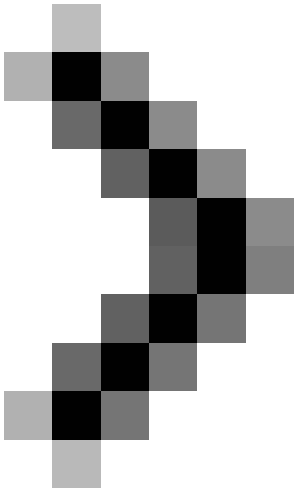
Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 3

Part I and Part II of this series showed PowerShell and SMO settings, simple WMI cmdlets. This part 3 will cover how to write code for the PowerShell cmdlet and execute them. Script code is essential for automated operations and



Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 1

Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 2

The MAK

Part I and Part II of this series showed PowerShell and SMO settings, simple WMI cmdlets. This part 3 will cover how to write code for the PowerShell cmdlet and execute them. Script code is essential for automatic and repetitive operations.

Enforcement policy

The four different types of Windows PowerShell enforcement policies are Restricted, AllSigned, RemoteSigned and Unrestricted. We will find the Windows PowerShell execution policy on the workspace. [Figure 1.0]

Cmdlet:
Get-executionpolicy
Result:
Restricted

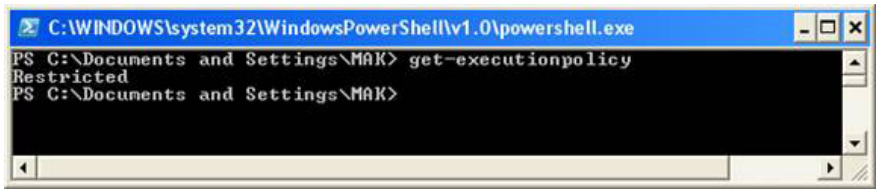


Figure 1.0
Suppose we have the following line of code on PowerShell script 'a.ps1'. [Figure 1.1]

Echo 'test'

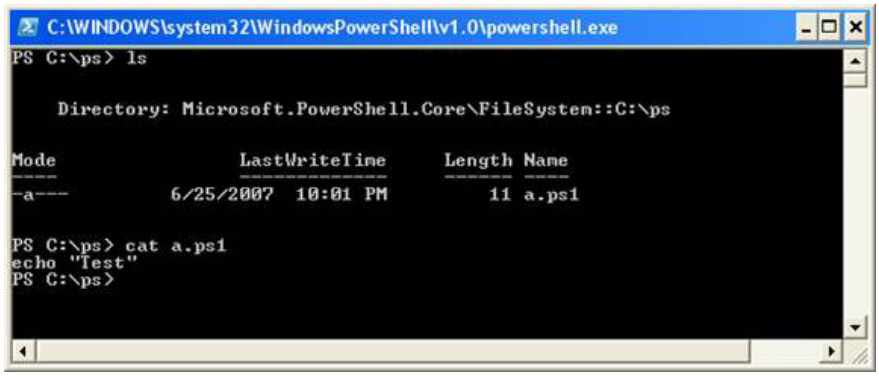


Figure 1.1

If you try to execute the code when PowerShell's execution policy is restricted, the following error message will appear. [Figure 1.2]

The command to execute PowerShell code

. /a.ps1

Result

File C: psa.ps1 cannot be loaded because the execution of scripts is disabled on this system. Please see "get-help about_signing" for more details.
At line: 1 char: 3
+ ./a

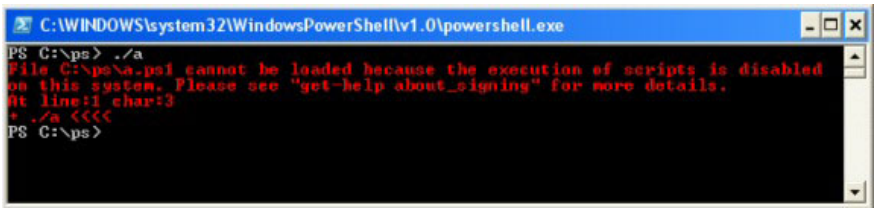


Figure 1.2

Please change the execution policy to unrestricted. The command to execute PowerShell code can already be executed by the following cmdlet. [Figure 1.3]

Figure 1.3

Now try executing the a.ps1 script as shown in the image below. [Figure 1.4]

Order

./a

Result

Ki?m TRA

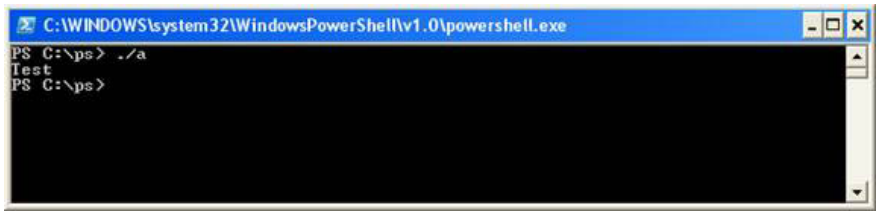


Figure 1.4

Control Input in PowerShell code

When performing repetitive operations, we like applications to interact more and build applications to require user input. We can do the same with PowerShell.
Create a PowerShell code that accepts the name of the SQL Server field and the database name. Also, let PowerShell display all tables on that database. This can be done with the read-host cmdlet.

Example 1: [Figure 1.5]

Read-host 'Please Enter Second Number'.

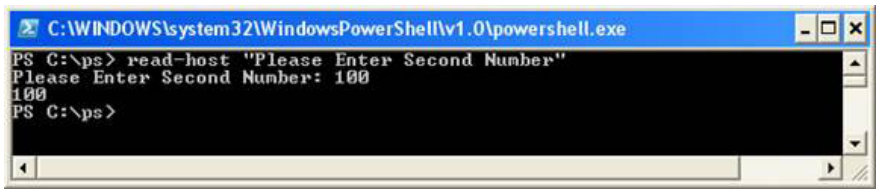


Figure 1.5
Example 2

We can assign the cmdlet value to be a variable. [Figure 1.6]

$ a = read-host "Please Enter Second Number" $ a

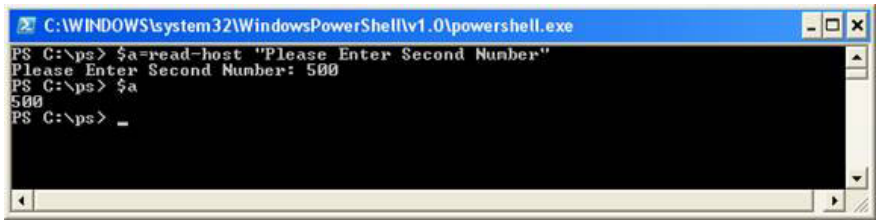


Figure 1.6

Combine example 1 and example 2 connect to SQL Server.

Generating PowerShell code is called connectsql.ps1. [Figure 1.7]

$ SQLSERVER = read-host "Enter SQL Server Name:"
$ Database = read-host "Enter Database Name:"
$ SqlConnection = New-Object System.Data.SqlClient.SqlConnection
$ SqlConnection.ConnectionString = "Server = $ SQLSERVER; Database = $ DATABASE; Integrated Security = True"
$ SqlCmd = New-Object System.Data.SqlClient.SqlCommand
$ SqlCmd.CommandText = "select name from sysobjects where type = 'u'"
$ SqlCmd.Connection = $ SqlConnection
$ SqlAdapter = New-Object System.Data.SqlClient.SqlDataAdapter
$ SqlAdapter.SelectCommand = $ SqlCmd
$ DataSet = New-Object System.Data.DataSet
$ SqlAdapter.Fill ($ DataSet)
$ SqlConnection.Close ()
$ DataSet.Tables [0]

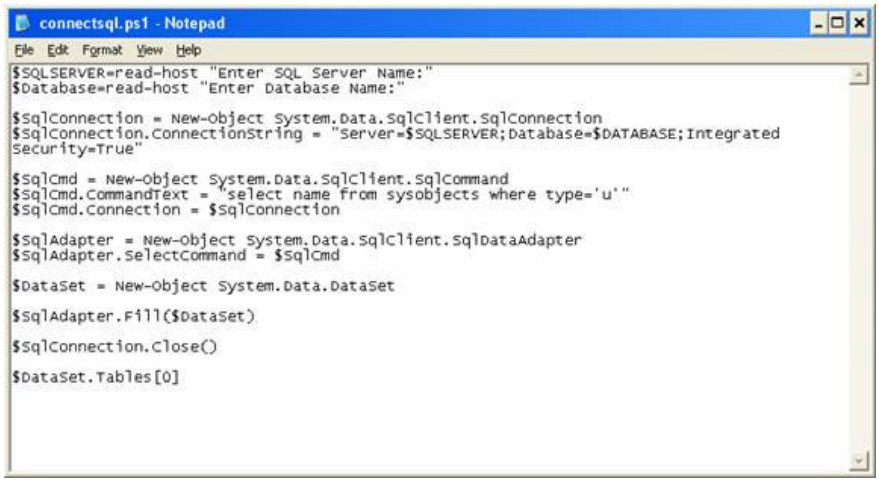


Figure 1.7

Now let's execute the above connectsql.ps1 code above. [Figure 1.8]

./connectsql
Enter SQL Server Name :: HOMESQLEXPRESS
Enter Database Name :: AdventureWorks

Note : HOME is the server and SQLEXPRESS is the instance name of SQL Server. Replace this name with your server name and SQL Server. AdventureWorks is the database name. You also need to replace this database name for the database name on the server.

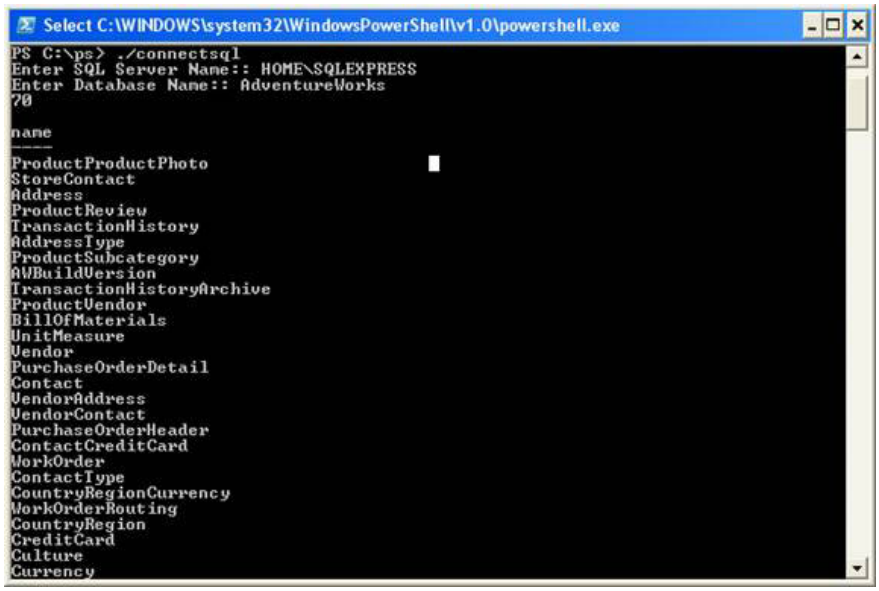


Figure 1.8

The connectsql code assigns the value entered into the $ SQLSERVER and $ DATABASE variables, connects the string using those variables and displays the result.

Result:

name

ProductProductPhoto
StoreContact
Address
ProductReview
TransactionHistory
AddressType
ProductSubcategory
AWBuildVersion
TransactionHistoryArchive
ProductVendor
BillOfMaterials
UnitMeasure
Vendor
PurchaseOrderDetail
Contact
VendorAddress
VendorContact
PurchaseOrderHeader
ContactCreditCard
WorkOrder
ContactType
CountryRegionCurrency
WorkOrderRouting
CountryRegion
CreditCard
Culture
Currency
SalesOrderDetail
CurrencyRate
Customer
SalesOrderHeader
CustomerAddress
Department
Document
Employee
SalesOrderHeaderSalesReason
SalesPerson
EmployeeAddress
EmployeeDepartmentHistory
EmployeePayHistory

SalesPersonQuotaHistory
Illustration
SalesReason
Individual
SalesTaxRate
JobCandidate
Location
SalesTerritory
Product
SalesTerritoryHistory
ScrapReason
Shift
ProductCategory
ShipMethod
ProductCostHistory
ProductDescription
ShoppingCartItem
ProductDocument
ProductInventory
SpecialOffer
ProductListPriceHistory
SpecialOfferProduct
ProductModel
StateProvince
ProductModelIllustration
DatabaseLog
ProductModelProductDescriptionCulture
ErrorLog
Store
ProductPhoto

However, when writing code automatically, you don't want users to enter data. Instead, we should accept the parameters.

Please upgrade the code above by accepting the parameters. [Figure 1.9]

```
param ([string] $ SQLSERVER, [string] $ Database) $ SqlConnection = New-Obje
??
SqlAdapter.Fill ($ DataSet) $ SqlConnection.Close () $ DataSet.Tables [0]
```

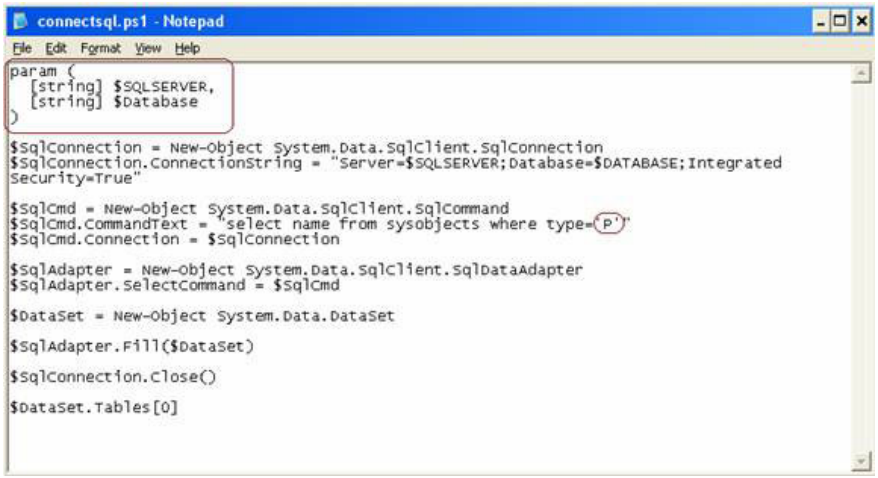


Figure 1.9

Perform the code as shown below. [Figure 2.0]

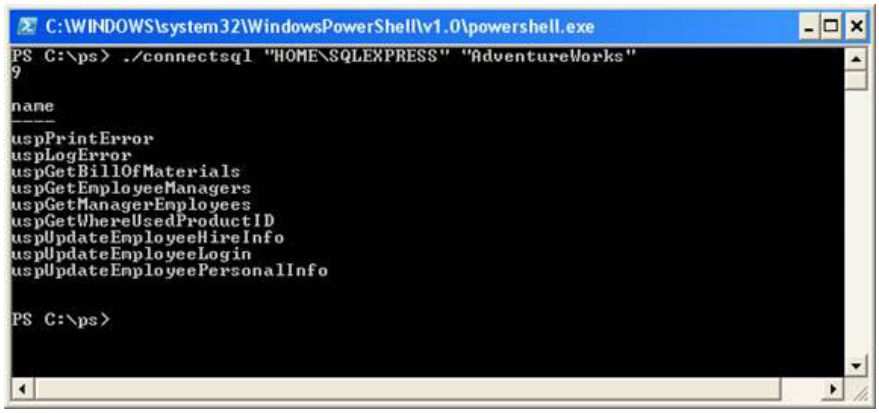


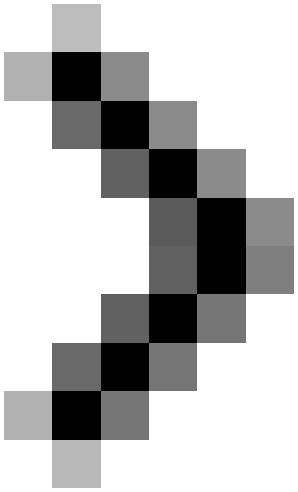
Figure 2.0

The connectsql assigns values ??as a parameter to the $ SQLSERVER and $ DATABASE separate variables, connects strings using those variables and displays the results.

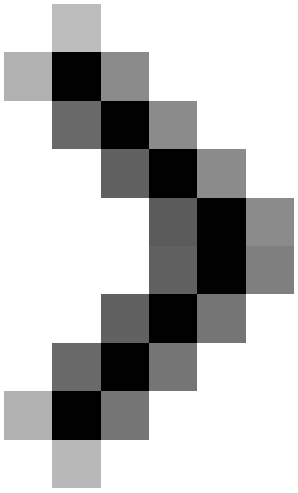
Result

name
----
uspPrintError
uspLogError
uspGetBillOfMaterials
uspGetEmployeeManagers
uspGetManagerEmployees
uspGetWhereUsedProductID
uspUpdateEmployeeHireInfo

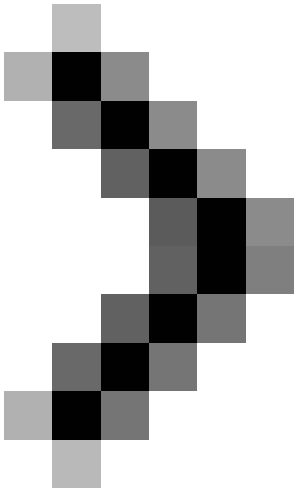
uspUpdateEmployeeLogin
uspUpdateEmployeePersonalInfo



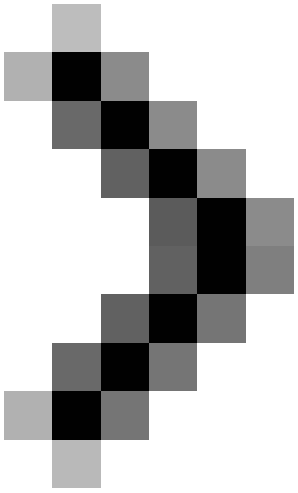
Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 4



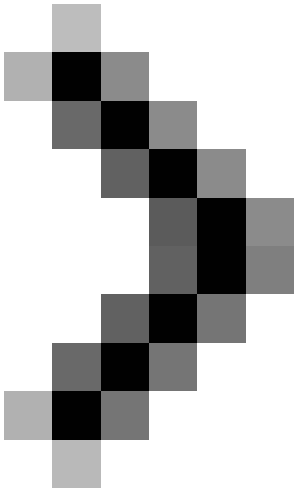
Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 5



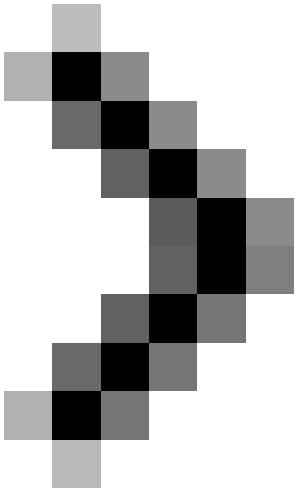
Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 6



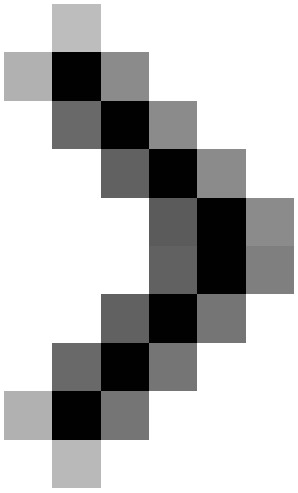
Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 7



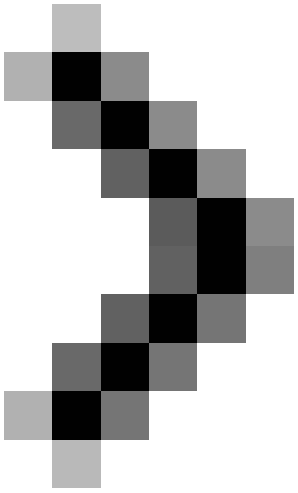
Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 8



Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 9



Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 10



Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 11

You finished reading the article "**Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 3**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
