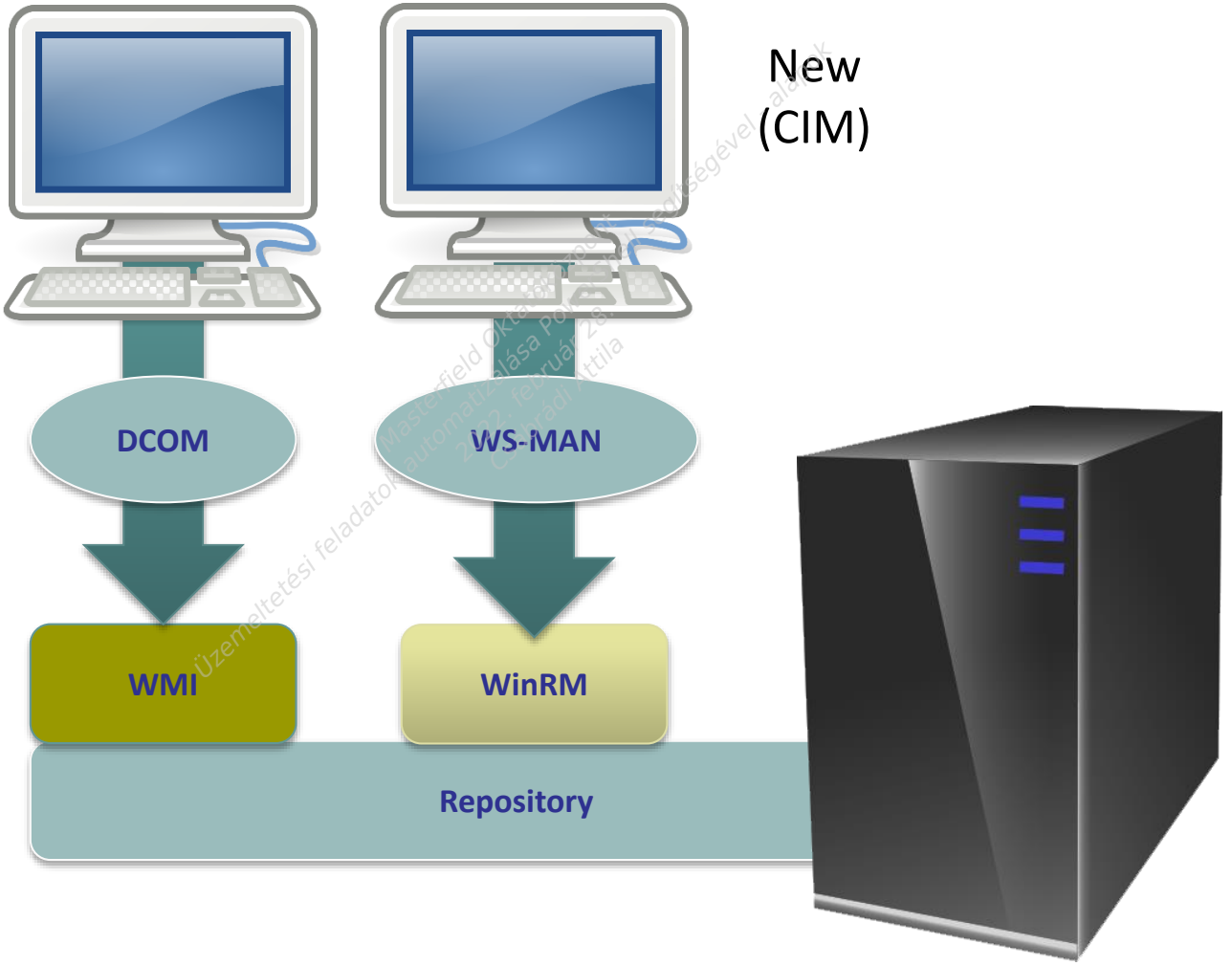
- Win32 Classes: A számítógép hardver elemeinek elérését biztosítja, még a processzor hűtőventillátor kezeléséhez (fordulatszám lekérdezés, szabályozás) is tartalmaz osztályokat.

- 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

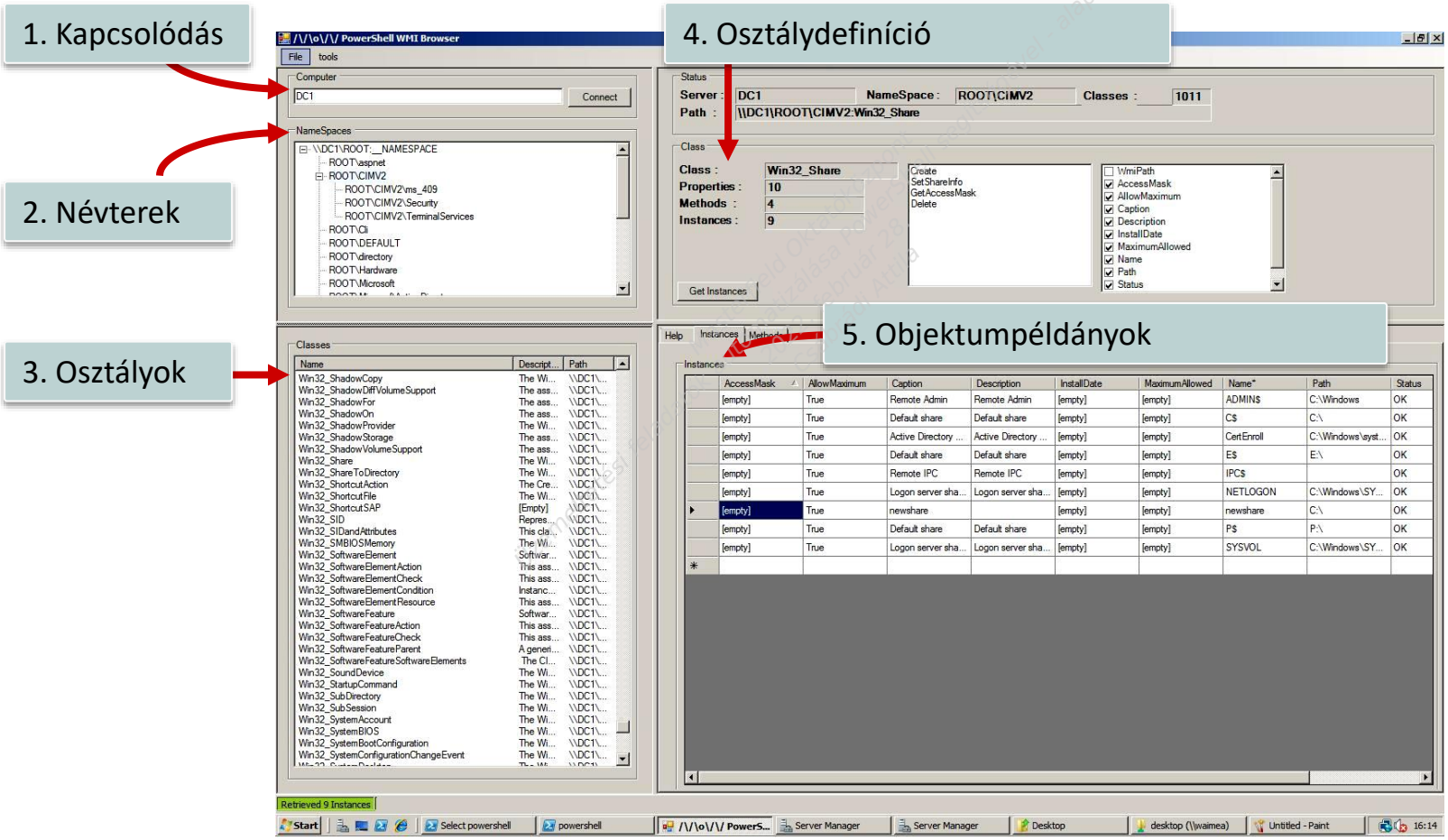
Cím vs. WMI



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

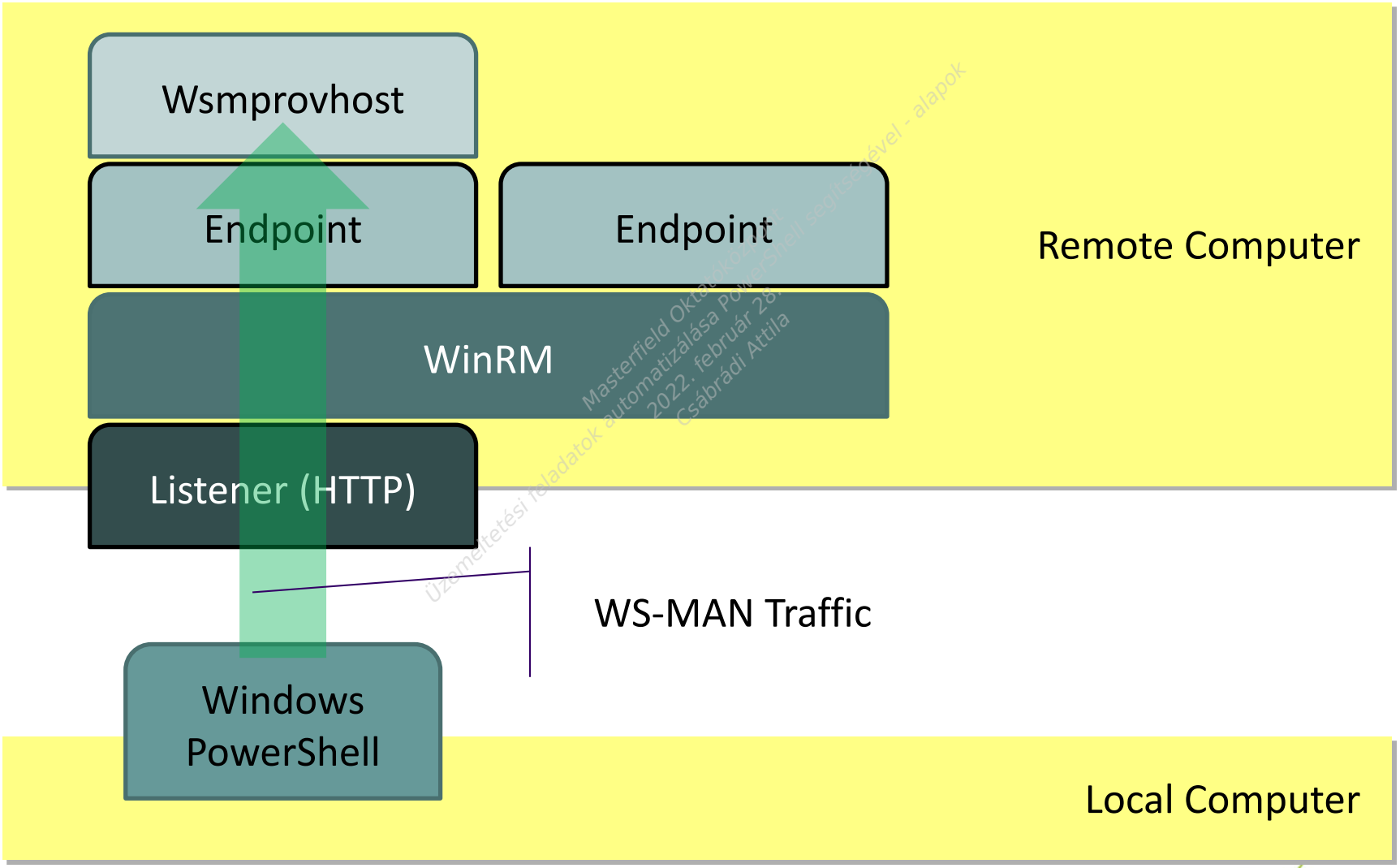
WMI Explorer



Távoli parancsfuttatás

- Protokoll
  - WS-MAN, HTTP (by default) / HTTPS
- WinRM szolgáltatás
- Alapértelmezetten bekapcsolva a Windows Server2012 óta szervereken. Klienseken explicit be kell kapcsolni.
  - Enable-PSRemoting
  - WinRM QuickConfig
- Nem csak a PowerShell használhatja

Remoting Architecture



Helyi és távoli végrehajtás

```
Invoke-Command
  -ScriptBlock {
     Param($c,$r)
     New-PSDrive –Name Z
              -Credential $c
              -PSProvider FileSystem
              -Root $r
     }
  -ComputerName SERVER1,SERVER2,SERVER3
  -ArgumentList     (Get-Credential) ,'Path'
```

Távoli gépen fut

Távoli gépek

Helyi gépen fut

Távoli paraméterátadás

```
Invoke-Command
  -ScriptBlock {
     Param($c,$r)
     New-PSDrive -Name Z
                 -Credential $c
                 -PSProvider FileSystem
                 -Root $r
     }
  -ComputerName SERVER1,SERVER2,SERVER3
  -ArgumentList (Get-Credential),'Path'
```


Ezek az objektumok a paraméterlistában átmásolódnak a távoli gépre és értéket adnak a hívott paramétereknek.

Job-ok

- Háttérben futó utasításaok
- Az eredmény a memóriában tárolódik a lekérésig
- Három job-típus
  - Helyi
  - Távoli
  - WMI

Job-ok indítása

- Helyi job
  - Start-Job –ScriptBlock { Dir }
- Távoli job
  - Invoke-Command –ScriptBlock { Get-Service } -ComputerName Server –AsJob
- WMI job
  - Get-WmiObject –Class Win32_BIOS -ComputerName Server -AsJob

Job-ok kezelése

- Get-Job
  - Egy magadott jobot az ID v. a Name tulajdonság értéke alapján tudunk azonosítani.
- Gyermek jobok azonosítása
  - Get-Job –ID <parent_ID> |
    Select –ExpandProperty ChildJobs
- Stop-Job
- Remove-Job
- Wait-Job

Job eredményének lekérdezése

- Receive-Job
  - Egy megadott job lekéréséhez használhatjuk az ID v. a Name paramétereket
- A –Keep paraméter megtartja az eredményt a memóriában. Enélkül a lekéréskor a job eredménye törlődik.
- Egy szülő job eredményének lekérésekor, az összes gyermek job eredménye is visszaadásra kerül.

Levélküldés PowerShellből

- Kétféle módszer
  - SMTPclient .Net osztály használata
  - $SMTPClient = New-Object Net.Mail.SmtpClient($SmtpServer, 587)
    - Send metódus
  - Send-MailMessage cmdlet használata
    - Send-MailMessage -BodyAsHtml -UseSsl -From $From
      - To $To
      - Cc $Cc
      - Subject $Subject
      - Body $Body
      - SmtpServer $SMTPServer
      - Port $SMTPPort
      - Credential $mycreds
      - Attachments $Attachment

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 06_02.txt

Agenda

PowerShell 3.0

- WorkFlow
  - 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.

PowerShell 3.0

- Where-Object
  - Get-Childitem C:\Windows\System32 | Where {$_.extension -eq ’.dll’}
  - Get-Childitem C:\Windows\System32 | Where extension -eq .dll
- Átirányítás
  - Get-Process svchost,nonexist 2>&1 | Out-File C:\Temp\Proc0.txt

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 07_01.txt

PowerShell 4.0

- Súgó
  - Get-Help
  - Save-Help
  - Update-Help
- Get-Module
- Get-Process

PowerShell 4.0

- 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)

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 07_02.txt

PowerShell 5.0

- Package management
  - A csomagkezelési alrendszer lehetővé teszi, hogy szoftvertárolókból (PS Gallery , NuGet, stb.) telepítsünk fel szoftvercsomagokat.
    - Find-Package
    - Find-PackageProvider
    - Get-Package
    - Get-PackageProvider
    - Get-PackageSource
    - Import-PackageProvider
    - Install-Package
    - Install-PackageProvider
    - Register-PackageSource
    - Save-Package
    - Set-PackageSource
    - Uninstall-Package
    - Unregister-PackageSource

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 07_03.txt

PowerShell 6.0 Core

- A PowereShell 6 Core volt a Microsoft első multiplatformos (Windows, Linux, MacOS) parancsértelmezője.
- A PowerSehll 6 core a .NET Core-ra épült, ami egy csökkentett, ámde teljesen multiplatformos kialakítású keretrendszer változat.
- $IsWindows, $IsMacOs, and $IsLinux belső változók
- Case Sensitive rendszereken az objektumhivatkozások kis-nagybetű érzékenyek (lehetnek).
- Docker támogatás

PowerShell 7.0

- Háromtagú feltétel-operátor
  - 5 -gt 4 ? "Nagyobb" : "Nem nagyobb"
- Parancsok feltételes összefűzése
  - Get-Item C:\Windows\System32\notepad.exe && Write-Host "Van ilyen fájl!"
  - Get-Item C:\Windows\System32\notepad.exe || Write-Host "Nincs ilyen fájl!"
- Null értékek kezelése
  - $a = $null; $a ?? 123
  - $a = $null; $a ??= 123
- Kísérleti funkciók
  - Get-ExperimentalFeature
- Parallel Foreach-Object
  - $tomb|ForEach-Object -Parallel {$_} -AsJob

Gyakorlat

- . .\start-demo.ps1
- Start-Demo 07_04.txt

Köszönöm a figyelmet!