Joseph Cooney's Weblog

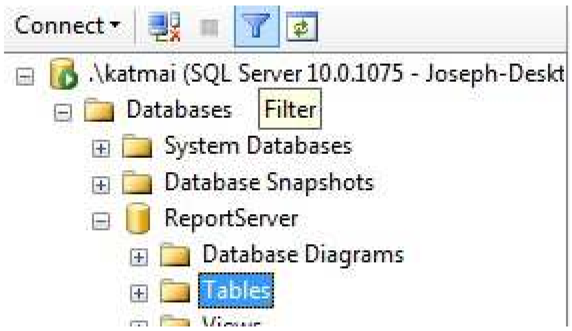
JCooney.NET

The Black Art of Writing a SQL Server Management Studio 2005 Add-In

It all started with me writing this kind of T-SQL query over and over:

```
Select * from sysobjects where xtype = 'U' and
name like '%foo%'
```

For the longest time I've wanted to write a SQL Server Management Studio (SSMS) add-in that would allow me to search for schema entities like tables and stored procedures by name non-modally to help me find things in "big" schemas. SSMS already has a schema entity search/filter feature built-in. You launch it by clicking the button with the funnel-shaped icon as shown in the screen-shot below. I wasn't too enamoured with this since it launches a modal dialog, it only lets you search across one database, and only for one kind of thing at a time (table, stored procedure, view etc). Yes, I am that confused/lazy that I may not know which database I want to look at when I start.



Instead I imagined something where I could press a key to launch the "search" functionality, start typing and see anything that loosely matched what I had typed. Then when I found what I wanted I could click on it in the list of results and go to that item in the object explorer.

My Links

Home
Contact
Syndication XML
Login

Blog Stats

Posts - 346
Stories - 1
Comments - 619
Trackbacks - 102

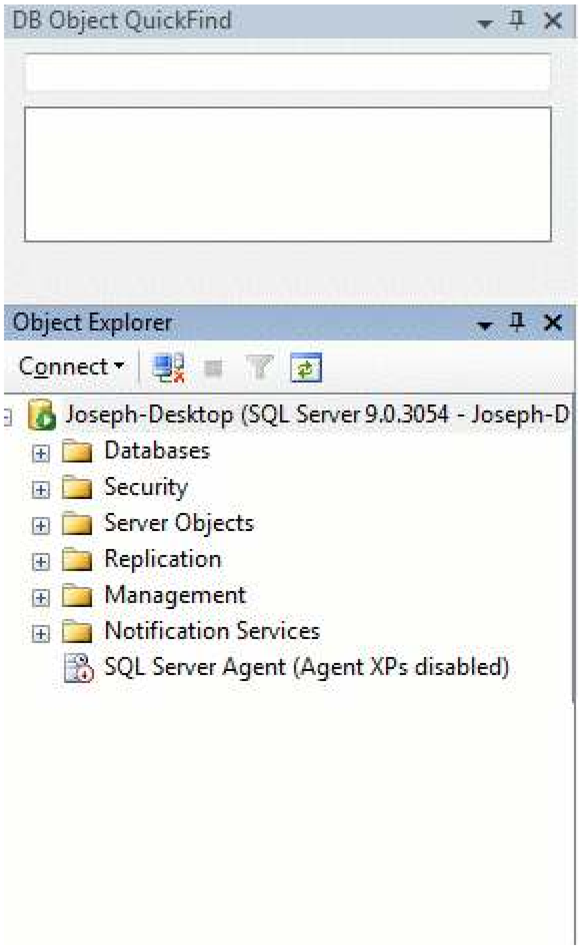
News

I work for:

see also:
Dominic Cooney
Patrick Cooney

Archives

December, 2007 (1)
November, 2007 (7)
October, 2007 (4)
September, 2007 (8)
August, 2007 (5)
July, 2007 (3)
June, 2007 (2)
May, 2007 (5)
April, 2007 (8)
March, 2007 (4)
February, 2007 (5)
January, 2007 (7)
November, 2006 (7)
October, 2006 (5)
September, 2006 (4)



Downloads

Download the MSI here if you just want to use the add-in

Download the code here if you want to modify the add-in or create your own

So Why Does This Qualify as a Black Art?

Writing add-ins for SSMS is something of a black art. It's not supported, the APIs aren't documented or designed for extensibility, and it's really only possible by virtue of the fact that SSMS is built on top of the Visual Studio codebase. The only decent coverage I was able to find on the web was this article on ASP Alliance which doesn't provide any code. I'm hoping by writing this post and providing a working add-in I can start a revolution in SQL Server Management Studio add-in writing.

Why You Should Write a Cool Add-In

So I can use it. So other developers can use it. Writing stuff against supported, published, documented APIs is....well...quite frankly it's passé. With intellisense and

August, 2006 (7)
July, 2006 (4)
June, 2006 (2)
May, 2006 (5)
April, 2006 (5)
March, 2006 (7)
February, 2006 (8)
January, 2006 (13)
December, 2005 (3)
November, 2005 (12)
October, 2005 (9)
September, 2005 (11)
August, 2005 (11)
July, 2005 (15)
June, 2005 (13)
May, 2005 (6)
April, 2005 (8)
March, 2005 (13)
February, 2005 (20)
January, 2005 (5)
November, 2004 (7)
October, 2004 (7)
September, 2004 (6)
August, 2004 (6)
July, 2004 (7)
June, 2004 (9)
May, 2004 (4)
April, 2004 (7)
March, 2004 (9)
February, 2004 (3)
January, 2004 (7)
December, 2003 (5)
November, 2003 (9)
October, 2003 (3)
September, 2003 (1)
August, 2003 (2)
July, 2003 (2)

Image Galleries

General

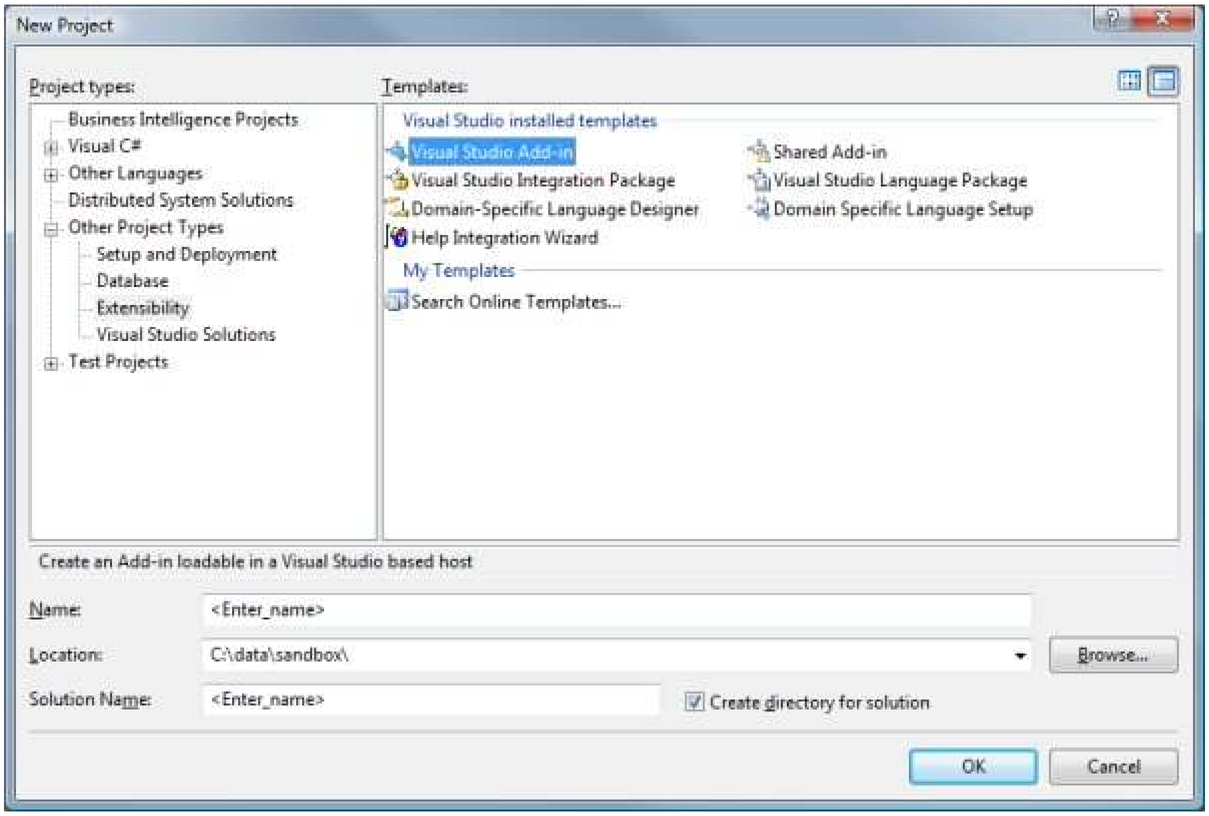
My GotDotNet Samples

Async Callbacks in ASP.NET 2.0
m3rlin - templating code generator
SmartResize - form resize with less CPU overhead
Typed Dataset Generator
XPath Expression Tester

google and codeproject and the MSDN and blogs even infrastructure people can write code these days. So where's the fun? It's here my friends – all the pain of the Visual Studio object model combined with the added bonus of being unsupported and undocumented!

Starting a new Add-In Project

Creating a new SSMS add-in is very much like creating a new Visual Studio 2005 add-in. Expand the "Other Project Types" node on the left-hand side of the "New Project" dialog and select the "Extensibility" node. Then choose a Visual Studio Add-in.


New Project
Project types:
Business Intelligence Projects
Visual C#
Other Languages
Distributed System Solutions
Other Project Types
Setup and Deployment
Database
Extensibility
Visual Studio Solutions
Test Projects
Templates:
Visual Studio installed templates
Visual Studio Add-in
Shared Add-in
Visual Studio Integration Package
Visual Studio Language Package
Domain-Specific Language Designer
Domain Specific Language Setup
Help Integration Wizard
My Templates
Search Online Templates...
Browse...
OK
Cancel

Create an Add-in loadable in a Visual Studio based host

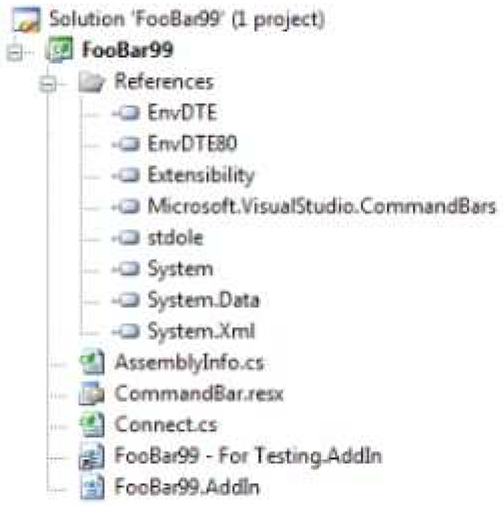
Name: <Enter_name>

Location: C:\data\sandbox\

Solution Name: <Enter_name>

☑ Create directory for solution

Step through the rest of the Visual Studio Add-in wizard, and examine the output. You should have a project that looks something like the following:



```
Solution 'FooBar99' (1 project)
FooBar99
    References
        EnvDTE
        EnvDTE80
        Extensibility
        Microsoft.VisualStudio.CommandBars
        stdole
        System
        System.Data
        System.Xml
    AssemblyInfo.cs
    CommandBar.resx
    Connect.cs
    FooBar99 - For Testing.AddIn
    FooBar99.AddIn
```

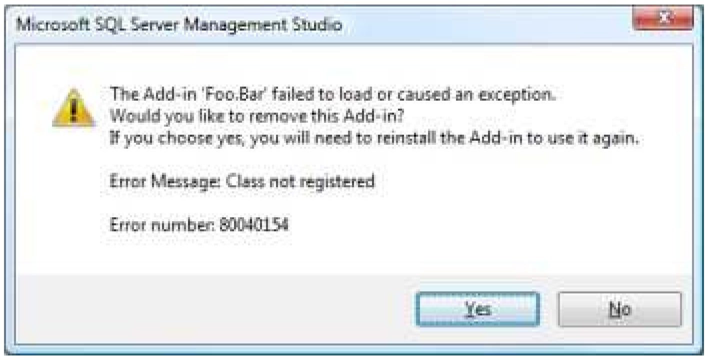
You can delete the .AddIn files – Visual Studio 2005 switched to a nicer, registry-free way of add-in registration using XML files, however SSMS does not seem to use these. Instead for SSMS to use your add-in you'll need to add some information about it to the windows registry.

Add-In Registration

As a .NET developer you probably hardly ever have to look at the registry. For those of you young enough to have never done any Microsoft development before .NET you've probably NEVER had to look at the registry. That's all about to change, because registering your SSMS add-in is ALL about the registry.
To register your add-in you'll have to add the following keys and values to the registry:

Key Name	Type	Value
CommandLineSafe	REG_DWORD	[0=addin is not necessarily command-line safe and may display UI, 1=command-line safe]
LoadBehavior	REG_DWORD	[0=manually loaded, 1=loaded automatically]
ProductDescription	REG_SZ	[a textual description of your add-in]
ProductName	REG_SZ	[the name of your add-in]
SatelliteDllName	REG_SZ	[name of the dll your add-in lives in]
SatelliteDllPath	REG_SZ	[path on the file system to where your add-in will be installed]

All of this information should be contained in a registry key with the namespace of your add-in, and the type inside that namespace that implements IDTExtensibility2. If you created a new add-in via the wizard in Visual Studio this will be .Connect.
There are a few different places in the registry this registry key can live. One is Computer\HKLM\SOFTWARE\Microsoft\Microsoft SQL Server\90\Tools\Shell\Addins\ (or Computer\HKLM\SOFTWARE\Wow6432Node\Microsoft\Microsoft SQL Server\90 \Tools\Shell\Addins\ for x64 based versions of windows).
Another is the equivalent registry hive for the current user: Computer\HKCU\SOFTWARE\Microsoft\Microsoft SQL Server\90\Tools\Shell\Addins\ - I think this is a better place to register your add-in because it isn't x86/x64 specific, and can be written to without admin privileges. The downside is that it will only register the add-in for that user. Adding this registration information as part of the add-in deployment is handled by the Registry target in the setup project.
If some aspect of your add-in registration is incorrect or your add-in throws an exception on load you will see the following error message:


Microsoft SQL Server Management Studio

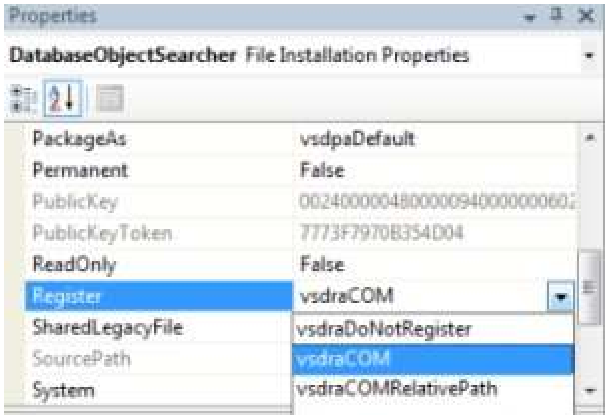
The Add-in 'Foo.Bar' failed to load or caused an exception.
Would you like to remove this Add-in?
If you choose yes, you will need to reinstall the Add-in to use it again.

Error Message: Class not registered

Error number: 80040154

Clicking "Yes" here will delete the registration information for the add-in in the registry, so don't do that unless you like re-typing all that stuff. Once you get the registry stuff right you should probably export the key as a back-up in case you accidentally DO click

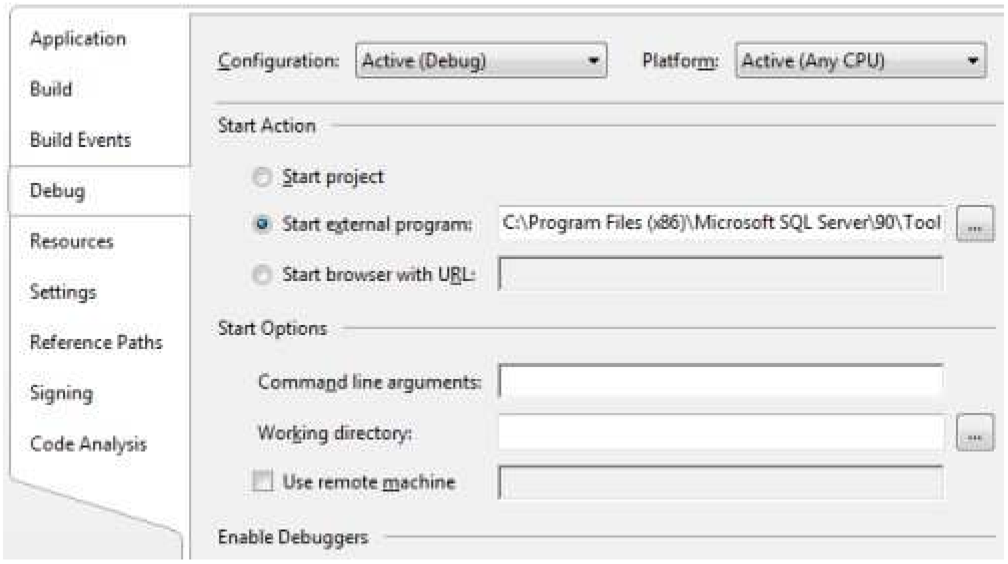
yes one time, or if you want to develop on a different machine.
SSMS is based on the Visual Studio shell, which still uses COM extensively. For this reason the type in your add-in that implements IDTEExtensivility2 and handles the "plumbing" inside of SSMS needs to be COM-visible, and registered (hence the GUID attribute on the Connect class if you ran the Visual Studio add-in wizard). Visual Studio handles the registration when you build the add-in on your machine, but you'll need to take care of that when you deploy your add-in to other users to get past the classic "but it works on my machine" problem.
You do this in your installer project by changing the Register property on the assembly from vsdraDoNotRegister to vsdraCOM as shown here



PackageAs	vsdpaDefault
Permanent	False
PublicKey	002400000480000094000000060
PublicKeyToken	7773F7970B354D04
ReadOnly	False
Register	vsdraCOM
SharedLegacyFile	vsdraDoNotRegister
SourcePath	vsdraCOM
System	vsdraCOMRelativePath

Add-In Debugging

To debug the add-in I set the "Debug" option to "Start external program" and provide a path to the SQL Server Management Studio executable as shown in the following screen shot.



Configuration: Active (Debug)
Platform: Active (Any CPU)
Start external program: C:\Program Files (x86)\Microsoft SQL Server\90\Tool
Command line arguments:
Working directory:

For x86 systems this will typically be something like:
C:\Program Files\Microsoft SQL Server\90\Tools\binn\VSShell\Common7\IDE\SqlWb.exe
For x64 it might look something like this:
C:\Program Files (x86)\Microsoft SQL Server\90\Tools\binn\VSShell\Common7\IDE\SqlWb.exe

The SSMS Object Model

SQL Server Management Studio must have once been known as SqlWorkbench, based on

the assembly and executable names. The foundation for understanding the managed code in SSMS is an assembly called SqlWorkbench.Interfaces.dll (typically located in the following location: C:\Program Files\Microsoft SQL Server\90 \Tools\binn\VSShell\Common7\IDE\SqlWorkbench.Interfaces.dll – add an ' (x86)' in after "Program Files" for an x64 machine)
This assembly contains 5 namespaces which in turn contain many classes and interfaces for interacting with SSMS. Since the purpose of this assembly seems to be a place to define common interfaces it is unsurprising that the majority of types in this assembly are interfaces. The namespaces are:

- Microsoft.SqlServer.Management
- Microsoft.SqlServer.Management.QueryExecution
- Microsoft.SqlServer.Management.SqlMgmt
- Microsoft.SqlServer.Management.UI.VSIntegration
- Microsoft.SqlServer.Management.UI.VsIntegration.ObjectExplorer

The concrete classes that implement these interfaces (which you will be dealing with at run-time) seem to be located in separate assemblies of their own. For example the Object Explorer concrete classes are in the ObjectExplorer.dll assembly, which should be in the same directory as theSqlWorkbench.Interfaces.dll. The reflector class browser was invaluable (as always) when trying to find my way around the object model. I'd advise you to add every single managed dll in the same directory as SqlWorkbench.Interfaces.dll into reflector and use the search functionality (search is the new UI) in that tool to find your way around.

SSMS uses another API called Sql Management Objects (or SMO for short) for lots of things. Unlike the SSMS APIs SMO is a fully documented, supported piece of code developed by the SQL team for administering SQL Server. SMO is the logical successor to DMO, a COM API for administering SQL Server 2000. SMO is fully managed and one of the best-designed APIs I've come across in recent times. If you want to script out SQL objects, enumerate through tables/views/columns or programmatically manipulate your SQL database this is the way to do it. In developing add-ins I would try to leverage SMO as much as possible since it integrates well with SSMS, and is much better documented and supported.

Since SSMS is built on Visual Studio knowledge of the Visual Studio object model is also quite helpful. Visual Studio makes extensive use of the "service provider" design pattern, and the IServiceProvider interface. Many, many objects in the VS object model (and SSMS also) implement this interface, or inherit from other objects that implement this. This API is means you should always be able to "get" the type you need, and presumably makes the object model smaller and less confusing. Unfortunately I find it hard to use, since there feels like less guarantees about if GetService will return you anything, and under what circumstances this might happen. Also the very high granularity of objects in the VS object model usually means it is hard to know which type to ask for, as there are often several likely-sounding candidates.

Adding Fuzzy Search

My original vision for searching for tables and stored procedures in the schema using T-SQL always involved a "%LIKE%" type search, and I wanted my add-in to continue in this tradition as much as possible. Fortunately I was able to find a commercial component called highlight express developed by a small ISV ShuffleText. They were keen for feedback, and the code changes required to use their stuff were so modest I decided to give it a go. I've been really happy – their stuff is all pure .NET code and runs quite fast. Now I keep thinking of other places a fuzzy text search would be useful – my messenger buddy list, Intellisense, active directory lookups, there are lots of places for non-exact-match searching could be used.

Add-Ins and Management Studio Express

Microsoft offer an express version of SSMS for administering SQL Server Express instances (or other SQL Server instances if necessary). Unfortuately I was unable to get my add-in to load and run inside the Express version. The SSMS tools pack (another free but not open-source) SSMS add-in say that they support SQL Server Management Studio Express. Other posts such as this one make it seem even less supported. SqlAssist (a pretty cool commercial product which to my knowledge is written as an add-in in the same manner I have described) states that the express version of SSMS does not allow add-ins.

TODO Tasks and Excuses for my Lame Add-in Code

Q: Does it work in SQL Server 2008 (Katmai)?
A: The add-in doesn't work in the latest (NOV 2007 CTP) of SQL Server 2008 (Katmai). It loads and doesn't crash, but doesn't work either. Naturally I'd prefer if it did work.

Q: Why do you have a static "Current" property on the SearcherController? Don't you know Singletons are evil?
A: I wanted to pass an instance of a SearcherController to the add-in UI ala dependency injection. I even created an interface IObjectSearcherUI which exists solely to allow the SearcherController to be injected. The CreateAddinWindow method of the SearcherController actually creates the VS toolwindow that hosts the add-in UI, however I was unable to find a way to get hold of an instance of the SearcherToolWindow in order to inject the dependency. I've left the interface in there in the hopes that I (or one of my clever readers) can show me how to get hold of the window instance, and get rid of the singleton forever!

Q: Why is the UI part factored out from the rest of the add-in? You've hard-coded which control you're going to instanciate so what's the point?
A: I thought about writing two UIs – one in WPF and one in WinForms. If WPF was present I was going to load that, and otherwise fall back to the legacy winforms, however I didn't get around to implementing both.

Q: You only search on databases, tables, views and stored procedures – could you search on columns or user defined functions too?
A: Certainly. To do that you should enhance the BuildDBObjectDictionary method in the DatabaseObjectSearcher, the searchText_TextChanged event handler in the SearcherToolWindow and probably the ObjectExplorerManager to handle the correct "drill down" to the right level.The code inside the ObjectExplorerManager is probably the ugliest code in the whole solution.

Q: Why the ugly reflection hacks in the ObjectExplorerManager?
A: Those methods weren't included in the interface that the ObjectExplorer exposed (IExplorerManager) but were necessary for me to "drill down" into the tree-view and select an item. I wish there had been another way.

Q: I tried to connect to a remote SQL Instance with a bazillion databases over a dial-up connection and your add-in didn't seem to do anything. What's going on?
A: When the add-in loads it builds a dictionary of all the things you might want to search on. If lots of data has to be retrieved from remote servers over slow networks then that

is going to be slow. We lock the dictionary while we're building it (to prevent people querying it while it is still being built), and also when we're querying it (since WinForms applications are STA threaded there shouldn't ever be more than 1 UI thread accessing the dictionary at a time anyway). In order to prevent the add-in locking up SSMS if the dictionary is still being built we wait for 1 second to see if we can acquire a lock on the dictionary. If we can't we just return nothing from the search results. You're welcome to look at the internals of the DatabaseObjectSearcher and the BuildObjectDictionary method to try and make this faster if you wish.

Please post any suggestions, bugs, anything I've overlooked, registration hacks, other tutorials I didn't find etc. as comments here

posted on Monday, November 26, 2007 8:17 AM

Feedback

# re: The Black Art of Writing a SQL Server Management Studio 2005 Add-In
11/27/2007 9:28 AM Joeseph

Couple adds-ins here. No downloadable code although you could probably ask and compare notes.
http://sqlblogcasts.com/blogs/seanprice/archive/2007/07/15/sql-management-studio-snapshot-add-in.aspx

# re: The Black Art of Writing a SQL Server Management Studio 2005 Add-In
11/28/2007 7:54 AM Mladen

Hi!
I'm the author of Ssms Tools Pack
and i can tell you that ssms express does allow add-ins with a few workarounds.
send me a mail via my blog or the ssms tools pack page and we can talk if you're interested...

about the search window injection. i've played with this a lot... with no luck since the tools windows (search is one) are pure com objects and they don't have any methods to hook into.

i've also tried to remove singletons from my add-in but i've seen that they work very well are not at all evil in this case :)

# re: The Black Art of Writing a SQL Server Management Studio 2005 Add-In
11/29/2007 2:51 PM Alek Davis

Great article, and great add-in. Thank you!

# links for 2007-11-30 11/30/2007 3:20 AM guidmaster´s .NET blog

Free Silverlight and WPF Training | Joshua | MIX Online (tags: Silverlight wpf) The Black Art of Writing

# Links of the week #13 (week 48/2007) 12/2/2007 2:35 PM Bite my bytes

Links of the week #13 (week 48/2007)

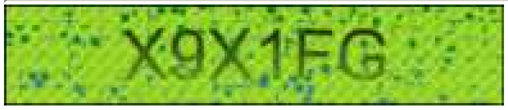
# re: The Black Art of Writing a SQL Server Management Studio 2005 Add-In 12/3/2007 4:26 AM Darren Gosbell

Hey great post Joeseph. I'm going to see if I can make some time to have a look and see if I can't build a couple of small features for SSAS.

# re: The Black Art of Writing a SQL Server Management Studio 2005 Add-In
12/17/2007 4:28 PM Chinkit Patel

Well Done Mate! Love your

Title	re: The Black Art of Writing a SQL Server Manage
Name	
Url	
Security Word (prevent comment spam)	



Comments

☐ Remember Me?

Submit

Powered by: