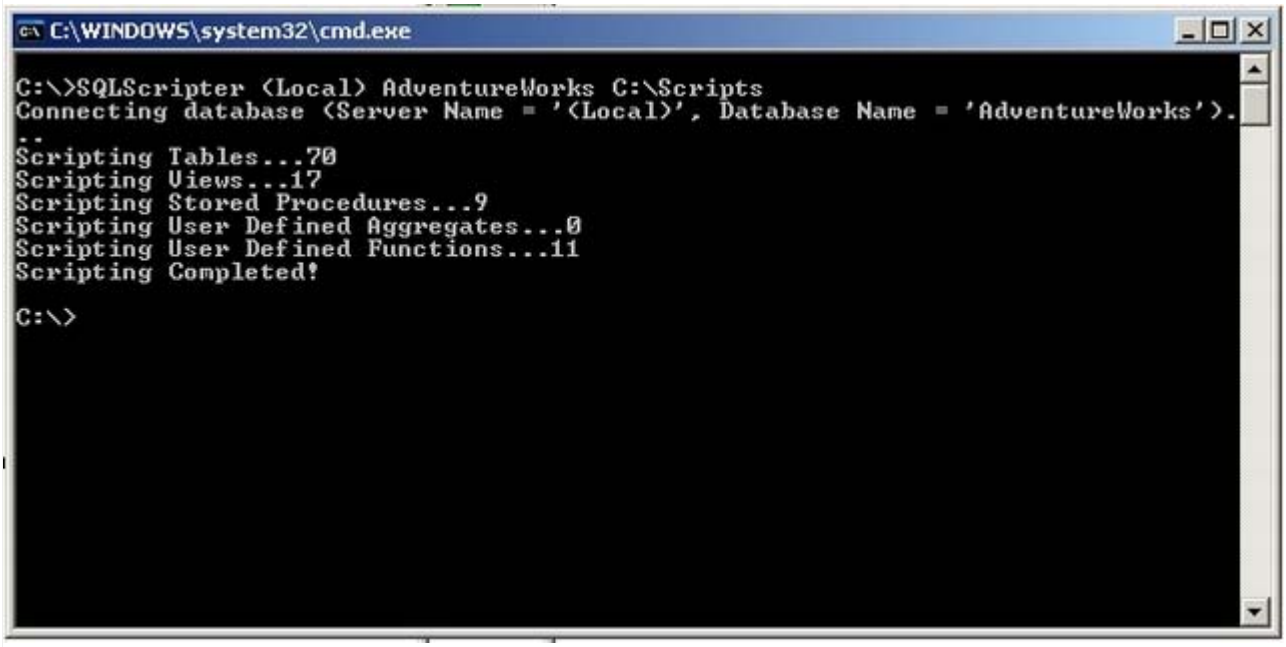
Automatic Script SQL Server 2005 Objects and Commit under Subversion

By **Ferreri Gabriele (Megasoft78)** | 10 Oct 2009

Automatic script SQL Server 2005 objects and commit under Subversion

Download source code - 548.57 KB



Introduction

Recently we changed our source control from Sourcesafe (I know it's prehistoric!) to SubVersion and except for some issues (mainly training ones), we are pretty happy with it. I really like the branching and merging functionalities that give us a big help in our deployment process.

Unfortunately this helps us only for the C# code but because our application is Database-Centric and a lot of our business logic is in tables, views and stored procedures, effectively everything is not under source control, and even more important, we have no history of changes.

The first solution I suggested to solve this issue was to buy Microsoft Visual Studio 2008 Database Edition that effectively give us a very powerful environment to handle the database changes like C# project and have a lot of functionalities like database reverse engineering, schema and data comparison. The first issue with this solution is that Microsoft Visual Studio 2008 Database Edition (VSDE) is strongly linked to Team Foundation Server (TFS), and TFS is not really cheat.

The second issue is that the developers need to change the way they work with databases. In fact, instead of opening SQL Management Studio and just changing the database directly, with VSDE, they will need to create the script to alter the database and then run it against a local temporary database. To be

honest, this seems to me the best way to work but it could be an issue for some developers.

The second solution I suggested to solve the issue was to periodically script the database objects and commit these scripts into SubVersion.

This solution will give me the history about database changes without changing the way we work with databases. This process can be automatic and I can setup on the server machine, using scheduler tasks, to run periodically every 30 minutes. If you like to have more granularity on your history, you should setup the scheduler to run every 15 minutes or less. Consider that if the database has not changes, the process is not going to commit anything.

This tutorial will explain how to create this automatic process.

Background

When I start to investigate on this issue, I was looking for a command line tool that will give me the ability to script all the database objects. I found a lot of nice open source projects that do this in different ways:

- http://scriptdb.codeplex.com/
- http://scriptio.codeplex.com/

Unfortunately scripting the entire database every time for a big project is very slow and for this reason, I tried to create a custom console application that using `Microsoft.SqlServer.Management.Smo` scripts only the differences between the current run and the previous one.

The Implementation

The console application is divided in two classes:

- *Program.cs*: Entry point of the application that handles the input parameters and calls the helper class to script the database objects.
- *ScripterHelper.cs*: The helper class that uses `Microsoft.SqlServer.Management.Smo` scripts the database objects.

The helper class stores the last modified object date in a text file named <Server>-<Database Name>.txt under the application's path. This gives the ability to the class for the next run to detect the changed objects and script just the differences. Embedded into the helper class I implemented a logic to add/delete script files for new/deleted objects in the database.

In this way, using a simple batch file, it is easy to automatically commit the changes in SubVersion.

The most interesting part of the code is how to use `Microsoft.SqlServer.Management.Smo` to script the database objects.

The class used to do that is `Scripter`:

```
Server server = new Server("(local)"); // Server name of the database to script
Database db = server.Databases["AdventureWorks"]         // Database name of the
                                                 // database to script

Scripter scripter = new Scripter(); // Class script the database objects
scripter.Server = server;
scripter.Options.IncludeHeaders = true;
scripter.Options.SchemaQualify = true;
scripter.Options.SchemaQualifyForeignKeysReferences = true;
scripter.Options.NoCollation = true;
scripter.Options.DriAllConstraints = true;
scripter.Options.DriAll = true;
scripter.Options.DriAllKeys = true;
scripter.Options.DriIndexes = true;
scripter.Options.ClusteredIndexes = true;
scripter.Options.NonClusteredIndexes = true;
scripter.Options.ToFileOnly = true;

foreach (Table table in db.Tables) // Script all tables
{
   if (!table.IsSystemObject) // Skip the system objects
   {
      scripter.Options.FileName =
        Path.Combine("C:\Scripts", table.Name + ".sql"); // Filename of the script
      scripter.Script(new Urn[] { table.Urn }); // Object to script
   }
}
```

The integration with SubVersion is done with a small batch file:

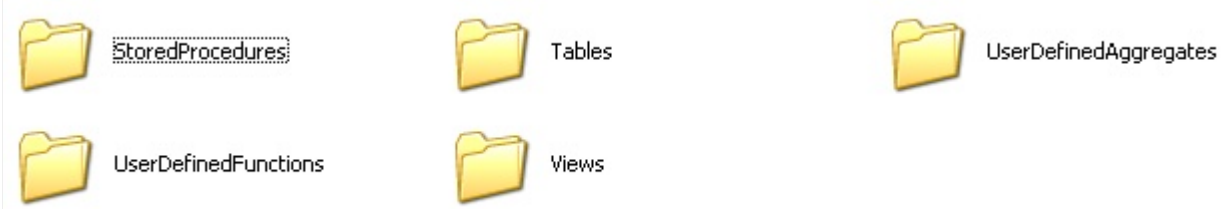
```
rem Script the database objects AdventureWorks under the
        local SQL Server into the folder C:\Scripts
SQLScripter.exe (local) AdventureWorks "C:\Scripts"

rem Change current Folder
C:\Scripts
rem Use for command to detect the file status and eventually
        add or delete the files from svn
for /f "tokens=2*" %%i in ('svn status ^| find "?"') do svn add %%i
for /f "tokens=2*" %%i in ('svn status ^| find "!"') do svn delete %%i

rem Commit the changes in svn
svn commit -m "Automatic commit" "C:\Scripts"
```

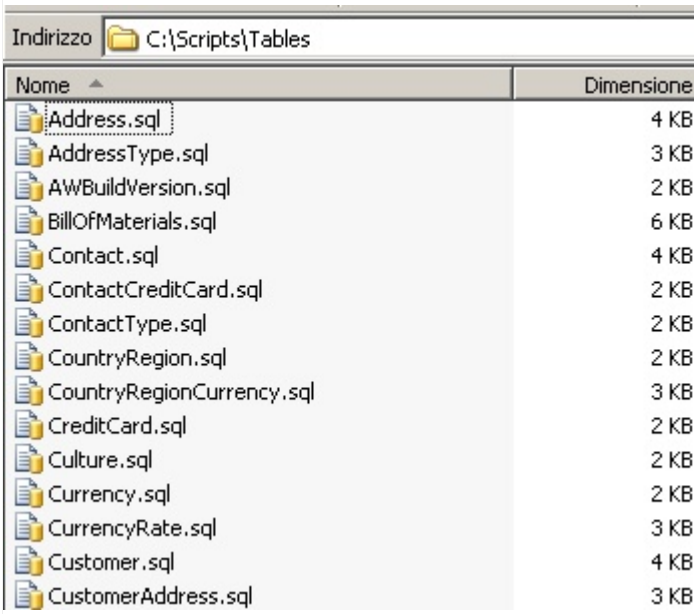
Using the Code

The console application accepts the following parameters:

- Server (Host of the database to script)
- Database Name (Database name of the database to script)
- Scripts Path (Path where the application will store the scripts)
- Object Types (The filter parameter to restrict the object types will be scripted: t=Tables,v=Views, s=Stored Procedures, a=User Defined Aggregates, f=User Defined Functions, Empty=All)

The following screenshots show the result structure created and the *tables* folder for `AdventureWorks` database:





Points of Interest

This solution can be improved to script the data of the lookup tables and put it under source control. Another idea could be to use it to generate a deployment script and simplify the deployment process.

Any suggestions are welcome! :)

History

- 09 October 2009 - First release

License

About the Author

Ferreri Gabriele (Megasoft78)

I'm an Italian Software Developer from about 10 years.
I worked a long time in south Italy (where I was born) and after 2 years in Milan and an year in UK, I'm working remotely from Italy as Senior ASP.NET C# Developer using ASP.NET Ajax technology for a UK company.

Visit my small project:
http://www.codeplex.com/VisualDB

Software Developer (Senior)
sparesFinder
Italy

Member

Comments and Discussions

11 messages have been posted for this article Visit **http://www.codeproject.com/KB/database/SQLScripter.aspx** to post and view comments on this article, or click **here** to get a print view with messages.