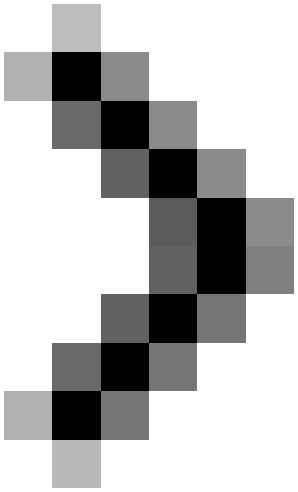
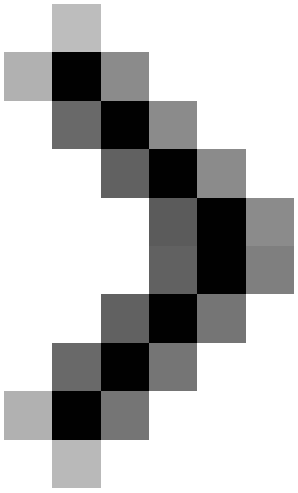
Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 7

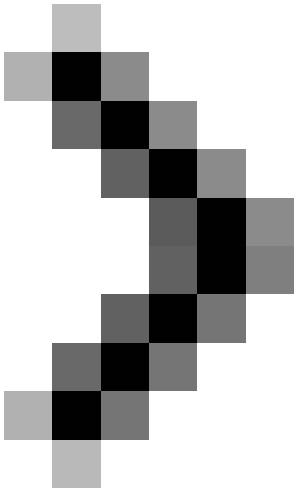
Each part of this series demonstrates how to use PowerShell in conjunction with SMO to present SQL Server objects. Part 7 of this series will demonstrate how to use PowerShell and SMO to find all available objects in a database on the server.



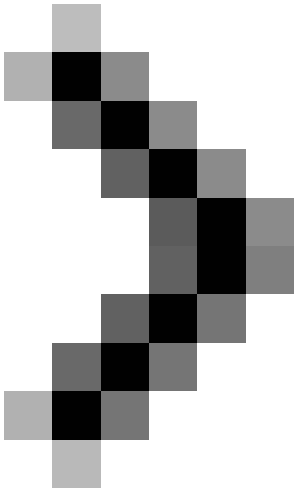
Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 1



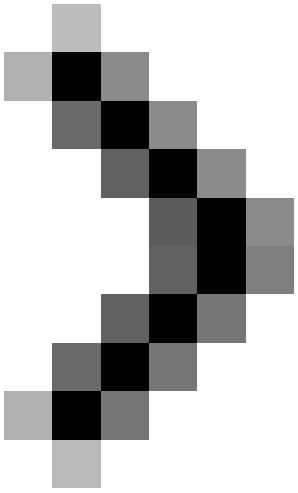
Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 2



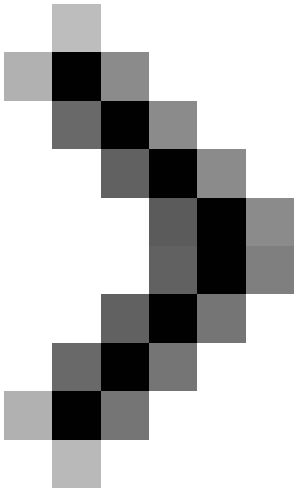
Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 3



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

The MAK

Part 1 and Part 2 of this series introduced PowerShell and simple SMO, WMI cmdlet installation. In Part 3, I will show you how to create a PowerShell script to connect to a SQL Server.

Part 4 introduces how to use a PowerShell script to perform a file loop and connect other servers. Part 5 shows you how to create a SQL Server database using PowerShell and SMO. Part 6 introduces backing up SQL Server databases with PowerShell and SMO.

Each part of this series demonstrates how to use PowerShell in conjunction with SMO to present SQL Server objects.

Method 1: Display the table names

Let's assume we want to find all the tables already in the ' *AdventureWorks* ' database, on the server ' *HOMESQLEXPRESS* '. Execute the following command, refer to Figure 1.1.

```
[reflection.assembly] :: LoadWithPartialName ("Microsoft.SqlServer.Smo") | out-null
$ srv = New-Object "Microsoft.SqlServer.Management.Smo.Server" "HOMESQLEXPRESS"
$ db = $ srv.Databases ["adventureworks"]
foreach ($ tbl in $ db.tables) {$ tbl.name}
```

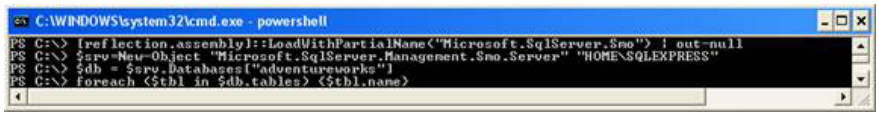


Figure 1.1

The above Cmdlets will display the table names in the AdventureWorks database on the server '*HOMESQLEXPRESS*' (see Figure 1.2).

Result

AWBuildVersion
DatabaseLog
ErrorLog
Department
Employee
EmployeeAddress
EmployeeDepartmentHistory
EmployeePayHistory
JobCandidate
Shift
Address
AddressType
Contact
ContactType
CountryRegion
StateProvince
BillOfMaterials
Culture
Document
Illustration
Location
Product
ProductCategory
ProductCostHistory
ProductDescription
ProductDocument
ProductInventory
ProductListPriceHistory
ProductModel
ProductModelIllustration
ProductModelProductDescriptionCulture
ProductPhoto
ProductProductPhoto
ProductReview
ProductSubcategory

ScrapReason
TransactionHistory
TransactionHistoryArchive
UnitMeasure
WorkOrder
WorkOrderRouting
ProductVendor
PurchaseOrderDetail
PurchaseOrderHeader
ShipMethod
Vendor
VendorAddress
VendorContact
ContactCreditCard
CountryRegionCurrency
CreditCard
Currency
CurrencyRate
Customer
CustomerAddress
Individual
SalesOrderDetail
SalesOrderHeader
SalesOrderHeaderSalesReason
SalesPerson
SalesPersonQuotaHistory
SalesReason
SalesTaxRate
SalesTerritory
SalesTerritoryHistory
ShoppingCartItem
SpecialOffer
SpecialOfferProduct
Store
StoreContact

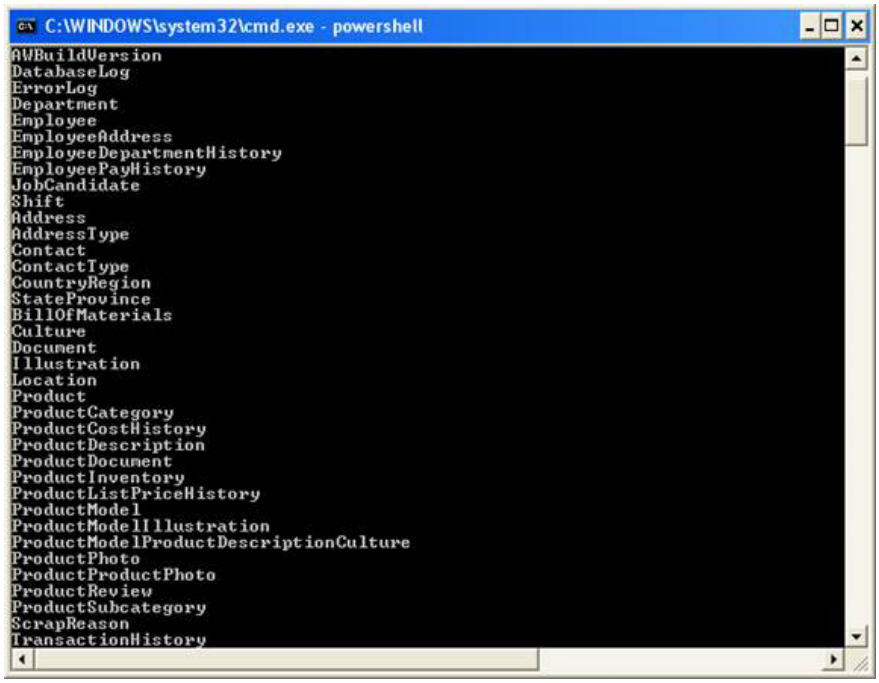


Figure 1.2

Method 2

Suppose that you want to find all the objects that are already in the ' *AdventureWorks* ' database, on the server ' *HOMESQLEXPRESS* '. Execute the following command, refer to Figure 1.3.

```
[reflection.assembly] :: LoadWithPartialName ("Microsoft.SqlServer.Smo") | out-null
$ srv = New-Object "Microsoft.SqlServer.Management.Smo.Server" "HOMESQLEXPRESS"
$ db = $ srv.Databases ["adventureworks"]
echo "Tables"
echo "------"
foreach ($ tbl in $ db.Tables) {$ tbl.name}
echo "Synonyms"
echo "------"
foreach ($ Synonyms in $ db.Synonyms) {$ Synonyms.name}
"Stored Procedures" echo
echo "------"
foreach ($ StoredProcedures in $ db.StoredProcedures) {$ StoredProcedures.name}
echo "Assemblies"
echo "------"
foreach ($ Assemblies in $ db.Assemblies) {$ Assemblies.name}
echo "User Defined Functions"
echo "------"
foreach ($ UserDefinedFunctions in $ db.UserDefinedFunctions) {$ UserDefinedFunctions.name}
echo "Views"
echo "------"
foreach ($ Views in $ db.Views) {$ Views.name}
```

echo "ExtendedStoredProcedures"
echo "------"
foreach ($ ExtendedStoredProcedures in $ db.ExtendedStoredProcedures) {$ ExtendedStoredProcedures.name}

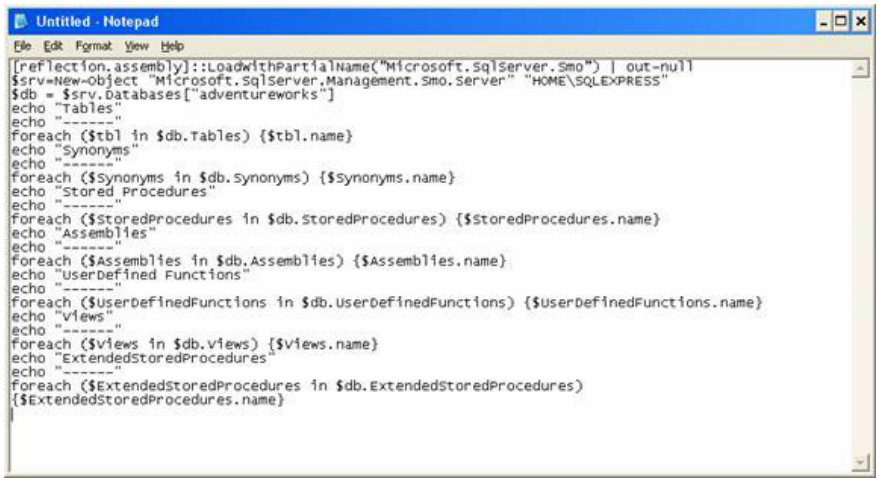


Figure 1.3

The above Cmdlets will display the object names in the AdventureWorks database on the server '*HOMESQLEXPRESS*' (see Figure 1.4).

Result

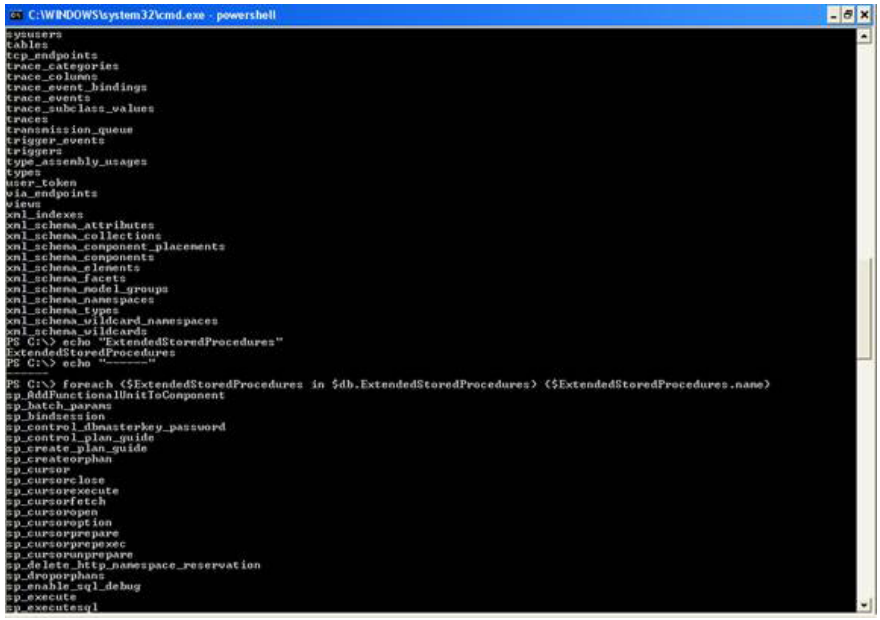


Figure 1.4

Method 3

Please connect method 1 and method 2 to a PowerShell script form to accept the following parameters. Create the ***listobjects.ps1*** file as shown below.

```
param
(
[string] $ ServerName,
[string] $ DatabaseName,
[string] $ ObjectType
)

[reflection.assembly] :: LoadWithPartialName ("Microsoft.SqlServer.Smo") | out-null
$ srv = New-Object "Microsoft.SqlServer.Management.Smo.Server" "$ ServerName"
$ db = $ srv.Databases ["$ DatabaseName"]

if ($ ObjectType -eq "TABLES")
{
echo "Tables"
echo "------"
foreach ($ tbl in $ db.Tables) {$ tbl.name}
}

if ($ ObjectType -eq "SYNONYMS")
{
echo "Synonyms"
echo "--------"
foreach ($ Synonyms in $ db.Synonyms) {$ Synonyms.name}
}

if ($ ObjectType -eq "SP")
{
"Stored Procedures" echo
echo "------------------"
foreach ($ StoredProcedures in $ db.StoredProcedures) {$ StoredProcedures.name}
}

if ($ ObjectType -eq "ASM")
{
echo "Assemblies"
echo "----------"
foreach ($ Assemblies in $ db.Assemblies) {$ Assemblies.name}
}

if ($ ObjectType -eq "UDF")
{
echo "User Defined Functions"
```

```
echo "----------------------"
foreach ($ UserDefinedFunctions in $ db.UserDefinedFunctions)
{$ UserDefinedFunctions.name}
}

if ($ ObjectType -eq "VIEWS")
foreach ($ Views in $ db.Views) {$ Views.name}
}

if ($ ObjectType -eq "XP")
{
echo "ExtendedStoredProcedures"
echo "------------------------"
foreach ($ ExtendedStoredProcedures in $ db.ExtendedStoredProcedures)
{$ ExtendedStoredProcedures.name}
}
```

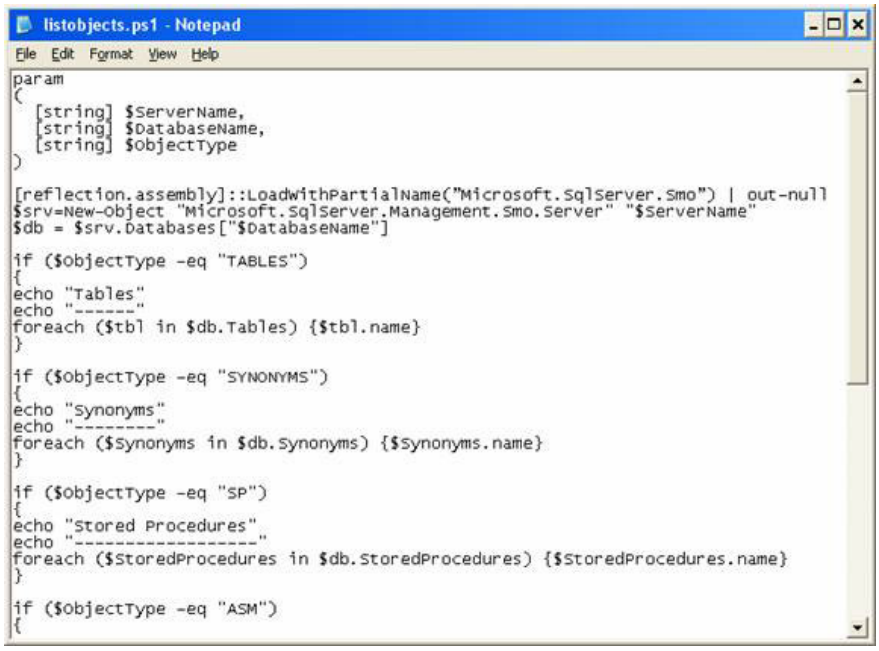


Figure 1.5

Now execute the ***listobjects.ps1*** file as shown below (see Figure 1.6).

```
./listobjects "HOMESQLEXPRESS" "AdventureWorks" "UDF"
```

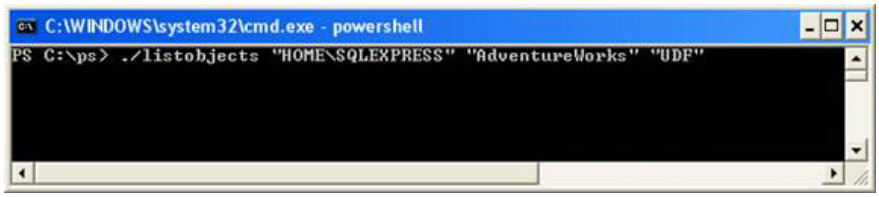


Figure 1.6

Explain the parameters

listobjects is the script file listobjects.ps1 in the c: ps directory.

HOME is the configuration name

SQLEXPRESS is the SQL server name on the master configuration named HOME

AdventureWorks is the database name residing in SQLEXPRESS.

UDF is a parameter, which is used to display all user defined functions in the AdventureWorks database.

Valid parameters for object types are

UDF - User Defined Functions

TABLES - Tables

ASM - Assemblies

SP - Stored Procedures

XP - Extended Stored Procedures

VIEWS - views

SYNONYMS - synonyms

The PowerShell script above shows the names of the objects of a particular database in the server. (see Figure 1.7)

Result

UserDefined Functions
----
ufnGetAccountingEndDate
ufnGetAccountingStartDate

ufnGetContactInformation
ufnGetDocumentStatusText
ufnGetProductDealerPrice
ufnGetProductListPrice
ufnGetProductStandardCost
ufnGetPurchaseOrderStatusText
ufnGetSalesOrderStatusText
ufnGetStock
ufnLeadingZeros
dm_db_index_operational_stats
dm_db_index_physical_stats
dm_db_missing_index_columns
dm_exec_cached_plan_dependent_objects
dm_exec_cursors
dm_exec_plan_attributes
dm_exec_query_plan
dm_exec_sql_text
dm_exec_xml_handles
dm_io_virtual_file_stats
fn_builtin_permissions
fn_cColvEntries_80
fn_check_object_signatures
fn_dblog
fn_dump_dblog
fn_EnumCurrentPrincipals
fn_fIsColTracked
fn_get_sql
fn_GetCurrentPrincipal
fn_GetRowsetIdFromRowDump
fn_helpcollations
fn_helpdatatypemap
fn_IsBitSetInBitmask
fn_isrolemember
fn_listextendedproperty
fn_MapSchemaType
fn_MSdayasnumber
fn_MSgeneration_downloadonly
fn_MSget_dynamic_filter_login
fn_MSorbitmaps
fn_MSrepl_map_resolver_clsid
fn_MStestbit
fn_MSvector_downloadonly
fn_my_permissions
fn_numberOf1InBinaryAfterLoc
fn_numberOf1InVarBinary
fn_repladjustcolumnmap
fn_repldecryptver4
fn_replformatdatetime

fn_replgetcolidfrombitmap
fn_replgetparsedddlcmd
fn_replreplacesinglequote
fn_replreplacesinglequoteplusprotectstring
fn_repluniquename
fn_RowDumpCracker
fn_servershareddrives
fn_sqlvarbasetostr
fn_trace_geteventinfo
fn_trace_getfilterinfo
fn_trace_getinfo
fn_trace_gettable
fn_translate_permissions
fn_varbintohexstr
fn_varbintohexsubstring
fn_virtualfilestats
fn_virtualservernodes
fn_yukonsecuritymodelrequired

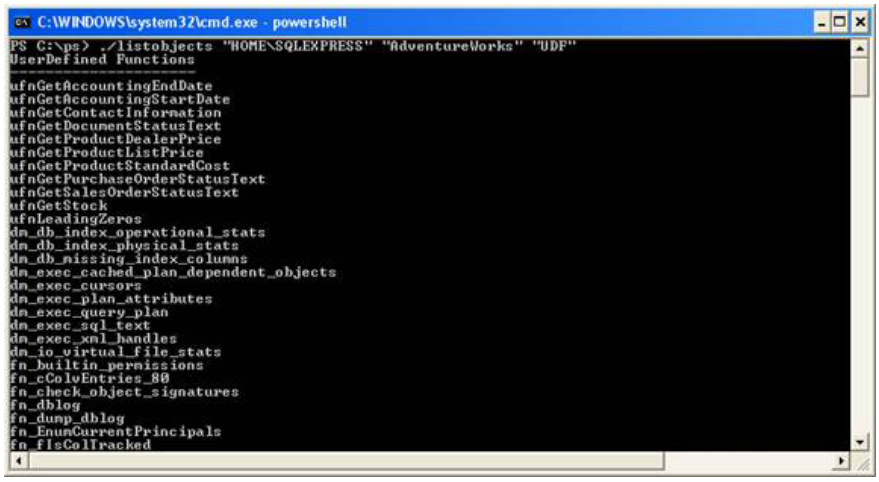
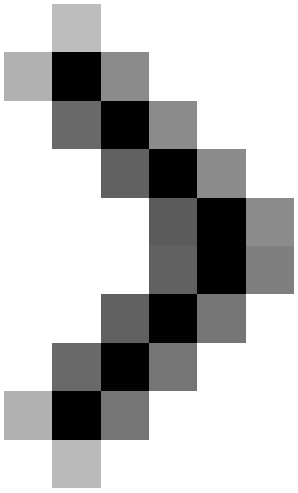


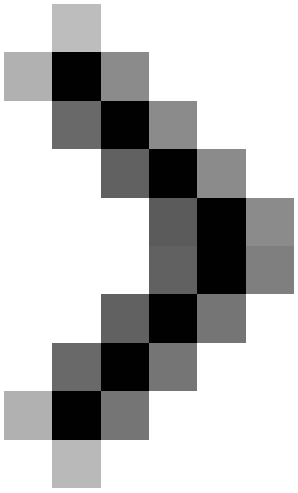
Figure 1.7

Conclude

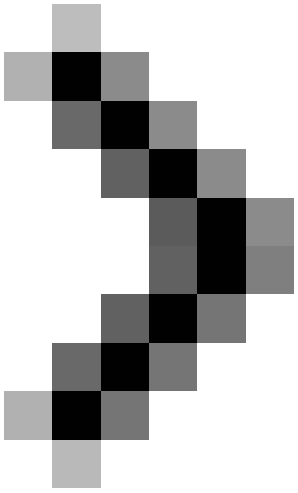
Part 7 of this series illustrated how to use PowerShell and SMO to find all available objects in a database on the server.



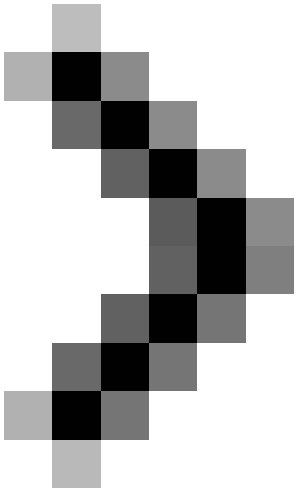
Microsoft Windows PowerShell and SQL Server 2005 SMO - Part 8



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 7**" 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.
