PowerShellについて

by むたぐち（牟田口大介）
Microsoft MVP
for Data Center Management -
Admin Frameworks

Windows PowerShellとは

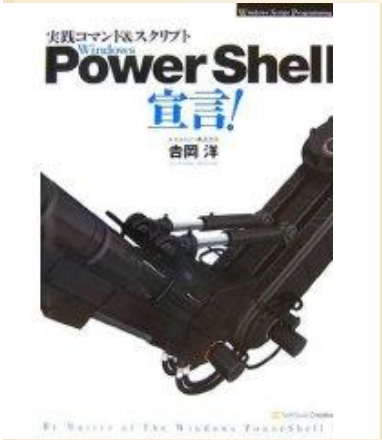
- Windows PowerShellとは
- .NET Frameworkをベースに動作する、Windowsの新しいシステム管理用シェル＆スクリプト実行環境
- Windows Server2003/XP/Vista用がダウンロード可能。Server 2008には標準搭載
- ver 2.0 CTP2も出ました

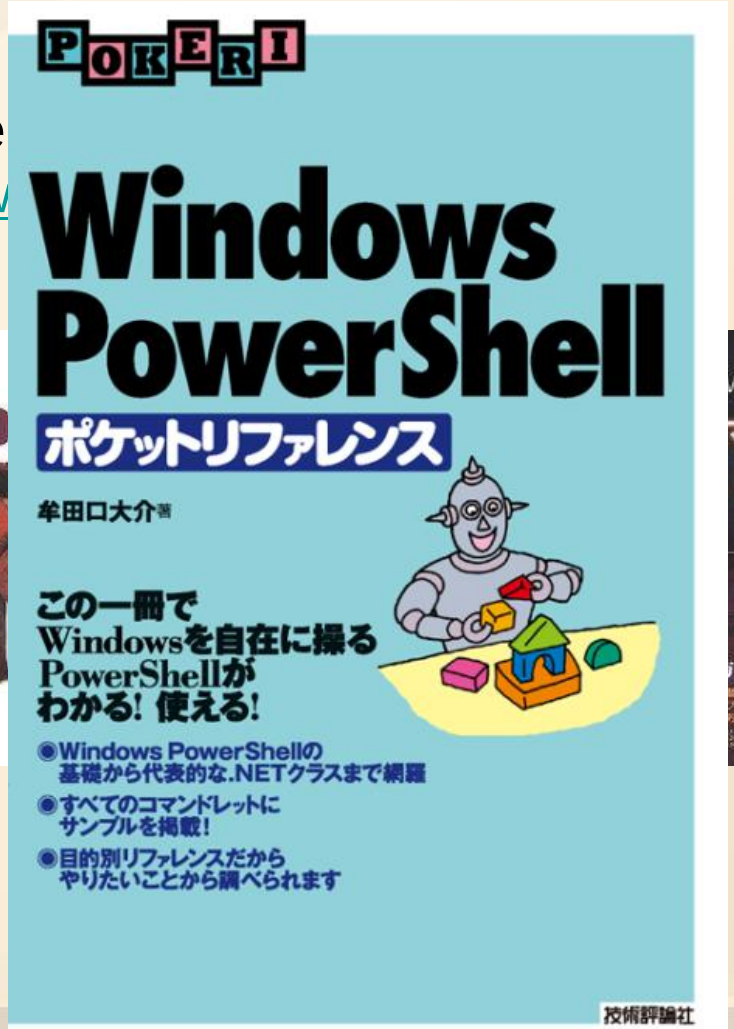
PowerShellの情報源（Web）

- Windows PowerShell でのスクリプティング
  http://www.microsoft.com/japan/technet/scriptcenter/hubs/msh.mspx
- PowerShell Memo
  http://d.hatena.ne.jp/newpops/
- Shigeya Tanabe's blog
  http://blogs.technet.com/stanabe/default.aspx
- その他紹介記事、初心者向け記事数点（@IT、ITPro、codezineなど）
- PowerShell Scripting（実質リンク集）
  http://www.roy.hi-ho.ne.jp/mutaguchi/powershell/
- Scripting Weblog
  http://blogs.wankuma.com/mutaguchi/ 上記2サイトへのポータルhttp://winscript.jp/

PowerShellの情報源（書籍）

- 雑誌
  - Windows Se
    http://www.w
- 書籍





PowerShellの基本・コマンドレット

- コマンドプロンプトの内部コマンド(cdとかdirとか)に相当する129種のコマンドレット（cmdlet）を組み合わせて使うのが基本です。
- 大きく分けてPSドライブ操作、PSユーティリティ、システム管理機能呼び出し、オブジェクトの操作の4種類（むたぐち分類法）。
- コマンドレットの引数（パラメータ）も戻り値もみな.NETのオブジェクトである。だからメソッドを呼び出したりもできる。

コマンドレットの基本(1)　命名法

- コマンドレット命名法は

  ”Verb-Noun”（動詞-名詞）

- 例：ディレクトリを移動する**Set-Location**コマンドレット(コマンドプロンプトのcdに相当)

```
Windows PowerShell
Copyright (C) 2006 Microsoft Corporation. All rights reserved.

PS C:¥Users¥daisuke> Set-Location -Path C:¥
PS C:¥>
```

コマンドレットの分類表

・PSドライブ操作

項目の内容を操作するには
Add-Content
Clear-Content
Get-Content
Set-Content

項目を操作するには
Get-ChildItem
Clear-Item
Copy-Item
Get-Item
Move-Item
New-Item
Remove-Item
Rename-Item
Set-Item

プロパティ操作を行うには
Clear-ItemProperty
Copy-ItemProperty
Get-ItemProperty
Move-ItemProperty
New-ItemProperty
Remove-ItemProperty
Rename-ItemProperty
Set-ItemProperty

項目を実行するには
Invoke-Item

PSドライブを扱うには
Get-PSDrive
Get-PSProvider
New-PSDrive
Remove-PSDrive

ロケーションを取得・設定するには
Get-Location
Set-Location
Push-Location
Pop-Location

パスの操作を行うには
Convert-Path
Resolve-Path
Join-Path
Split-Path
Test-Path

ACLの取得・設定を行うには
Get-Acl
Set-Acl

・PSユーティリティ

ヘルプを取得するには
Get-Help

コマンドレットの一覧を取得するには
Get-Command

ファイルまたは文字列から指定の文字列を抜き出すには
Select-String

変数の値を操作するには
Get-Variable
Set-Variable
New-Variable
Remove-Variable
Clear-Variable

履歴を参照・追加・実行するには
Add-History
Get-History
Invoke-History

エイリアスを操作するには
Get-Alias
Set-Alias
New-Alias
Export-Alias
Import-Alias

メッセージを書き込むには
Write-Host
Write-Output
Write-Verbose
Write-Warning
Write-Debug
Write-Error

文字列の入力を読み取るには
Read-Host

暗号化文字列・セキュア化された文字列を扱うには
ConvertFrom-SecureString
ConvertTo-SecureString

プログレスバーを表示するには
Write-Progress

シェル・スクリプトを中断するには
Start-Sleep

デバッグを行うには
Set-PSDebug
Start-Transcript
Stop-Transcript
Get-TraceSource
Set-TraceSource
Trace-Command

コマンドの実行時間を計測するには
Measure-Command

文字列をコマンドとして実行するには
Invoke-Expression

ホストオブジェクトを参照するには
Get-Host

設定ファイルを読み込むには
Update-FormatData
Update-TypeData

実行ポリシー取得・設定するには
Get-ExecutionPolicy
Set-ExecutionPolicy

スクリプトファイルの署名を取得・設定するには
Get-AuthenticodeSignature
Set-AuthenticodeSignature

Get-PfxCertificate

PSスナップインを操作するには
Get-PSSnapin
Add-PSSnapin
Remove-PSSnapin
Export-Console

・管理機能呼び出し

日付時刻を取得・設定するには
Get-Date
Set-Date

TimeSpanオブジェクトを作成するには
New-TimeSpan

サービスを操作するには
Get-Service
Start-Service
Stop-Service
Restart-Service
Suspend-Service
Resume-Service
Set-Service
New-Service

WMIのオブジェクトを扱うには
Get-WmiObject

資格情報を取得するには
Get-Credential

プロセスを取得・停止するには
Get-Process
Stop-Process

イベントログの情報を取得するには
Get-EventLog

カルチャ、UIカルチャに関する情報を取得するには
Get-Culture
Get-UICulture

・オブジェクト操作

オブジェクトをフィルタ・加工するには
Where-Object
Tee-Object
Sort-Object
Select-Object
Get-Unique
Group-Object

.NETかCOMのオブジェクトを作成するには
New-Object

オブジェクトを計測するには
Measure-Object

オブジェクトのメンバを列挙するには
Get-Member

オブジェクトにメンバを追加するには
Add-Member

オブジェクトの比較を行うには
Compare-Object

オブジェクトを列挙するには
ForEach-Object

オブジェクトの出力の形式を変更するには
Format-List
Format-Table
Format-Wide
Format-Custom

オブジェクトを様々なデバイス、ファイル、文字列に出力するには
Out-Default
Out-Host
Out-File
Out-Null
Out-Printer
Out-String

オブジェクトをファイルに出力・ファイルから入力するには
Export-Clixml
Import-Clixml
Export-Csv
Import-Csv
ConvertTo-Html

コマンドレットの基本(2)　使い方

- もっとも単純な例
  - PS >【コマンドレット名】　ex) Get-Date
- パラメータを取る場合
  - PS >【コマンドレット名】【-パラメータ名】 ex) Get-ChildItem -force
- パラメータと値を取る場合
  - PS >【コマンドレット名】【-パラメータ名】【パラメータ】
  - ex) Get-Command -type Cmdlet
- 複数パラメータを取る場合
  - PS >【コマンドレット名】【-パラメータ名1】【パラメータ1 】【-パラメータ名2】【パラメータ2】 ex) Get-Eventlog -LogName system -Newest 5
- パラメータ名を省略した場合
  - PS >【コマンドレット名】【パラメータ1 】【パラメータ2】
  - ex) Rename-Item a.txt b.txt

コマンドレットの基本(3)　ヘルプ

- どんなコマンドレットがあるのかを調べるには

  Get-Command

- コマンドレットのヘルプを引くには

  Get-Help コマンドレット名

  または

  コマンドレット名 -?

- .NETオブジェクトのメンバ(プロパティ、メソッドなど)を調べるには

  コマンドレットなどの後に**|Get-Member**

PSドライブ(1) 概要

- 従来のシェルはファイルシステムのドライブしか扱えなかったが、
- PowerShellでは、デフォルトで、ファイルシステム、レジストリ、デジタル署名、環境変数、エイリアス、スクリプト変数、関数を「PSドライブ」として扱うことができる。（Get-PSDrive）
- PSドライブを扱うための.NETプログラムが「PSプロバイダ」。
- コマンドレットとPSプロバイダを含む.NETアセンブリを「PSスナップイン」という。PSスナップインをインストールすることで機能拡張が可能。

PSドライブ(2) 項目の操作

作成	New-Item	ni
名前変更	Rename-Item	rni, ren
移動	Move-Item	mi, mv, move
コピー	Copy-Item	cpi, cp, copy
削除	Remove-Item	ri, rm, rmdir, del, erase, rd
実行	Invoke-Item	ii
プロパティ操作	Get-ItemProperty, Copy-, Clear-, Move-, Rename-, Remove-, Set-	

- これらの操作がどのドライブでも可能

DEMO1

従来のシェルにおけるパイプ

- カレントのファイルをファイル名で逆順ソート

  dir / bの出力＝テキストがパイプを通る

```
C:¥WINDOWS¥system32¥drivers¥etc>dir /b | sort /r
services
protocol
networks
lmhosts.sam
hosts
```

- 上の例は単純なテキストなのでソートできるが、ではサイズでソートするには？

  ？？？

オブジェクトが渡るパイプ(1) 概要

- ファイルサイズでソート、PowerShellなら可能です。
  Get-ChildItemの出力＝オブジェクトの配列がパイプを通る

```
PS C:¥WINDOWS¥system32¥drivers¥etc> Get-ChildItem | Sort-Object -property Length


    ディレクトリ: Microsoft.PowerShell.Core¥FileSystem::C:¥WINDOWS¥system32¥dri
    vers¥etc


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---          2004/08/05     21:00        407 networks
-a---          2004/08/05     21:00        734 hosts
-a---          2004/08/05     21:00        799 protocol
-a---          2004/08/05     21:00       3683 lmhosts.sam
-a---          2004/08/05     21:00       7116 services
```

FileInfoオブジェクトのLengthプロパティを元にソートされる。

オブジェクトが渡るパイプ(2) 通っているもの

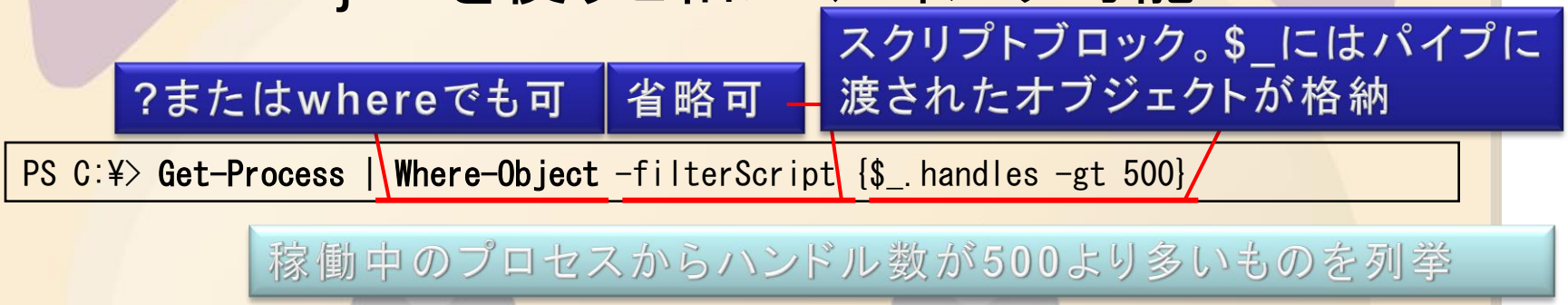
- コマンドレットの戻り値は.NETのオブジェクト

```
PS C:¥WINDOWS¥system32¥drivers¥etc> Get-ChildItem | Get-Member


   TypeName: System.IO.FileInfo

Name                      MemberType     Definition
----                      ----------     ----------
AppendText                Method         System.IO.StreamWriter AppendText()
CopyTo                    Method         System.IO.FileInfo CopyTo(String de...
Create                    Method         System.IO.FileStream Create()
CreateObjRef              Method         System.Runtime.Remoting.ObjRef Crea...
CreateText                Method         System.IO.StreamWriter CreateText()
・・・
LastAccessTime            Property       System.DateTime LastAccessTime {get...
LastAccessTimeUtc         Property       System.DateTime LastAccessTimeUtc {...
LastWriteTime             Property       System.DateTime LastWriteTime {get;...
LastWriteTimeUtc          Property       System.DateTime LastWriteTimeUtc {g...
Length                    Property       System.Int64 Length {get;}
Name                      Property       System.String Name {get;}
・・・
```

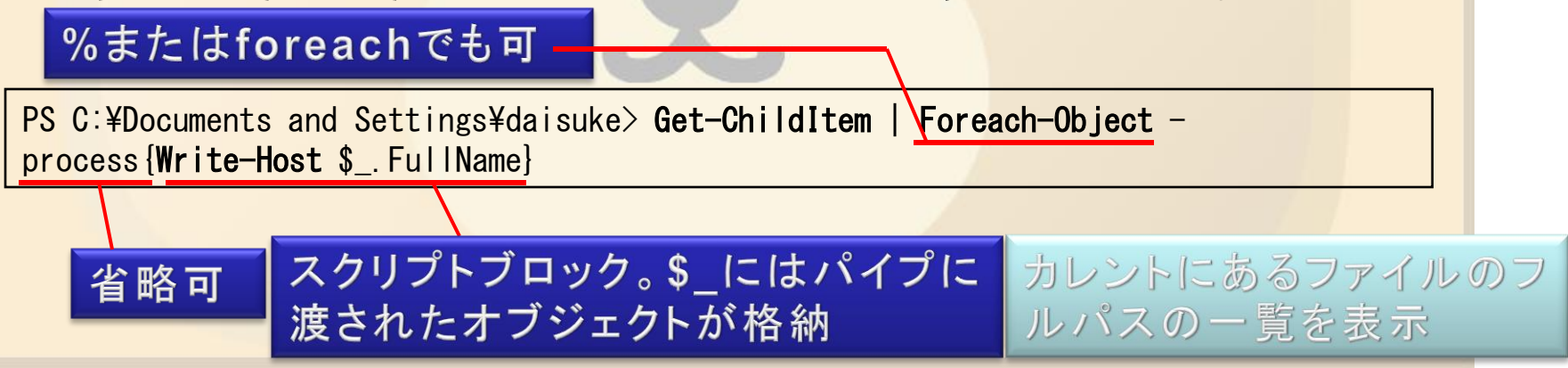
System.IO名前空間に属するFileInfoクラスのインスタンス(オブジェクト)

オブジェクトが渡るパイプ(3) フィルタと列挙

- Where-Objectを使うと細かくフィルタ可能



- Foreach-Objectでパイプを渡ったオブジェクト配列の要素それぞれに対してコマンド実行可能。



WMIも自由自在 Before & After

- WMI (Windows Management Instrumentation)のクラスのインスタンスを簡単に呼び出せる。

WSH with VBScriptでは…

```
Set wbemServices = GetObject("winmgmts:¥¥.")
Set wbemObjectSet = wbemServices.InstancesOf("Win32_PhysicalMemory")

For Each wbemObject In wbemObjectSet
    WScript.Echo "物理メモリ(bytes): " & wbemObject.Capacity
Next
```

長ったらしい…

PowerShellではこんなに簡単！

```
PS C:¥> Get-WMIObject -class Win32_PhysicalMemory -property capacity
```

省略すればさらに簡単！

```
PS C:¥> gwmi Win32_PhysicalMemory -p Capacity
```

パイプ＆WMI　デモ

DEMO2

おわりに

- 新しいオブジェクトベースのシェル、いかがでしたか？
- 今回はシェルとしてのPowerShellを見てきましたがスクリプティングもできます！
- クライアントで使うもよし、サーバー管理に使うもよし
- あなたなら何をしますか？　あるいはこういうの自動化できない？みたいなご意見募集！