



6장

Pipeline 활용하기

전체 내용

Provider와
Drive란
무엇인가?

PSDrive
활용하기

Pipeline 이해

Pipeline 처리1:
ByValue

Pipeline 처리2:
ByPropertyName

Pipeline을
사용하여 Active
Directory
관리하기

쉬어가는 코너

1 - Provider와 Drive란 무엇인가?

• 우리가 살고 있는 집에는 어떤 저장소가 있을까?

- 창고, 책장, 빨래 통, 전자제품 보관함, 화장품 함, 옷장
 - 물건을 분류하여 저장 하여야만 물건 관리가 잘 되기 때문이다
- 나중에 쓸 물건을 몽땅 창고에 넣어 두면 젖은 빨래, 전자 기기, 화장품이 뒤죽박죽이 되어서 물건이 훼손된다
- 이렇듯 Windows Server에도 다양한 용도와 목적으로 저장소가 다양하다



1 - Provider와 Drive란 무엇인가?

- Provider는 "데이터 저장소(data store)"이고, Drive는 "데이터 저장소(connected data store)에 연결된 것"을 말한다
 - 다양한 종류의 Data store가 있고(**Get-PSProvider**), 그 Data Store에 연결한 것을 Drive라고 한다(**Get-PSDrive**)

```
PS C:\> Get-PSProvider
```

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

- Provide: (원하는 것을 사용하도록) 미리 준비하다
 - to prepare **in advance**: **미리** 준비하다
 - to supply or make available (something that is wanted or needed): (원하는 것을) 사용할 수 있게 하다, 제공하다

2 – PSDrive 활용하기

- PSProvider의 장점은 **하나의 인터페이스**로 다양한 Data Store를 관리할 수 있다는 것이다
 - FileSystem, Alias, Registry, Environment, Variable, ActiveDirectory, WSMAN, Function, Certificate에 데이터를 저장할 수 있다는 말이다
 - 이러한 데이터 저장소에 액세스하여 일반적인 명령어를 사용 가능
 - **cd, pwd, dir, mkdir, del, move, ren, copy, Get-Item, Get-ItemProperty, Set-ItemProperty**
 - 사용 예
 - **Import-Module -Name ActiveDirectory**
 - **Get-PSDrive / cd AD: / dir / dir "dc=mctconnected,dc=com"**
 - **dir / cd "ou=lab" / dir / cd "ou=sales" / dir**
- PSProvider의 장점은 어떤 저장소가 생기더라도 관리할 수 있다는 것이다. 예를 들면, IIS를 설치하면 그것도 Data Store로 관리할 수 있다
- FileSystem과 유사하게 계층적으로 Data Store에 접근할 수 있다

2 – PSDrive 활용하기

- **Get-PSProvider: PSProvider 목록 확인하기**

- 다양한 종류의 Provider를 확인 가능하다
- **Help <PSProvider이름>**를 통하여 각 데이터 저장소에 대한 자세한 내용을 알 수 있다
 - Help WSMAN
 - Help Registry

- **New-PSDrive: 새로운 PSDrive 생성하기**

- **New-PSDrive -Name "이름" -PSProvider FileSystem -Root "경로"**
- New-PSDrive -Name **P** -PSProvider **FileSystem** -Root [\\Server1\Public](#)
 - P 드라이브가 윈도우 탐색기에 나타나지 않는다
- New-PSDrive S -PSProvider FileSystem -Root \\Server1\Scripts **-Persist**
 - **-Persist**를 사용하면 윈도우 탐색기에 나타나기 때문에 반드시 영문자 한 글자를 사용해야 한다. 이것은 세션 기반이 아니다
- New-PSDrive -Name MyDocs -PSProvider FileSystem -Root "\$home\Documents" **-Description** "Maps to my Documents folder."

2 – PSDrive 활용하기

- **Remove-PSDrive**

- **Remove-PSDrive -Name 이름 -Force**
- Remove-PSDrive -Name MyDocs -Force
- Get-PSDrive -Name P, S | Remove-PSDrive -Force

- **New-PSDrive 활용하기**

- New-PSDrive -Name MyCompany -PSProvider **Registry** -Root HKLM:\Software\MyCompany
- Script 또는 Profile에 New-PSDrive를 만들어 File server에 접속하여 윈도우 탐색기에 보이지 않게 파일을 복사할 수 있다
- 기존의 드라이브를 확인 및 제거한 후 다시 생성하는 스크립트
 - \$Network = [\\DC1\SharedAlias](#)
 - **If** ((Get-PSDrive [a-z]) -Match 'X') {Remove-PSDrive X}
 - New-PSDrive -Name **X** -PSProvider FileSystem -Root \$Network **-Persist** | Format-Table Name, DisplayRoot, Root -AutoSize
 - **Start-Sleep 3**
 - **Invoke-Item X:**

2 – PSDrive 활용하기

- **Get-Item, Get-ItemProperty, Set-Item 사용하기**

- **Get-Item:** 특정한 위치에 있는 Item을 말한다. (파일과 폴더, 키)
- **Get-ItemProperty:** Item, 파일과 폴더의 속성을 말한다. 즉, archive, hidden 등등을 말한다.
- Get-PSDrive에 보이는 Data Store(Registry, IIS)를 이 명령어로 활용
- PSDrive의 있는 저장소를 Get-ItemProperty, Set-ItemProperty를 사용하여 설정을 확인하고 변경할 수 있다

- **사용 예제**

- **(Get-Item C:\Lab).LastAccessTime**
- \$version=**Get-ItemProperty** 'HKLM:\Software\Microsoft\Windows NT\CurrentVersion'
Write-Host "이 컴퓨터를 등록한 사람은..." \$version.RegisteredOwner
- **Cd HKLM:**
Get-ItemProperty
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto Update\RebootRequired"

3 - Pipeline 개요

- **Pipeline이란 무엇인가?**

- 앞 명령어의 결과를 뒤 명령어의 입력 값으로 처리하는 것이다
- “현재 등록된 서비스 중에서 중지된 서비스”를 알고자 하는 경우
 - **Get-Service | Where {\$_.Status -eq “Stopped”} | Sort-Object name**

- **Pipeline이 실패하는 예제**

- “DC1”, “Server2” | Get-Service
- Get-Service | Stop-Process
- Get-ADComputer -Filter * | Get-Service
- Test-Connection -ComputerName Server2 | Restart-Computer

3 - Pipeline 개요

- 어떤 경우에 명령어에 **Pipeline**을 사용할 수 있는가?
 - 로컬 컴퓨터에서 실행중인 **bits** 서비스를 중지하기 위해서는...
 - **Stop-Service -Name** bits
 - **Stop-Service -DisplayName** "Background Intelligent Transfer Service"
 - **Get-Service -Name** bits | **Stop-Service**
 - **Get-Service -DisplayName** "Background Intelligent Transfer Service" | **Stop-Service**
 - 메모리에서 실행중인 **notepad** 프로세스를 중지하기 위해서는...
 - **Stop-Process -Name** notepad
 - **Stop-Process -ID** 7552
 - **Get-Process -Name** notepad | **Stop-Process**
 - **Get-Process -ID** 7552 | **Stop-Process**
 - **Stop-Process**를 직접 입력하는 것보다는 **Get-Process**를 먼저 실행하여 결과를 확인하여 그 다음 작업하는 것이 안전하고, 다양하게 응용할 수 있기 때문에 Pipeline을 사용하는 것이다
 - 그렇다면 어떤 원리로 Pipeline이 작동하는 것일까?

3 - Pipeline 개요

• 어떤 순서로 Pipeline이 처리되는가?

- Stop-Service를 입력한 후 어떠한 **Parameter**와 **value**를 사용할지를 결정해야 한다
- 이런 경우, 앞의 명령어의 결과에서 그 Parameter와 Value를 가져와서 처리하는 것이 바로 Pipeline이다
- 그 방법을 알기 위해서 다음과 같이 순서대로 Pipeline을 처리한다
 - ① 우선 cmdletB의 [상세한 도움말]에서 **[-InputObject]**가 존재해야 함
 - ② 이것이 충족된 후에는 cmdletB의 [상세한 도움말]에서 **[-InputObject]**의 세부 항목인 **[파이프라인 입력 적용 여부]**가 **True(By Value)**이어야 한다
 - ③ cmdletB의 [상세한 도움말]에서 **[-InputObject]**의 **Value**(예:ServiceController)와 cmdletA의 **Get-Member의 Class Name이 동일**해야만 그 **Parameter의 Value**를 **ByValue**로 Pipeline을 처리할 수 있다
 - ④ 이것이 실패하는 경우에, 그 대안으로서 **ByPropertyName**를 진행한다
 - ⑤ **cmdletB의 Parameter 이름**과 **cmdletA의 Property 이름**을 비교하여 **동일한 경우에** Pipeline을 처리하는 것이 **ByProperty**로 처리하는 것이다
(cmdletA의 Property 이름으로 처리하는 것을 ByPropertyName이라고 하는 것이다)

3 - Pipeline 처리1: ByValue

- 어떤 원리로 Pipeline을 처리하는가?

- 뒤(後) 명령어인 Stop-Service의 [도움말]을 볼 때, 첫 번째 Parameter Set에 **Stop-Service [-InputObject] <ServiceController[]>**라고 되어 있다

구문

```
Stop-Service [-InputObject] <ServiceController[]>  
[-Exclude <String[]>] [-Force] [-Include <String[]>]  
[-PassThru] [-Confirm] [-WhatIf]  
[<CommonParameters>]
```

- Stop-Service [-InputObject] <ServiceController[]>**은 **Stop-Service -InputObject (Get-Service -Name bits)**를 말한다
즉, Get-Service | GM의 결과가 반드시 Class가 ServiceController이어야 한다는 것이다

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

```
TypeName: System.ServiceProcess.ServiceController
```

3 - Pipeline 처리1: ByVal

- 어떤 원리로 Pipeline을 처리하는가?

- Stop-Service -InputObject (Get-Service -Name bits)을 다르게 표현하면 **Get-Service -Name bits | Stop-Service**로 할 수 있다.
- 우리가 궁극적으로 처리하는 것은 Stop-Service인데, Stop-Service -Name bits로 표현할 것을 Pipeline으로 표현하면 Get-Service -Name bits | Stop-Service이 된다. 둘 다 같은 결과를 가져오지만 후자가 좋다
- Get-Service -Name bits | Stop-Service를 **A | B**로 단순화한다면, B의 결과를 내기 위해서...
 - 1) A의 **Class 이름**과 B의 **-InputObject의 value**가 [동일]해야 한다
 - 2) 그럴 경우, **B의 -InputObject**의 [파이프라인 입력 적용 여부]가 True이고, (ByValue)인 경우, **A의 Parameter** 중에서 [파이프라인 입력 적용 여부]가 ByValue인 Parameter가 있다면 그것의 값을 가져와서 처리한다.
- 여기서는 A인 Get-Service의 **-Name이 ByValue**이다

```
-Name <String[]>  
    Specifies the service names of services to be retrieved.  
    Wildcards are permitted. By default, Get-Service gets all of the  
    services on the computer.
```

필수 여부	false
위치	1
기본값	All services
파이프라인 입력 적용 여부	true (ByPropertyName, ByValue)

3 - Pipeline 처리1: ByVal

- 어떤 원리로 Pipeline을 처리하는가?

- 결론적으로 B의 Stop-Service는 **Stop-Service -Name bits**가 되기 때문에 정상적으로 처리가 되는 것이다
- Get-Service **-DisplayName "Background Intelligent Transfer Service"**
| **Stop-Service**은 어떻게 처리될까?
앞의 것과 동일하게 처리된다.
 - 1) Get-Service의 Class 이름이 ServiceController이고, Stop-Service의 -InputObject의 값이 ServiceController[]이므로 동일하다
 - 2) **B인 Stop-Service의 -InputObject의 [파이프라인 입력 적용 여부]가 True(ByValue)이다**
 - 3) **A인 Get-Service의 Parameter 중에서 [파이프라인 입력 적용 여부]가 True(ByValue)인 것은 -Name 매개변수(Parameter)이다**
 - 4) 결론적으로 Stop-Service **-Name bits**가 되므로 정상적으로 처리된다

3 - Pipeline 처리1: ByValue

- **Plan A - ByValue**

- 정리하면...

Get-Service -Name BITS | Stop-Service를 처리할 때

- 1) Stop-Service의 -InputObject의 value와 Get-Service의 class가 동일한지를 검사한다. 동일하지 않으면 Pipeline을 사용할 수 없다

"ServiceController"

- 2) Stop-Service의 -InputObject의 [파이프라인 입력 적용 여부]를 확인한다.

"ByValue"

- 3) Get-Service의 Parameter 중에서 [파이프라인 입력 적용 여부]가 True(ByValue)인 Parameter를 찾아서 Stop-Service에서 사용하게 된다

"-Name"

- 결론적으로 | Stop-Service의 숨겨진 Parameter는 -Name이고, 이 -Name의 value는 Get-Service의 -Name 값이다

"Stop-Service -Name BITS"

4 - Pipeline 처리2: ByPropertyName

- 어떤 원리로 Pipeline을 처리하는가?

- Get-Process -Name notepad | Stop-Process**은 어떻게 처리되는가?
- 뒤 명령어인 Stop-Process의 [도움말]을 볼 때, 두 번째 Parameter Set에 **Stop-Process [-InputObject] <process[]>**라고 되어 있다

구문

```
Stop-Process [-Id] <Int32[]> [-Force ] [-PassThru ]  
[-Confirm ] [-WhatIf ] [<CommonParameters>]
```

```
Stop-Process [-InputObject] <Process[]> [-Force ]  
[-PassThru ] [-Confirm ] [-WhatIf ]  
[<CommonParameters>]
```

```
Stop-Process [-Force ] [-PassThru ] -Name <String[]>  
> [-Confirm ] [-WhatIf ] [<CommonParameters>]
```

- Stop-Process [-InputObject] <process[]>**는 **Stop-Process -InputObject (Get-Process -Name notepad)**를 말한다 즉, Get-Process | GM의 결과가 반드시 Class가 Process이어야 한다는 것이다 (확인해 보면 맞다)

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

```
TypeName: System.Diagnostics.Process
```


4 - Pipeline 처리2: ByPropertyName

• 어떤 원리로 Pipeline을 처리하는가?

- Stop-Process -InputObject (Get-Process -Name notepad)를 다르게 표현하면 **Get-Process -Name notepad** | Stop-Process로 할 수 있다.
- **Get-Process -Name notepad** | Stop-Process을 A | B로 단순화한다면, B의 결과를 내기 위해서...
 - 1) A의 **Class 이름**과 B의 **-InputObject의 value**가 [동일]해야 한다
 - 2) B의 -InputObject의 [파이프라인 입력 적용 여부]가 **True(ByValue)**인 경우, **A의 Parameter** 중에서 [파이프라인 입력 적용 여부]가 **True(ByValue)인 Parameter가 없다.**

```
-InputObject <Process[]>
  Stops the processes represented by the specified process
  objects. Enter a variable that contains the objects, or type a
  command or expression that gets the objects.

필수 여부          true
위치                1
기본값
파이프라인 입력 적용 여부 true (ByValue)
와일드카드 문자 적용 여부 false
```

- 이런 경우에는 Plan B로 넘어 간다. 즉, ByPropertyName으로 해결한다

4 - Pipeline 처리2: ByPropertyName

- 어떤 원리로 Pipeline를 처리하는가?

- Get-Process -Name notepad | Stop-Process
- 이 명령어는 ByValue로 처리된 것이 아니고, ByPropertyName으로 처리된 것이다 (Stop-Process -Name notepad)
- Stop-Process 다음에 어떤 Parameter가 올 것인지 알아야 한다. Plan A로는 실패했기 때문에 Plan B를 시도한다
- Plan B는 Get-Process의 Property 이름(ProcessName=Name)과 Stop-Process의 Parameter 이름(Name)이 같게 하는 것이다. 여기서는 다행히 동일하다(**ByPropertyName**)

```
PS C:\#> Get-Process -Name notepad
```

Handles	NPM(K)	PM(K)	WS(K)	VM(M)	CPU(s)	Id	ProcessName
128	11	2196	10016	100	0.05	4512	notepad

```
PS C:\#> Help Stop-Process -Parameter Name
```

```
-Name <String[]>
```

Specifies the process names of the processes to be stopped. You can use wildcards or use wildcard characters.

필수 여부 true

위치 named

기본값

파이프라인 입력 적용 여부 true (ByPropertyName)

와일드카드 문자 적용 여부 true

4 - Pipeline 처리2: ByPropertyName

- 어떤 원리로 Pipeline를 처리하는가?

- Get-Process -ID 4512 | Stop-Process
- 이 명령어는 ByValue로 처리된 것이 아니고, ByPropertyName으로 처리된 것이다 (Stop-Process -Name notepad)
- Stop-Process 다음에 어떤 Parameter가 올 것인지 알아야 한다. Plan A로는 실패했기 때문에 Plan B를 시도한다
- Plan B는 Get-Process의 Property 이름과 Stop-Process의 Parameter 이름이 같게 하는 것이다. 여기서는 다행히 동일하다(**ByPropertyName**)

```
PS C:\> Get-Process -Id 4512
```

Handles	NPM(K)	PM(K)	WS(K)	VM(M)	CPU(s)	Id	ProcessName
128	11	2196	10016	100	0.05	4512	notepad

```
PS C:\> Help Stop-Process -Parameter Name
```

-Name <String[]>
Specifies the process names of the processes to be stopped. You can
s) or use wildcard characters.

필수 여부	true
위치	named
기본값	
파이프라인 입력 적용 여부	true (ByPropertyName)
와일드카드 문자 적용 여부	true

4 - Pipeline 처리2: ByPropertyName

- **Plan B – ByPropertyName(1)**

- C:\Lab\Alias.csv 파일을 아래와 같이 생성한다
Name,Value
d,Get-ChildItem
sel,Select-Object
go,Invoke-Command
- 내용을 확인하기 위해서 Import-Csv -Path C:\Lab\Alias.csv
- **Import-csv -Path C:\Lab\Alias.csv | New-Alias**
- **Help New-Alias**하면 **-InputObject** Parameter가 없다.(중요) 이렇다면 기본적으로 ByValue로는 처리할 수 없다. 그러면 Plan B로 처리해야 한다
- **Import-csv -Path C:\Lab\Alias.csv | Get-Member**를 실행하면
TypeName: System.Management.Automation.PSCustomObject이 나온다.
- **Import-Csv**의 **Property**와 **New-Alias**의 **Parameter** 중에서 동일한 것을 찾는 것이 Plan B이다

4 - Pipeline 처리2: ByPropertyName

- **Plan B – ByPropertyName(1)**

- **Import-csv -Path C:\Lab\Alias.csv | New-Alias**에서 New-Alias를 실행하기 위해서는 Parameter가 필요한데, 여기서는 New-Alias에서 -InputObject가 없기 때문에 앞 명령어의 PropertyName을 그대로 사용한다

```
PS C:\Lab> Import-csv -Path C:\Lab\Alias.csv | Get-Member
```

Property

Type: System.Management.Automation.PSCustomObject

Name	MemberType	Definition
Equals	Method	bool Equals(System.Object obj)
GetHashCode	Method	int GetHashCode()
GetType	Method	type GetType()
ToString	Method	string ToString()
Name	NoteProperty	System.String Name=d
Value	NoteProperty	System.String Value=Get-ChildItem

```
PS C:\Lab> Help New-Alias
```

이름
New-Alias

개요
Creates a new alias.

구문
New-Alias **[-Name]** <String> **[-Value]** <String>
hru] [-Scope <String>] [-Confirm] [-WhatIf] [

- 앞 명령어의 Name, Value라는 Property와 뒤의 명령어의 Parameter 이름이 동일하기 때문에 사용할 수 있다
- Import-Csv의 Name, Value의 value를 그대로 가져와서 New-Alias를 실행한다. 즉, New-Alias -Name sel -Value Select-Object가 되는 것이다

4 - Pipeline 처리2: ByPropertyName

- **Plan B – ByPropertyName(1)**

- Get-Process -Name notepad | Stop-Process 또는 Import-csv -Path C:\Lab\Alias.csv | New-Alias은 **ByValue가 안 된다**. 그 이유는 각각 다르다. 하지만 ByValue가 안되면 ByPropertyName으로 처리하면 된다
- ByPropertyName으로 처리할 때는 반드시 **앞의 Property 이름**과 **뒤의 Parameter 이름**이 동일하도록 해야 한다는 것이다

4 - Pipeline 처리2: ByPropertyName

- **Plan B – ByPropertyName(2)**

- 가장 많이 사용하는 *-Object 및 Format-*라 들어 가는 cmdlet들의 Pipeline에 대하여 알아보자
(Select-Object, Sort-Object, Group-Object, Format-Table)
- **Get-Process | Select-Object -Property Name**
- 두 명령어는 class가 맞지 않으므로 ByPropertyName으로 가야 한다
- Help Select-Object -Parameter InputObject를 하면 value가 <PSObject>로 나온다. 즉, 입력 값으로 아무 Object를 사용해도 된다
- 결론은 이런 것들은 지정한 Property의 이름(name,pm,pid 등등)으로 처리한다는 것이다.

4 - Pipeline 처리2: ByPropertyName

- **Plan B – ByPropertyName(2)**

- Where-Object에 대하여 알아보자
- Get-Service | Where-Object -FilterScript {\$_.Status -eq "Stopped"}
- Get-Service | Where-Object -Property Status -eq -Value "Stopped"
(version 3.0에서부터 지원)
- -FilterScript는 Object를 Filtering을 하는 것이다
- Where-Object, Where, ?도 원하는 Property를 기준으로 연산자를 통하여 Filtering하여 정보를 알아내는 것이다

4 - Pipeline 처리2: ByPropertyName

- **Plan B - ByPropertyName**

- 정리하면...

Get-Process -Name notepad | Stop-Process

Import-csv -Path C:\Lab\Alias.csv | New-Alias

Get-Service | Where-Object -FilterScript {\$_.Status -eq "Stopped"}

- ByValue로 처리되지 않으면 ByPropertyName으로 처리하면 된다
- ByPropertyName으로 처리할 때는 A의 Property 이름과 B의 Parameter 이름을 동일하게 맞춰주면 된다
- *-Object인 것은 A의 Property 이름을 사용하면 된다

5 – Pipeline을 사용하여 Active Directory 관리하기

- **Active Directory 사용 계정을 한꺼번에 생성하기**

- 사용자 계정을 하나씩 생성하기 보다는 수백 개의 계정을 한 번에 생성할 수 있는 방법이 바로 Pipeline을 사용하는 것이다
- `Help New-ADUser -ShowWindow`
- `C:\Lab\multiusers1.csv` 파일을 생성한다
 - **Name,SamAccountName,Department,Title**
AD User100,aduser100,Training,Instructor
AD User200,aduser200,HR,Daeri
AD User300,aduser300,Administration,Sawon
- `Import-Csv c:\Lab\multiuser.csv | New-ADUser`
- `Get-ADUser -Filter * | ft name`
 - 여기서 중요한 것은 제일 위에 쓴 Name, SamAccountName, Department, Title이 New-ADUser의 Parameter 이름과 동일해야 한다는 것이다

5 – Pipeline을 사용하여 Active Directory 관리하기

- **Active Directory 사용 계정을 한꺼번에 생성하기**

- 이제는 한 번 더 만들어 본다.
- C:\Lab\multiusers2.csv 파일을 생성한다. 그런데 SamAccountName 대신 LogonID를 사용했다. 이것은 New-ADUser의 Parameter에 없는 것이다
 - **Name, LogonID, Department, Title**
AD User101,aduser101,Training,Instructor
AD User201,aduser201,HR,Daeri
AD User301,aduser301,Administration,Sawon
- Import-Csv c:\Lab\multiuser2.csv | New-ADUser (실패)
- 이럴 때는 **Custom Property**를 이용하면 된다
 - Import-Csv C:\Lab\multiusers2.csv | **Select-Object -Property * , @{n='SamAccountName';e={\$_.LogonID}}** | New-ADUser (성공)

5 – Pipeline을 사용하여 Active Directory 관리하기

- **Active Directory 컴퓨터 계정 이용하기**

- AD의 컴퓨터 계정을 이용하여 원하는 정보를 불러 온다
 - \$ad=Get-ADComputer -Filter *
\$ad | Select-Object -Property Name
\$ad | Select-Object -Property Name | Get-Service -Name BITS (실패)
 - \$ad | Select-Object -**ExpandProperty** Name
\$ad | Select-Object -**ExpandProperty** Name | Get-Service -Name BITS
(실패)
 - \$ad | Select-Object -Property @{n="**ComputerName**";e={\$_.name}} | Get-Service -Name BITS (성공!)
- 괄호를 이용하여 실행 (괄호부터 먼저 실행)
Get-Service -ComputerName (\$ad | Select -**ExpandProperty** Name)
(성공!)

5 – Pipeline을 사용하여 Active Directory 관리하기

- **String으로 컴퓨터 이름을 불러 오는 방법: 괄호 사용하기**
 - 안되는 방법
 - `Get-Content d:\computers.txt | Get-WmiObject -Class Win32_Bios`
 - 되는 방법
 - .txt 파일 사용하기
`Get-WmiObject -Class Win32_Bios -ComputerName (Get-Content d:\computers.txt)`
 - .csv 파일 사용하기
`Get-Process -ComputerName (Import-csv c:\com.csv | Select-Object -ExpandProperty hostname)`
 - AD에 있는 컴퓨터 목록 불러 오기
`Get-Hotfix -ComputerName (Get-ADComputer -Filter * | Select-Object -ExpandProperty name)`

6 – 쉬어가는 코너

Start-Sleep 사용하기

Get-Random 사용하기

6 – 쉬어 가는 코너

- **Start-Sleep**

- 지정한 시간 동안 잠시 다른 명령어의 실행을 멈춘다
- Script 작성할 때 유용하게 사용할 수 있다
- **30초 후에 컴퓨터 종료하기**

Start-Sleep -Seconds 30 ; Stop-Computer -Force

- **사다리 타기**

Start-Sleep -S 3; Write-Host "과연...." ; Start-Sleep -S 2 ; Get-Random "1등","사러 가기","돈 내기" ; Start-Sleep -S 2 ; Write-Host "당첨을 축하해요. ㅋㅋㅋ"

- **5초 후에 현재 실행중인 Spooler 서비스 재시작하기**

Gsv Spooler ; Start-Sleep -S 2 ; Stop-Service Spooler -Force ; Gsv Spooler ; Start-Sleep -s 5 ; Start-Service Spooler ; Gsv Spooler

6 – 쉬어 가는 코너

- **Get-Random**

- 무작위로 선택한다
- Get-Random
- Get-Random -Max 10
- Get-Random -Min 5 -Max 20
- Get-Random -InputObject 6, 7, 0, 5, 2, 6
- Get-Random -InputObject 6, 7, 0, 5, 2, 6 -Count 2
- Get-Random -InputObject "Faith", "Hope", "Love"
- Start-Sleep -S 3; Write-Host "과연...." ; Start-Sleep -S 2 ; Get-Random "1등","사러 가기","돈 내기" ; Start-Sleep -S 2 ; Write-Host "당첨을 축하해요. ㅋㅋㅋ"
- Get-Process | Get-Random
- dir -Path C:* -Recurse | Get-Random -Count 50

- PSProvider 및 PSDrive 사용하기
- PSProvider에 접속하여 IIS Web Site를 설정 변경하기

정리

- Provider와 Drive란 무엇인가?
- PSDrive 활용하기
- Pipeline 처리1: By Value
- Pipeline 처리2: By PropertyName