



1장

Windows PowerShell 개요 및 장점

전체 내용

PowerShell이란
무엇인가?

PowerShell을
어떤 상황에
사용하는가?

PowerShell의
장점

PowerShell의
사용상의 특징

PowerShell
Version 및 기본
프로그램 사용

쉬어가는 코너

1-PowerShell이란 무엇인가?

Object-based Shell

CLI 기반의 관리 작업 수행

GUI 프로그램 개발 용도

Scripting 환경 제공

맞춤 가능(Customizable)

확장 가능(Expandable)

.NET Framework 기반

1 - PowerShell이란 무엇인가?

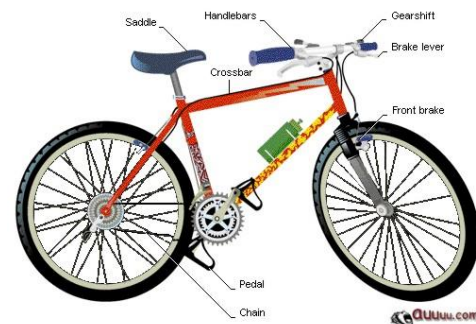
- Object-based Shell

- Object = Property + Method
- 사용 가능한 Property와 Method는 cmdlet마다 다르다
- cmdlet의 Property와 Method를 확인: **cmdlet | Get-Member**

```
PS C:\> Get-Service | Get-Member

TypeName: System.ServiceProcess.ServiceController

Name      MemberType Definition
-----
Name      AliasProperty Name = ServiceName
RequiredServices AliasProperty RequiredServices = $
Disposed  Event System.EventHandler
Close     Method void Close()
Continue  Method void Continue()
```



- Property 내용 보기

- (Get-Service -Name bits).**Status**
- Get-Service | **Where-Object** {\$_.**Status** -eq "Stopped"}
- 각 cmdlet의 Property의 값 확인: **cmdlet | Format-List ***

- Method 실행하기

- (Get-Service -Name bits).**Stop()**
- 100일 후의 날짜 알아보기: (Get-Date).**AddDays(100)**

1 - PowerShell이란 무엇인가?

- Object-based Shell

- Object는 Class에 기반하여 만들어진다
- Class는 "**건물의 설계도**"와 같아서 Object를 생성하거나 조작할 때는 Class를 참고하여 작업한다(Property, Method)
- 건물의 설계도를 보고 집을 건축하고 유지 보수를 하듯이 Object의 Class를 참고하여 Object를 다루게 된다
- **File**이라는 Object에 대하여 살펴본다
 - **Property**
 - Name
 - Size
 - Location
 - Permission
 - **Method**
 - Read
 - Write
 - Copy
 - Move

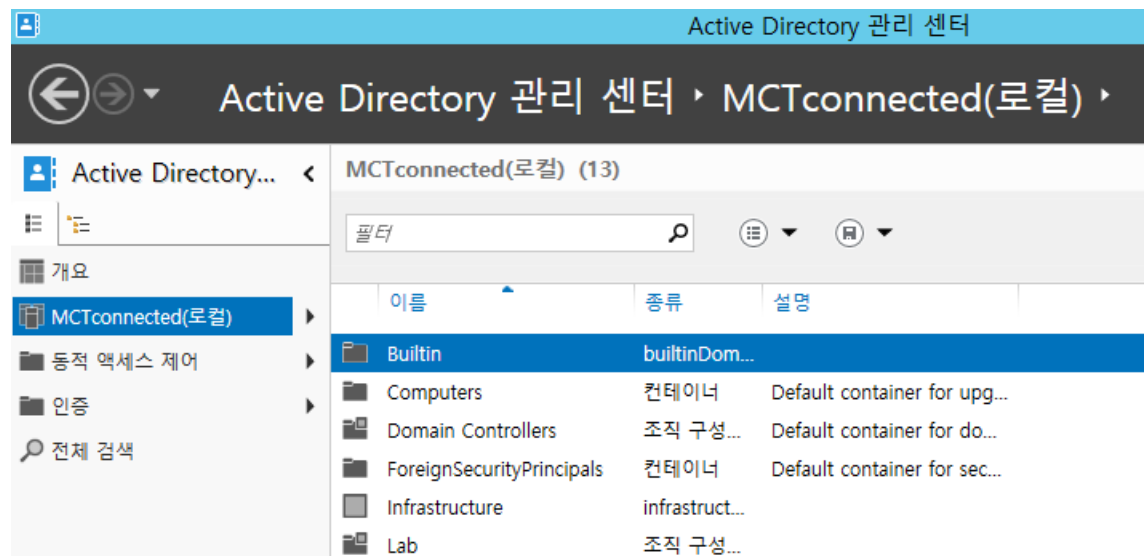
1 - PowerShell이란 무엇인가?

- CLI 기반의 관리 작업 수행
 - GUI는 작업 속도가 느리고, 세부적인 정보를 확인하는데 한계가 있고, 원격 작업할 때도 RPC를 사용하므로 방화벽을 통과하지 못할 수도 있다
 - Windows OS는 PowerShell version2인 Windows 7, Windows 2008R2 이후부터는 CLI를 통하여 GUI의 단점을 보완할 수 있다
 - **Get-Disk**
 - **(Get-Hotfix | Sort-Object -Property InstalledOn)[-1] ##(cmdlet)[-1]**은 최신 결과임
 - 특별히 WinRM 서비스 5985, 5986 포트를 사용하여 원격 관리를 하므로, 방화벽 관리가 쉽다
 - **Enter-Pssession** -ComputerName Server1
 - **Invoke-Command** -ComputerName Server1 {**route print**}
- PowerShell은 기존의 Windows 명령어를 그대로 사용할 수 있다
 - Net users, route, ipconfig, ping
 - Netsh.exe, fsutil.exe, dism.exe, netdom.exe
- Pipeline과 Script를 적절히 사용하면 시스템을 강력하게 관리 가능하다
 - **Get-ChildItem** d:\ -Filter *.png -Recurse | **Copy-Item** -Destination e:\backup

1 - PowerShell이란 무엇인가?

- GUI 프로그램 개발 용도

- 명령어를 사용하여 작업을 해도 되지만 복잡한 내용을 간편한 GUI 프로그램으로도 만들 수 있다
- 예: Active Directory 관리 센터, 서버 관리자



1 - PowerShell이란 무엇인가?

- Scripting 환경 제공

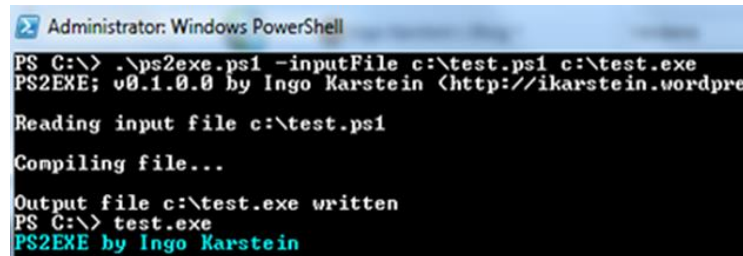
- 반복적으로 해야 할 작업이나 자동화 해야 할 작업에 적합하다
- 유용한 예제 스크립트를 모아 둔 곳: Script Center
 - <https://gallery.technet.microsoft.com/ScriptCenter/>

PS2EXE : "Convert" PowerShell Scripts to EXE Files

This PowerShell script lets you "convert" PowerShell scripts into EXE files.

Download

PS2EXE-v0.5.0.0.zip



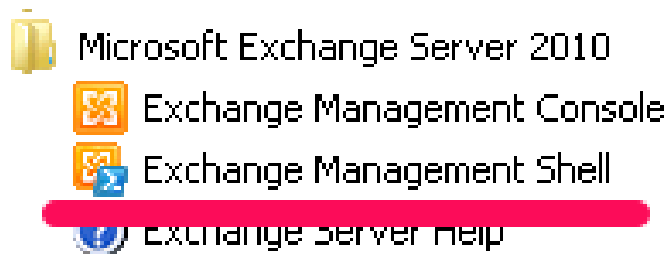
```
Administrator: Windows PowerShell
PS C:\> .\ps2exe.ps1 -inputFile c:\test.ps1 c:\test.exe
PS2EXE; v0.1.0.0 by Ingo Karstein <http://ikarstein.wordpress.com>
Reading input file c:\test.ps1
Compiling file...
Output file c:\test.exe written
PS C:\> test.exe
PS2EXE by Ingo Karstein
```

- Script 예

- 사용자 계정 수백 개 동시 생성(**create_users.ps1**)
 - Import-Csv -Path adusers.csv | New-ADUser
- 특정한 서비스(bits)가 중지되어 있다면 강제로 시작시키기(**start_bits.ps1**)
 - \$bits = (Get-Service -Name bits).Status
 - if (\$bits -eq "Stopped") {Start-Service -Name bits -PassThru}

1 - PowerShell이란 무엇인가?

- 맞춤 가능(Customizable)
 - Exchange Server의 Exchange Management Shell(EMS) 관리도구
 - System Center 2012 R2를 관리하는 PowerShell 관리도구

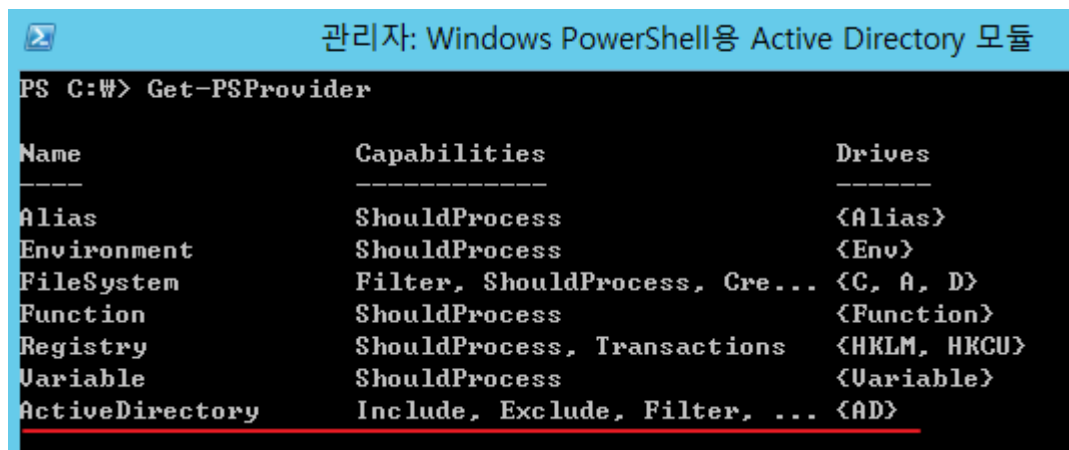


```
[PS] C:\>Get-Mailbox -Identity ADUser*
```

Name	Alias
----	-----
aduser1	aduser1
aduser2	aduser2
aduser3	aduser3

1 - PowerShell이란 무엇인가?

- 확장 가능(Expandable)
 - Script, Function, Module 생성 가능
 - 필요한 Role을 설치하면 추가적으로 다양한 Provider를 제공
 - Get-PSProvider



```
관리자: Windows PowerShell용 Active Directory 모듈
PS C:\> Get-PSProvider
```

Name	Capabilities	Drives
Alias	ShouldProcess	<Alias>
Environment	ShouldProcess	<Env>
FileSystem	Filter, ShouldProcess, Cre...	<C, A, D>
Function	ShouldProcess	<Function>
Registry	ShouldProcess, Transactions	<HKLM, HKCU>
Variable	ShouldProcess	<Variable>
ActiveDirectory	Include, Exclude, Filter, ...	<AD>

1 - PowerShell이란 무엇인가?

- .NET Framework 기반

- PowerShell은 .NET Framework을 이용하여 만든 일종의 Application이다
- PowerShell 명령어인 Get-Process와 .NET Framework의 class에 대한 Method인 [System.Diagnostics.Process]::GetProcesses()의 결과는 동일하다
 - Get-Process | Get-Member를 해 보면 TypeName이 **System.Diagnostics.Process**이다
 - [**System.Diagnostics.Process**] | Get-Member -Static을 입력하면 GetProcesses라는 Method가 있다
 - 즉, .NET 프로그래밍에서 [**System.Diagnostics.Process**]:**GetProcesses()**에 입력한 것의 결과와 PowerShell에서 Get-Process의 결과는 동일하다

```
PS C:\> Get-Process | Get-Member
```

TypeName: System.Diagnostics.Process

```
PS C:\> [System.Diagnostics.Process] | Get-Member -Static
```

TypeName: System.Diagnostics.Process

Name	MemberType	Definition
EnterDebugMode	Method	static void EnterDebugMode()
Equals	Method	static bool Equals(System.Object)
GetCurrentProcess	Method	static System.Diagnostics.Process GetCurrentProcess()
GetProcessById	Method	static System.Diagnostics.Process GetProcessById(int)
<u>GetProcesses</u>	<u>Method</u>	<u>static System.Diagnostics.Process[] GetProcesses()</u>
GetProcessesByName	Method	static System.Diagnostics.Process[] GetProcessesByName(string)

1 - PowerShell이란 무엇인가?

- .NET Framework 기반

- PowerShell의 태생이 .NET Framework이므로 PowerShell 명령어로 작업을 못하면 **그 대안으로** .NET Framework 기능을 사용할 수 있다
 - 사용자 계정이 언제 정확하게 AD에서 인증을 받았는지 확인할 때 .NET Framework 기능을 사용하면 해결된다
 - LastLogon에 대한 값이 우리가 읽을 수 없는 Tick으로 되어 있다. 이것을 해결하기 위해서 .NET Framework 기능을 이용하는 것이다
 - `Get-ADUser -Identity Administrator -Properties * | ft Name, LastLogon -AutoSize`
 - `Get-ADUser -Identity Administrator -Properties * | Select-Object Name,@{N='LastLogon'; E=[DateTime]::FromFileTime($_.LastLogon)}`

```
PS C:\#> get-aduser -Identity administrator -Properties * | ft name, lastlogon -a

name                lastlogon
-----
Administrator 130759771557670896

PS C:\#> Get-ADUser -Identity Administrator -Properties * | Select-Object Name,@{N='LastLogon'; E=[DateTime]::FromFileTime($_.LastLogon)}}

Name                LastLogon
-----
Administrator 5/13/2015 4:52:35 PM
```


2 - PowerShell을 어떤 상황에 사용하는가?

반복적인 작업을 할 때

세부적인 내용을 확인할 때

한꺼번에 여러 개의 데이터를 수정할 때

일상적인 작업을 신속하게 처리할 때

동시에 여러 대의 컴퓨터에 원격 관리 작업할 때

Script 파일을 Group Policy로 적용할 때

2 - PowerShell을 어떤 상황에 사용하는가?

- 반복적인 작업을 할 때
 - 사용자 계정을 한 개만 생성하는 것이 아니라 1000개를 생성할 때
 - `Import-Csv -Path adusers.csv | New-ADUser`
 - File Server의 특정한 폴더를 1주일에 한 번씩 Copy 또는 Backup할 때
- 세부적인 내용을 확인하고자 할 때
 - GUI의 단점인 하나의 화면에 원하는 내용을 맞춤형으로 볼 수 없을 뿐 아니라, 세부적으로 확인할 수 없지만 PowerShell은 가능하다
 - `Get-ADUser -Identity Administrator -Properties * | Format-List *`

2 - PowerShell을 어떤 상황에 사용하는가?

- 한꺼번에 여러 개의 데이터를 수정할 때
 - 부서 이름을 한꺼번에 수정한다
 - `Get-ADUser -Filter {department -eq "Sales"} | Set-ADUser -Department "Sales1"`
- 일상적인 작업을 신속하게 처리할 때
 - IP Address 구성 정보 확인
 - **Get-NetIPConfiguration** (또는 gip)
 - 공유된 폴더 목록 확인
 - **Get-SmbShare**
 - 컴퓨터 이름 변경하기
 - **Rename-Computer** -NewName Server2 -Restart
 - 도메인에 가입하기
 - **Add-Computer** -DomainName powershell.kr -DomainCredential powershell\administrator -Restart
 - 특정한 프로그램 종료하기
 - **Get-Process** -Name chrome | **Stop-Process** -Force

2 - PowerShell을 어떤 상황에 사용하는가?

- 동시에 여러 대의 컴퓨터에 원격 관리 작업할 때
 - WinRM 서비스를 이용한 5985포트로 접속하여 작업하기
 - 10대의 컴퓨터의 최신 보안 패치 현황을 확인하기
 - **Invoke-Command** -ComputerName (Get-Content c:\computers.txt) **{(Get-Hotfix | Sort-Object -Property InstalledOn)[-1]}**
 - 10대의 컴퓨터에서 컴퓨터 부팅시 자동으로 시작해야 할 서비스가 시작되지 않은 것들만 찾아서 자동으로 서비스 시작하기
 - **Invoke-Command** -cn (Get-Content c:\computers.txt) **{Gwmi Win32_Service | Where {\$_.State -ne "Running" -and \$_.Startmode -eq "Auto"} | Start-Service -PassThru}**
- Script 파일을 Group Policy로 적용할 때
 - 도메인의 구성원 컴퓨터 1000대에 동일한 작업을 하고자 하면 .ps1 파일을 생성하여 Startup Script에 추가하면 사용자가 로그인하기 전에 모두 작업이 완료된다

3 - PowerShell의 장점

cmd.exe에서 사용하는 거의 모든 명령어 사용 가능

cmd.exe에서 부족했던 Pipeline 기능 사용 가능

Cmdlet이 계속 늘어난다

명령어의 구문이 이해하기 쉽다

도움말 기능이 탁월하다

사용한 명령어와 관련된 명령어를 쉽게 알 수 있다

3 - PowerShell의 장점

GUI 관리도구에는 없는 기능을 구현한다

자동화 관리 및 반복 작업에 유용하다

Group Policy에서는 .vbs, .bat, .ps1 파일을 사용한다

원하는 cmdlet이 없는 경우에 Script 및 Function을 만들어 사용할 수 있다

원격 관리 기능이 탁월하다

3 - PowerShell의 장점

- cmd.exe에서 사용하는 거의 모든 명령어를 사용 가능
 - 기존에 알고 있던 윈도우 명령어를 버릴 필요가 없을 뿐 아니라, 원격 관리에는 오히려 더욱 강력하게 사용할 수 있다
 - 윈도우 명령어인 .exe 프로그램 및 관리 도구인 .msc , .cpl 등이 어떤 것이 있는지 알 수 있다
 - `Get-Command -CommandType Application | Where-Object {$_.name -like "*.exe"}`
 - 원격 컴퓨터의 mac address를 확인할 수 있다
 - `Invoke-Command -ComputerName Server1, Server2 {getmac}`
- cmd.exe에서 부족했던 Pipeline 기능을 사용 가능
 - 일반적으로 수정하거나 원하는 항목만 보기 위해서는 먼저 Get을 사용하여 확인한 후 이어진 작업을 Pipeline으로 처리한다
 - `Get-Process -Name notepad | Stop-Process`
 - `Get-Process | Where-Object {$_.vm -gt 100000} | Sort-Object -Property vm -Descending | Select-Object -First 5`

3 - PowerShell의 장점

- Cmdlet이 계속 늘어난다
 - 각 역할(Role)의 관리 도구 설치하면 해당 modul이 추가된다
 - **Get-Module** -ListAvailable
 - Import-Module** -Name ActiveDirectory
 - Get-Command** -Module ActiveDirectory
 - 현재 사용할 수 있는 cmdlet 수량을 확인한다
 - Get-Command | Measure-Object ## 또는 (Get-Command).Count
 - Get-Excommand | Measure-Object
- 명령어의 구문이 이해하기 쉽다
 - 명령어를 사용할 때는 Parameter를 잘 활용하는 것이 매우 중요하다
 - Verb-Noun
 - Verb-Noun **-Parameter value**
 - Verb-Noun -Parameter **value1, value2**
 - Verb-Noun -Parameter1 value **-Parameter2 value**
 - Verb1-Noun -Parameter value | **Verb2-Noun** | **Verb3-Noun**
 - Verb1-Noun ; Verb2-Noun

3 - PowerShell의 장점

- 도움말 기능이 탁월하다
 - PowerShell의 시작과 끝은 도움말에 있으므로 도움말을 정확히 이해하는 것이 매우 중요하다
 - 다음과 같이 하면 다양한 방법으로 도움말을 볼 수 있다
 - **Get-Help** -Name Get-Process
 - Help** -Name Get-Process
 - Help **Get-Process**
 - Help Get-Process **-Examples**
 - Help Get-Process **-Full**
 - Help Get-Process **-Detailed**
 - Help Get-Process **-Parameter** ComputerName
 - Help Get-Process **-Online**
 - Help Get-Process **-ShowWindow** (검색하기 좋음)
 - Get-Process **-?** (cmdlet를 사용하다가 갑자기 도움말을 볼 때)

3 - PowerShell의 장점

- 사용한 명령어와 관련 명령어를 쉽게 알 수 있다
 - Get-Process를 알게 되면 Get-Command *-Process를 해보면 Process가 들어가는 Verb를 모두 알 수 있다
 - Network과 관련된 명령어를 알고 싶으면 Get-Command *-net*
 - VM과 관련된 명령어를 알고 싶으면 Get-Command *-vm*
 - AD와 관련된 명령어를 알고 싶으면 Get-Command *-ad*
 - Help *-ad*로 알 수도 있다
- GUI 관리도구에는 없는 기능을 구현한다
 - GUI로 구현하기 전에 먼저 CLI를 먼저 구현한다
 - GUI에는 기능이 없다면 PowerShell에서 찾아서 해결한다
 - Active Directory Recycle Bin 기능
 - Hyper-V에서 Virtual Switch 관리 기능 구현

3 - PowerShell의 장점

- 자동화 관리 및 반복 작업에 유용하다
 - 매일, 1주일에 한 번, 로그인할 때 마다 등등 반복적으로 하는 작업 (예를 들면 30일마다 [C:\Temp](#) 폴더 파일 지우기)
 - 직원들이 출근하여 컴퓨터를 켜서 로그인 하거나 퇴근하기 위해서 컴퓨터를 종료할 때 상사에게 이메일을 자동으로 보내도록 Script를 설정하면 관리자는 직원들의 출퇴근 여부도 확인할 수 있다
 - 신입/경력 사원이 입사할 때마다 사용자 계정을 만들 때 Script 사용하거나, 사용자 계정을 한꺼번에 1000명을 만들기 위해서 GUI보다는 Script를 이용하면 금방 만들 수 있다
- Group Policy에서는 .vbs, .bat, ps1 파일을 사용한다
- 원하는 cmdlet이 없는 경우에 Script 및 Function을 만들어 사용할 수 있다

3 - PowerShell의 장점

- 원격 관리 기능이 탁월하다
 - 여러 개의 원격 컴퓨터에 현재 로그인한 사용자를 확인한다
 - Invoke-Command -ComputerName Server1, Server2, Server3 {**quser.exe**}
- 업무 시간에 직원들이 Gom Player를 사용하고 있는지 확인한다
 - Invoke-Command -ComputerName Server1, Server2, Server3 {**Get-Process -Name Gom**}

4 - PowerShell의 사용상의 특징

Tab 키를 사용한 명령어 자동 완성 기능

V3.0 이상에서 명령어 중간에도 Tab을 사용 가능

사용한 명령어의 이력(History) 확인하기

Alias 사용 가능

Profile 사용

cmd.exe에서 사용하는 모든 명령어를 사용한다

Module 자동 로드하여 사용하기

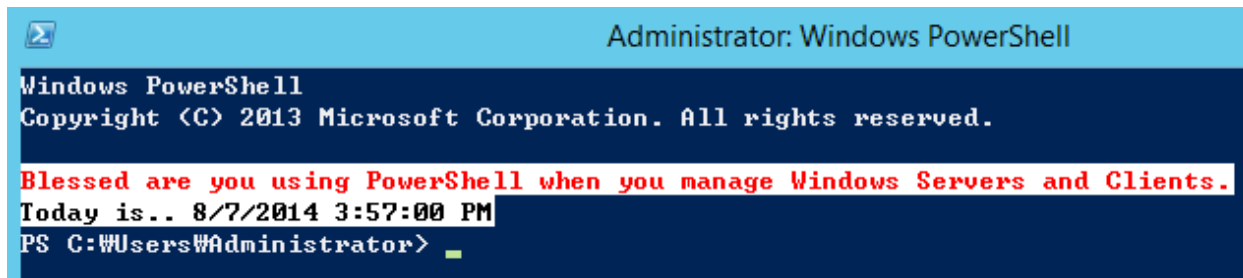
4 - PowerShell의 사용상의 특징

- Tab 키를 사용한 명령어 자동 완성 기능
 - Get-P <Tab>
- V3.0 이상에서 **명령어 중간에도 Tab을 사용 가능**
 - Start-Job -N <Tab> {Gwmi -Class Win32_Process}
 - 매개변수의 값을 모를 때는 그냥 Tab 키를 누르면 사용할 수 있는 값이 나타난다
Set-ExecutionPolicy -**ExecutionPolicy** <Tab>
- 사용한 명령어의 이력(History) 확인하기
 - Get-History (또는 H, F7키)
- Alias 사용 가능
 - Get-Alias
 - Get-Alias -**Name** dir
 - Get-Alias -**Definition** Get-ChildItem

4 - PowerShell의 사용상의 특징

- Profile 사용

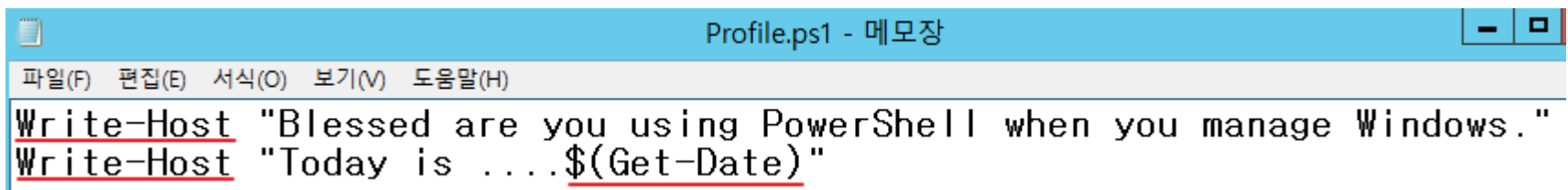
- 새로운 세션을 시작할 때 마다 Profile을 로딩하여 사용한다
- 늘 원하는 명령어 및 환경을 사용하고자 할 때 Profile을 설정하여 이용
- **\$Home\Documents\WindowsPowerShell\Profile.ps1**의 내용이 실행



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

Blessed are you using PowerShell when you manage Windows Servers and Clients.
Today is.. 8/7/2014 3:57:00 PM
PS C:\Users\Administrator>
```

- 원격 컴퓨터에 접속할 때는 Profile을 로드하지 않는다



```
Profile.ps1 - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
Write-Host "Blessed are you using PowerShell when you manage Windows."
Write-Host "Today is ....$(Get-Date)"
```

4 - PowerShell의 사용상의 특징

- cmd.exe에서 사용하는 모든 명령어를 사용한다
 - Get-Command -CommandType Application | Where-Object {\$_.name -like "*.exe"}
 - ipconfig, shutdown, nslookup, ping, net users, net view...
- Module 자동 로드하여 사용하기
 - Version 3.0 이상부터는 특정한 module이 메모리에 로딩되지 않아도 정확한 Cmdlet을 입력하면 자동으로 해당 Module이 메모리에 로딩되어 해당 명령어를 사용할 수 있다
 - 이를 위해서는 반드시 해당 cmdlet이 소속된 Module이 로컬 컴퓨터에 저장되어 있어야 한다

4 - PowerShell의 사용상의 특징

- Module 자동 로드하여 사용하기(계속)
 - 원칙적으로 cmdlet(Get-ADUser)이 실행되는 순서
 - 특정한 Module을 설치하기 위해 역할(Role)의 관리도구를 설치하여 해당 Module이 특정한 위치에 먼저 저장되어 있어야 함
 - Get-Module -ListAvailable**
 - Import-Module -Name ActiveDirectory**
 - Module을 디스크에서 메모리로 로드함
 - Get-Module**
 - Get-Command -Module ActiveDirectory**
 - cmdlet 실행
 - Get-ADUser -Identity administrator | fl ***

5 - PowerShell Version 및 기본 프로그램 사용

Windows PowerShell Version

Console 및 ISE

5 - PowerShell Version 및 기본 프로그램 사용

- Windows PowerShell Version

	2.0	3.0	4.0
Windows XP	Download하여 사용	사용 못함	사용 못함
Windows Server 2003	Download하여 사용	사용 못함	사용 못함
Windows Vista	Download하여 사용	사용 못함	사용 못함
Windows Server 2008	Download하여 사용	SP2인 경우 사용 가능	사용 못함
Windows 7	이미 설치됨	SP1인 경우 사용 가능	Download하여 사용
Windows Server 2008 R2	이미 설치됨	SP2인 경우 사용 가능	Download하여 사용
Windows 8	추가하여 사용	이미 설치됨	Download하여 사용
Windows Server 2012	추가하여 사용	이미 설치됨	Download하여 사용
Windows 8.1 and Windows Server 2012 R2	추가하여 사용	사용 못함	이미 설치됨

5 - PowerShell Version 및 기본 프로그램 사용

- PowerShell 버전에 따라 적합한 .NET Framework이 필수
- PS version 확인하기

- **\$PSVersionTable**

또는 Get-Host

```
PS C:\> $psversiontable
```

Name	Value
PSVersion	4.0
WSManStackVersion	3.0
SerializationVersion	1.1.0.1
CLRVersion	4.0.30319.34014
BuildVersion	6.3.9600.17090
PSCompatibleVersions	{1.0, 2.0, 3.0, 4.0}
PSRemotingProtocolVersion	2.2

- Windows 2012/2012R2에서 version2를 사용
 - 먼저 관리도구에서 version2를 추가한다
 - **Install-WindowsFeature -Name PowerShell-V2**
 - 그 다음 cmd.exe를 실행하여 **powershell.exe -version 2**를 입력하여 PowerShell을 version 2로 변경하여 사용한다

```
PS C:\> Install-WindowsFeature -Name PowerShell-V2
```

Success	Restart Needed	Exit Code	Feature Result
True	No	Success	<Windows PowerShell 2.0 Engine>

```
PS C:\> cmd
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\>powershell -version 2
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.
```

5 - PowerShell Version 및 기본 프로그램 사용

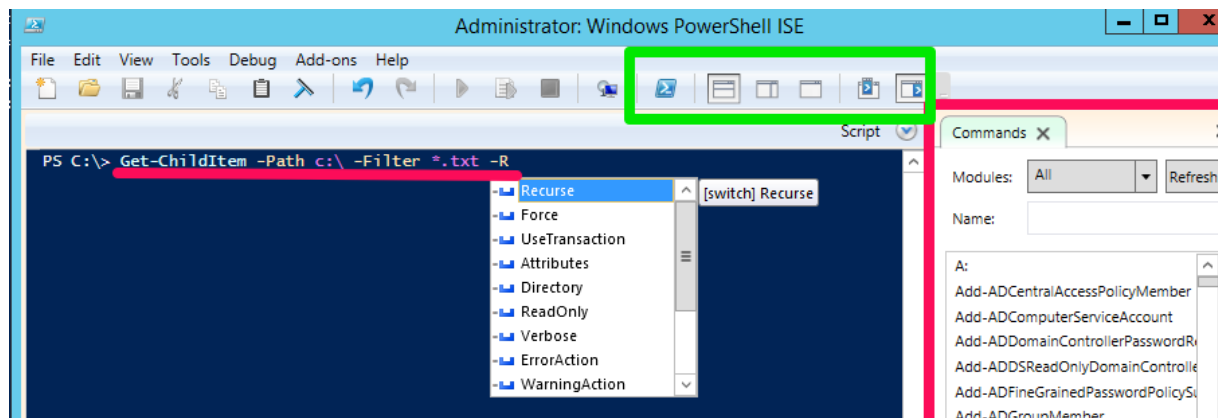
- Console

- 기본적인 command-line interface
- PowerShell 기능을 **최대 지원함**
- **최소한의 편집** 기능 제공



- ISE

- Script 편집기와 Console 기능을 합쳐 놓은 것임
- **강력한 편집** 기능 제공
- PowerShell Console에서 ise를 입력하여 실행하면 된다
- **PSEdit.exe** c:\script.ps1



6 – 쉬어가는 코너

Windows 명령어들을 찾아서 사용하기

cmdlet의 실행 결과를 파일로 저장하기

cmdlet의 demo 내용을 script로 만들어 실행하기

Windows 명령어들을 찾아서 사용하기

- Windows command 확인하여 사용하기
 - **Get-Command -CommandType Application | ? {\$_.name -like "*.exe"}**
 - Gcm **-Command** application | **Where-Object** {\$_.name -like "q*.exe"}
 - Gcm -Command application | **Where** {\$_.name -like "*.msc"}
 - Gcm -Command application | where {\$_.name -like "c*.msc"}
 - Gcm -Command application | where {\$_.name -like "*.cpl"}
 - Gcm -Command application | where {\$_.name -like "*.vbs"}
- Invoke-Command -ComputerName Server1 {**quser.exe**}
- **Ncpa.cpl**
- **Compmgmt.msc**

cmdlet의 실행 결과를 파일로 저장하기

- 실행한 cmdlet의 **목록과 그 결과**를 파일로 저장하기
 - **Start-Transcript C:\Day1.txt**
 - notepad
 - Get-Process -Name notepad
 - Get-Service | Where-Object {\$_.Status -eq "Running"}
 - **Stop-Transcript**
 - Notepad c:\Day1.txt

cmdlet의 실행 결과를 파일로 저장하기

- cmdlet **결과만**을 **파일로 저장하기**
 - Get-Process | **Out-File** C:\Lab\command.txt
 - Get-Process | **Out-File** C:\Lab\command.txt **-Append**
- cmdlet **결과만**을 **Clipboard에** 임시적으로 저장하기
 - Get-Process | **clip**
 - Notepad.exe를 실행하여 Ctrl+V로 붙여 넣기를 한다

cmdlet의 demo 내용을 script로 만들어 실행하기

- StartDemo.ps1를 사용하여 데모 코드를 실행
 - c:\demo라는 폴더를 생성한다
 - c:\demo 폴더에 StartDemo.ps1 파일과 실행할 명령어가 들어 있는 파일(demo1.txt/demo2.txt) 파일을 저장한다
 - 또는 c:\demo 하위에 각 Module별로 데모 파일을 txt 파일로 생성한다.
 - 예를 들면 c:\demo\module1 폴더 생성하고 m1-demo1.txt, m1-demo2.txt 파일을 생성한다
 - PowerShell console에서 c:\demo 폴더로 이동한 후 function 파일인 StartDemo.ps1 파일을 실행한다
 - **..\StartDemo.ps1**
 - 데모 코드 파일을 실행한다
 - **Start-Demo .\demo1.txt**
 - 각 Module에 있는 데모 코드 파일을 실행한다
 - **Start-Demo c:\demo\module1\m1-demo1.txt**

- Windows PowerShell Console 실행하여 사용하기
- ISE 사용하기
- 원하는 명령어 찾기
- Domain에 가입하기
- PowerShell v2.0 설치하여 사용하기
- Telnet Server 및 Telnet Client 설치하기

정리

- Windows PowerShell이란 무엇인가?
- PowerShell을 어떤 곳에 사용하는가?
- PowerShell의 장점
 - cmd.exe에서 사용하는 거의 모든 명령어를 사용 가능
 - 강력한 Pipeline 기능
 - Cmdlet이 계속 증가
 - 도움말이 탁월하다
 - 하나의 명령어를 알게 되면 연관 명령어를 쉽게 찾을 수 있다
 - GUI에 비하여 내용을 구체적으로 확인 및 상세히 설정 가능
 - 반복 작업에 유용하다
 - Group Policy를 통하여 PowerShell Script도 적용할 수 있다
 - 원하는 cmdlet이 없는 경우에 Script를 만들어 사용할 수 있다
 - Function, Module 등을 사용자가 필요에 따라서 생성할 수 있다
 - 원격 관리 기능이 탁월하다

정리

- Cmdlet을 사용할 때 꼭 기억할 사항
 - Server Manager의 Role 및 Feature를 설치할 때 **-Restart, -IncludeAllSubfeature, -IncludeManagementTools** 포함시킨다
 - Cmdlet의 Property를 활용하는 것이 중요하기 때문에 cmdlet | **Get-Member**, 또는 cmdlet | **Format-List ***를 통하여 Property를 알아낸다
 - Cmdlet 마다 사용할 수 있는 Parameter가 다르므로 **cmdlet의 Help**를 통하여 적절한 Parameter를 찾아서 사용한다
 - 명령 프롬프트 명령어를 잘 활용한다. (원격 및 로컬 컴퓨터 작업시)
 - 원하는 cmdlet 찾는 방법을 꼭 기억한다