



IUT d'Angers
License Sari
Module FTA3

Compte Rendu

« TP Powershell »

- Prise en main
- Pipeline
- Scripts

Par

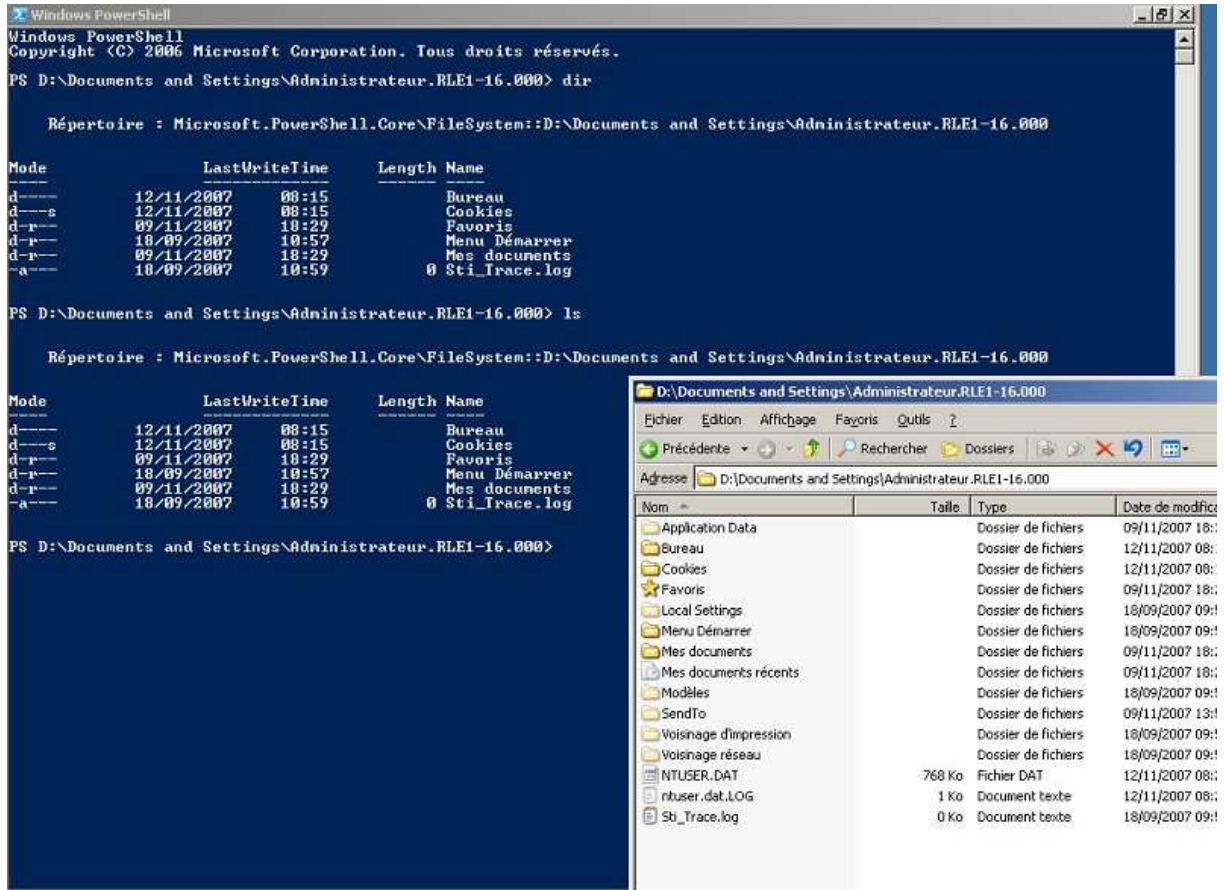
Sylvain Lecomte

Le 19/11/2007

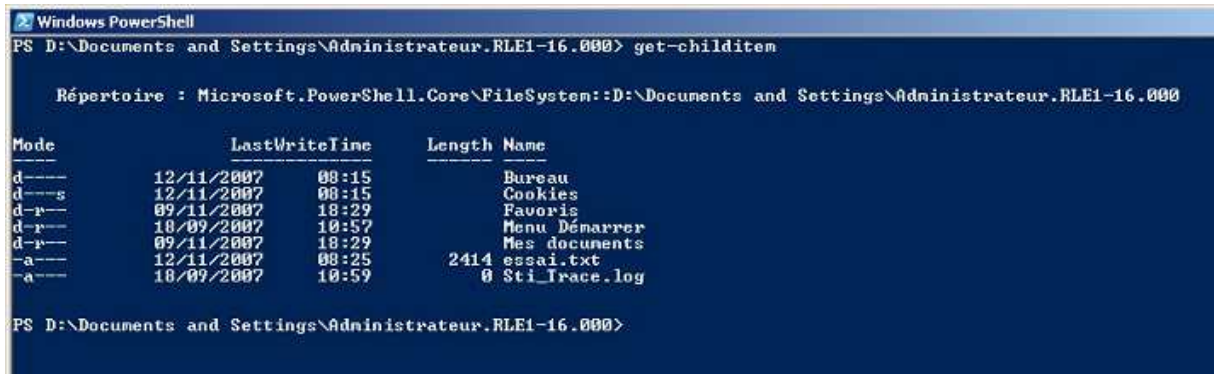
1. Introduction

Le PowerShell se rapproche plus du langage Perl que du Batch. Ce langage intègre beaucoup de fonctionnalité ainsi que différents niveaux de sécurité.

La commande « dir » ou « ls » sont des commandes permettant de lister des fichiers et dossier dans un répertoire.



Ces 2 commandes sont identiques car elles sont des alias de la commande « Get_Childitem ».



Pour obtenir la liste des alias, il est possible d'utiliser la commande « Get_Alias ».

```

Windows PowerShell
Windows PowerShell
Copyright (C) 2006 Microsoft Corporation. Tous droits réservés.

PS D:\Documents and Settings\Administrateur.RLE1-16.000> get-alias

CommandType      Name                Definition
-----
Alias             ac                  Add-Content
Alias             asnp               Add-PSSnapin
Alias             clc                Clear-Content
Alias             cli               Clear-Item
Alias             clp               Clear-ItemProperty
Alias             clv               Clear-Variable
Alias             cpi               Copy-Item
Alias             cpp               Copy-ItemProperty
Alias             cupsa            Convert-Path
Alias             diff             Compare-Object
Alias             epal             Export-Alias
Alias             epsv             Export-Csv
Alias             fc               Format-Custom
Alias             fl               Format-List
Alias             foreach          ForEach-Object
Alias             %                ForEach-Object
Alias             ft               Format-Table
Alias             fw               Format-Wide
Alias             gal              Get-Alias

```

La commande « Get_process », donne la liste des processus en cours.

```

PS D:\dossier_essai> Get-Process

Handles  NPM(K)  PM(K)  WS(K)  UM(M)  CPU(s)  Id  ProcessName
-----
112      3       896    2732   22      0,11    1292  appmgr
581      6       2500   2596   26      45,58    308   csrss
79       3       392    1340   17      0,14    3648  ctfmon
120      6       1648   2704   35      0,08    1376  dfssvc
167      16      6832   2772   50      0,13    1400  dns
78       2       528    1468   19      0,02    1452  elementmgr
444      14      10760  8828   68      17,00    3552  explorer
65       2       1832   1900   30      2,80    856   HoverSnap
0        0       0       16     0       0        0     Idle
354      11      9972   1860   86      4,34    3960  iexplore
471      31      9260   8912   89      0,70    1560  inetinfo

```

La commande « Get_command » donne la liste des commandes.

Une commande très pratique est « Get_help », elle permet d'obtenir de l'aide sur une commande. Exemple « Get_help restart_service » donne l'aide de restart service :

```

Windows PowerShell
PS D:\Documents and Settings\Administrateur.RLE1-16.000> get-help restart-service
NOM
Restart-Service
RÉSUMÉ
Arrête, puis démarre un ou plusieurs services.
SYNTAXE
Restart-Service [-name] <string[]> [-force] [-include <string[]>] [-exclude <string[]>] [-passthru] [-whatif] [-confirm] [<CommonParameters>]
Restart-Service -displayName <string[]> [-force] [-include <string[]>] [-exclude <string[]>] [-passthru] [-whatif] [-confirm] [<CommonParameters>]
Restart-Service [-inputObject <ServiceController[]>] [-force] [-include <string[]>] [-exclude <string[]>] [-passthru] [-whatif] [-confirm] [<CommonParameters>]
DESCRIPTION DÉTAILLÉE
Pour chaque service que vous redémarrez, l'applet de commande Restart-Service envoie un message d'arrêt, puis un message de démarrage au Contrôleur de services Windows. Si un service est déjà arrêté, il est démarré sans notification d'erreur. Vous pouvez spécifier les services à l'aide de leurs noms de service ou de leurs noms d'affichage. Il est également possible d'utiliser le paramètre InputObject pour passer un objet représentant chaque service à redémarrer.
LIENS CONNEXES
Start-Service
Stop-Service
Suspend-Service
Resume-Service
New-Service
Get-Service
Set-Service
REMARQUES
Pour plus d'informations, tapez : "get-help Restart-Service -detailed".
Pour obtenir des informations techniques, tapez : "get-help Restart-Service -full".
PS D:\Documents and Settings\Administrateur.RLE1-16.000>

```

2. Pipeline

« Pipe-liner » consiste utiliser le résultat d'une commande en entrée d'une autre. Exemple « Get-Childitem | more » renvoie à more le résultat de Get-Childitem.

EXERCICE 1 :

Cet exercice est le premier exercice de mise en application des pipelines. Tout d'abord plusieurs fichiers sont créés dans un répertoire essai. Voici ensuite les commandes exécutées.

```

Windows PowerShell
PS D:\Documents and Settings\Administrateur.RLE1-16.000> cd ..
PS D:\Documents and Settings> cd ..
PS D:\> cd dossier_essai
PS D:\dossier_essai> $variable
PS D:\dossier_essai> $variable = Get-ChildItem
PS D:\dossier_essai> $variable

Répertoire : Microsoft.PowerShell.Core\FileSystem::D:\dossier_essai

Mode                LastWriteTime         Length Name
----                -
d-----          12/11/2007   08:42           Sous rep1
d-----          12/11/2007   08:43           Sous rep2
-a-----          12/11/2007   08:41           0 Copie (10) de Copie de Nouveau Document texte.txt
-a-----          12/11/2007   08:41           0 Copie (10) de Nouveau Document texte.txt
-a-----          12/11/2007   08:41           0 Copie (11) de Nouveau Document texte.txt
-a-----          12/11/2007   08:41           0 Copie (12) de Nouveau Document texte.txt
-a-----          12/11/2007   08:41           0 Copie (13) de Nouveau Document texte.txt
-a-----          12/11/2007   08:41           0 Copie (14) de Nouveau Document texte.txt

```

La première ligne déclare la variable appelé ici « Variable ».

La deuxième ligne renvoie le résultat de Get_Childitem dans la variable sous forme de tableau.

La troisième ligne lit le contenu du tableau variable.

```
PS D:\dossier_essai> $variable[1]

Répertoire : Microsoft.PowerShell.Core\FileSystem::D:\dossier_essai

Mode                LastWriteTime         Length Name
----                -
d-----          12/11/2007         08:43             Sous rep2

PS D:\dossier_essai>
```

Ceci permet de lire la première ligne du tableau variable.

```
PS D:\dossier_essai> $variable[1] | Get-Member

TypeName: System.IO.DirectoryInfo

Name                MemberType          Definition
----                -
Create              Method              System.Void Create(), System.Void C
CreateObjRef        Method              System.Runtime.Remoting.ObjRef Crea
CreateSubdirectory  Method              System.IO.DirectoryInfo CreateSubdir
Delete              Method              System.Void Delete(), System.Void D
Equals              Method              System.Boolean Equals(Object obj)
GetAccessControl    Method              System.Security.AccessControl.Direc
GetDirectories      Method              System.IO.DirectoryInfo[] GetDirect
GetFiles            Method              System.IO.FileInfo[] GetFiles(Strin
GetFileSystemInfos  Method              System.IO.FileSystemInfo[] GetFileS
GetHashCode         Method              System.Int32 GetHashCode()
GetLifetimeService Method              System.Object GetLifetimeService()
```

Le résultat est maintenant renvoyé à Get_member. On obtient la liste des propriétés et méthode applicables à l'objet \$variable1 (liste écourtée dans cette illustration ».

On essaye maintenant la propriété root qui figure dans la liste :

```
PS D:\dossier_essai> $variable[1].root

Mode                LastWriteTime         Length Name
----                -
d--hs             12/11/2007         08:41             D:\
```

Le résultat est le mode d'accès autorisé, la date de dernière écriture ainsi que le répertoire.

EXERCICE 2 :

L'exercice 2 a pour but de lister les fichiers d'un répertoire qui ont pour extension Txt via la commande Where-Object.

```
PS D:\dossier_essai> Get-Childitem | Where-Object { $_.get_extension() -eq ".txt" }

Répertoire : Microsoft.PowerShell.Core\FileSystem::D:\dossier_essai

Mode                LastWriteTime         Length Name
----                -
a-----          12/11/2007         08:41             0 Copie (10) de Nouveau Document texte.txt
a-----          12/11/2007         08:41             0 Copie (11) de Nouveau Document texte.txt
a-----          12/11/2007         08:41             0 Copie (12) de Nouveau Document texte.txt
a-----          12/11/2007         08:41             0 Copie (13) de Nouveau Document texte.txt
a-----          12/11/2007         08:41             0 Copie (14) de Nouveau Document texte.txt
a-----          12/11/2007         08:41             0 Copie (15) de Nouveau Document texte.txt
a-----          12/11/2007         08:41             0 Copie (16) de Nouveau Document texte.txt
```

On utilise ici un pipeline entre Get_childitem et Where-Object. La propriété de Where-Object utilisée est get_extension et associé à Txt.

Une autre solution, utilisé la propriété include de get_childitem.

```
PS D:\dossier_essai> get-childitem * -Include *.txt
```

EXERCICE 3:

La ligne de commande suivante liste les fichiers d'un répertoire dans l'ordre de la dernière modification

```
PS D:\dossier_essai> Get-Childitem | Sort-Object -property lastwritetime -descending | more
```

Répertoire : Microsoft.PowerShell.Core\FileSystem::D:\dossier_essai

Mode	LastWriteTime	Length	Name
d----	12/11/2007 08:43		Sous rep2
d----	12/11/2007 08:42		Sous rep1
-a---	12/11/2007 08:41	0	Copie (60) de Nouveau Document texte.txt
-a---	12/11/2007 08:41	0	Copie (61) de Nouveau Document texte.txt
-a---	12/11/2007 08:41	0	Copie (62) de Nouveau Document texte.txt
-a---	12/11/2007 08:41	0	Copie (6) de Copie de Copie de Nouveau Document texte.txt
-a---	12/11/2007 08:41	0	Copie (6) de Copie de Nouveau Document texte.txt
-a---	12/11/2007 08:41	0	Copie (6) de Nouveau Document texte.txt

La ligne de commande suivante tu tous les processus qui s'appellent Notepad. Elle s'exécute en plusieurs étapes. Création d'un tableau \$a contenant les processus Notepad.

```
PS D:\dossier_essai> $a = Get-Process notepad
PS D:\dossier_essai> $a
```

Handles	NPM(K)	PM(K)	WS(K)	UM(M)	CPU(s)	Id	ProcessName
54	2	800	2592	27	0.06	2188	notepad

Puis utilisation de la propriété kill du tableau.

```
PS D:\dossier_essai> $a.kill()
```

3. Scripts

EXERCICE 4:

Il y a différents niveau d'autorisation pour l'exécution de scripts, la commande suivante donne le niveau actuel d'autorisation.

```
PS D:\dossier_essai> Get-ExecutionPolicy
RemoteSigned
```

Le niveau actuel est « RemoteSigned », les scripts exécutés localement ne nécessite pas de signature, les scripts téléchargés doivent être signés.

Changement de la stratégie en « AllSigned », maintenant tous les scripts doivent être signés :

```
PS D:\dossier_essai> Set-ExecutionPolicy allsigned
PS D:\dossier_essai> Get-ExecutionPolicy
AllSigned
```

EXERCICE 5:

Dans cet exercice, il faut écrire un script qui à partir d'un fichier texte qui contient des noms d'utilisateurs, crée un répertoire pour chaque utilisateur. En voici la solution

```
PS D:\dossier_essai> get-content ex5.txt
Roberta
Monika
PS D:\dossier_essai> $test=get-content ex5.txt
PS D:\dossier_essai> $test
Roberta
Monika
PS D:\dossier_essai> new-item $test -type directory

Répertoire : Microsoft.PowerShell.Core\FileSystem::D:\dossier_essai
```

Mode	LastWriteTime	Length	Name
d----	12/11/2007 10:32		Roberta
d----	12/11/2007 10:32		Monika

EXERCICE 6:

Dans cet exercice il faut développer un script qui à chaque utilisateurs fait correspondre un répertoire personnel, précédemment créé. Ceci se fait via ADSI.

Nous n'avons pas réussi à mettre en place la totalité du cahier des charges puisque nous n'avons modifié le répertoire personnel que pour un seul utilisateur.

```
PS D:\dossier_essai> $objUser=[ADSI]"LDAP://localhost/cn=Roberta,cn=Users,dc=domeq1-4,dc=fr"
PS D:\dossier_essai> $objUser.homeDirectory = "\\RLE1-16\dossier_essai\Roberta"
PS D:\dossier_essai> $objUser.homeDirectory
\\RLE1-16\dossier_essai\Roberta
```

L'interface graphique UOAD nous confirme la bonne mise en place de l'exercice précédent :

