**Automatically Build a SQL Server Script Database Library from Your Script Files**

|  |  |
| --- | --- |
|  | By: [Jeffrey Yao](https://www.mssqltips.com/sqlserverauthor/121/jeffrey-yao/)  FROM: <https://www.mssqltips.com/sqlservertip/4102/automatically-build-a-sql-server-script-database-library-from-your-script-files/> |

**Problem**

As a contract consultant, I frequently need to create scripts for various tasks, so over the years with different clients, I have accumulated lots of scripts and they are often located in complex folders on different computers, is there any good way to archive and manage these scripts?

**Solution**

Being a SQL Server DBA, I always enjoy creating solutions based on databases. For text-based scripts, like T-SQL, batch, PowerShell, VBScript, etc., the file sizes are usually not big, in the range of a few KBs to a few hundred KBs. They are pretty suitable to be stored in a regular SQL Server table with a varchar(max) column. So with PowerShell, we can easily create a script repository in a few lines of codes and the best part is that you can always backup this database to a small file (assuming you can use backup compression), and carry it with you and restore it in a new environment. Searching scripts on file names or the script content on a table is way easier than searching the file system. So here is the solution logic:

1. Use PowerShell get-childitem (alias: dir) to recursively scan through a folder of interest for script files
2. Read the file content via get-content (alias: gc), and then insert the content, with other collected information, into a pre-defined database table.

Here is the code.

First we will create a table in SSMS to store the data.

Use MSSQLTips -- change to your own

if object\_id('dbo.ScriptRepository', 'U') is not null

drop table dbo.ScriptRepository;

go

create table dbo.ScriptRepository (

id int identity primary key

, Folder varchar(300)

, [FileName] varchar(300)

, Script varchar(max) -- change to nvarchar(max) if needed

, LogDate DateTime default Current\_Timestamp

);

Next, we will use a PowerShell script to do the file archiving.

#requires -version 3.0;

$folder ='c:\DBAdmin\'; #change to your own folder

[string[]]$include ="\*.sql", "\*.ps1"; # you may add other file types, such as ".bat", ".mdx", ".xmla" etc.

#create a System.DataTable object for the file uploading

$dt = New-Object System.Data.DataTable;

$dc = New-Object System.Data.DataColumn('Folder', [System.String]);

$dt.Columns.Add($dc);

$dc = New-Object System.Data.DataColumn('FileName', [System.String]);

$dt.Columns.Add($dc);

$dc = New-Object System.Data.DataColumn('Script', [System.String]);

$dt.Columns.Add($dc);

dir -LiteralPath $folder -Include $include -File -Recurse |

foreach {

$dr = $dt.NewRow();

$dr.Folder = $\_.Directory;

$dr.FileName = $\_.name;

$dr.Script = (gc -path $\_.FullName -raw);

$dt.Rows.Add($dr);

}

Save-DataTable -SQLInstance tp\_w520 -DestinationDB mssqltips -DestinationTable 'dbo.ScriptRepository' -SourceData $dt `

-ColumnMapping @{Folder='Folder'; FileName='FileName'; Script='Script'; } # replace tp\_w520 / mssqltips to your own SQL Server instance and database

To read more about the Save-DataTable function you can read my previous [tip](https://www.mssqltips.com/sqlservertip/3632/monitoring-sql-server-transactional-replication-latency-via-powershell-and-rmo/).

When we want to extract a specific script from the repository to a local drive, we can do this like this:

1. Define a query to retrieve the script(s) of interests
2. Define the destination folder where we want to save the script(s)
3. Run the PS script

Import-Module sqlps -DisableNameChecking; #this is only applicable to sql 2012+ versions

set-location c:

[string]$destPath = 'c:\temp\'; # change to you own destination folder

[string]$qry = "select filename, script from dbo.ScriptRepository where id in (1, 3)"; # modify the query to your needs

$db = get-item sqlserver:\sql\tp\_w520\default\databases\mssqltips #change tp\_w520\default to your own sql instance

#mssqltips is the db where dbo.ScriptRepository resides

$ds = $db.ExecuteWithResults($qry);

$ds.tables[0].rows | % { $\_.script | out-file -FilePath "$destPath\$($\_.filename)" -Force}

**Summary**

This tip discusses a simple way to quickly archive all script files to a central repository database. It helps me to preserve my numerous scripts in a place where I can easily access and search for my needed scripts.

There can be other uses for this tip, for example, in a team environment, we can load all the team's scripts into a repository database instead of SharePoint or a shared storage folder, which is difficult for finding a script. If necessary, we can rely on the powerful Full-Text Search function to do sophisticated queries on scripts.

One note, if the length of a fully qualified file name is more than 260 characters, you will get error like the following

*dir :The specified path, file name, or both are too long. The fully qualified file name must be less than 260 characters, and the directory name must be less than 248 characters.*

**Next Steps**

You can modify the code to add any additional properties you want, such as file CreationTime, LastWriteTime, Version, etc. You may also consider categorizing all your scripts based on the folders (assuming your folder names contain some category meaning) and to make it even cooler, you can think bigger, such as how to automate or schedule your administration from the script library, or how to respond to common issues by running pre-defined scripts.

The following tips do similar tasks in different ways, so check them out:

* [Using OPENROWSET to read large files into SQL Server](https://www.mssqltips.com/sqlservertip/1643/using-openrowset-to-read-large-files-into-sql-server/)
* [Import multiple images to SQL Server using SSIS](https://www.mssqltips.com/sqlservertip/2616/import-multiple-images-to-sql-server-using-ssis/)

Test the scripts in development first and modify the PS scripts to make PS functions you can use in command line mode.

Last Update: 11/24/2015