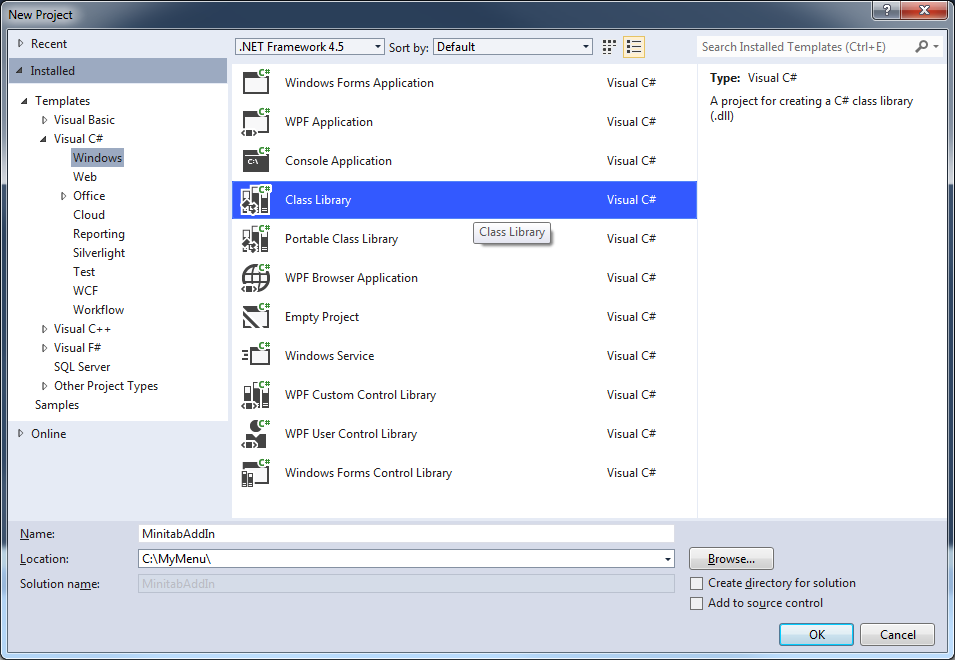
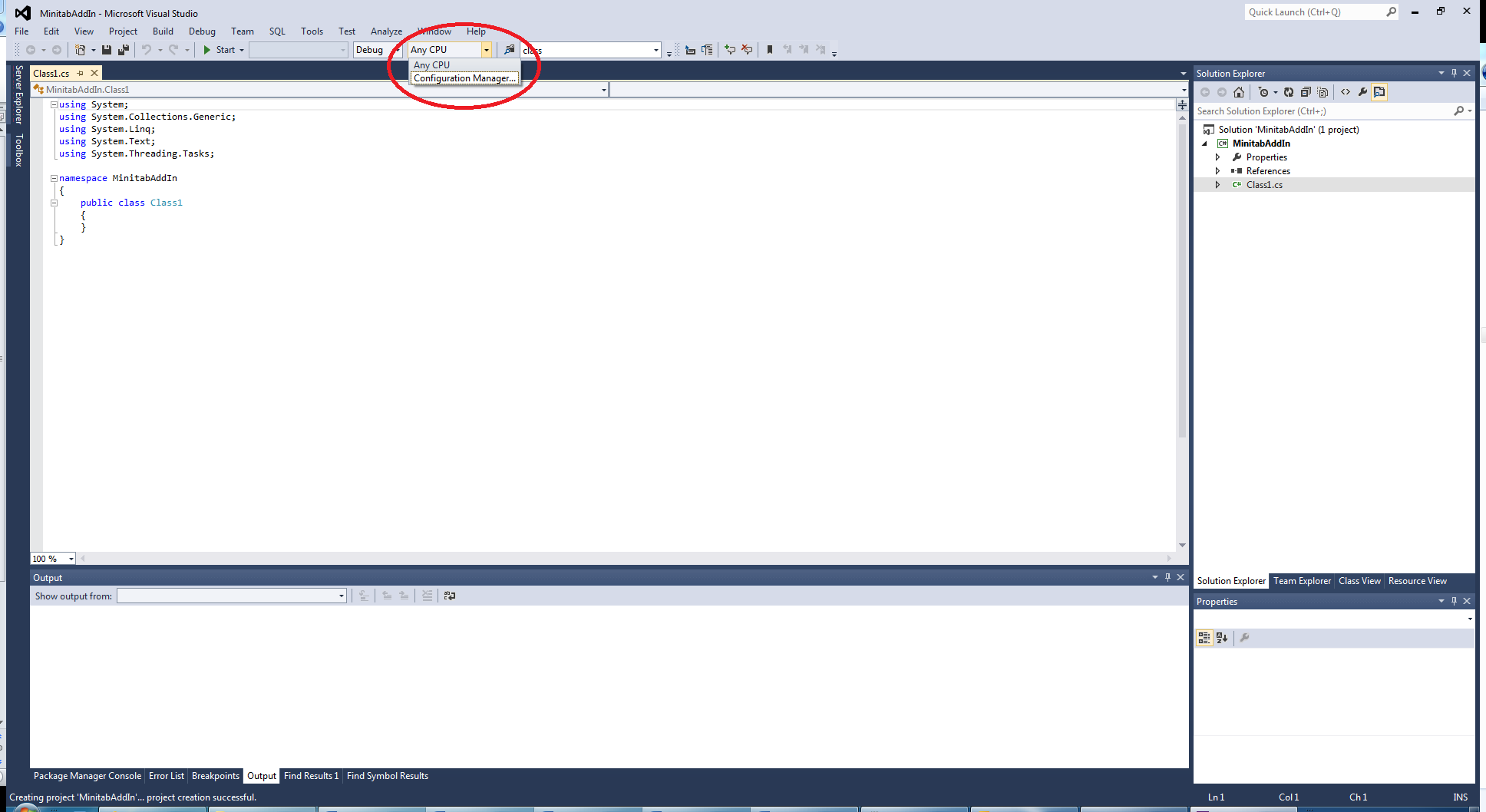
The following steps will allow you to create a C# add-in to Minitab 16 or 17.

1. In Microsoft Visual Studio 2012 (or 2010), create a new C# class library project.

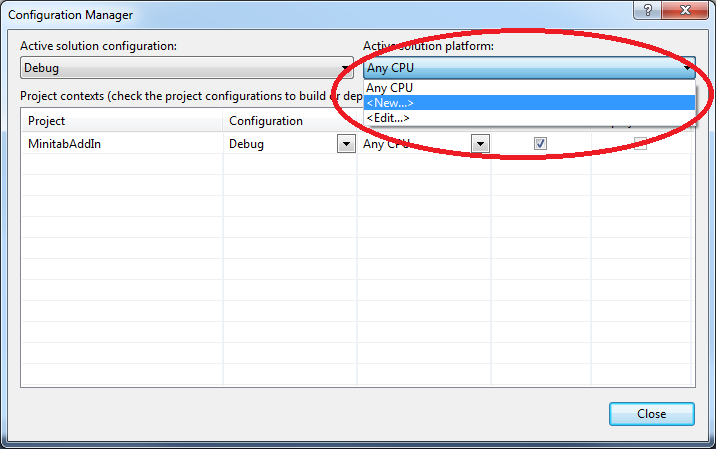


1. Set the solution platform to “x86”.

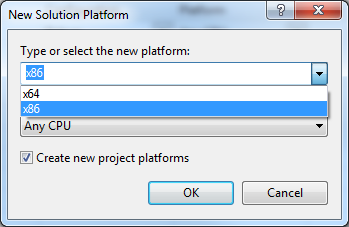
Click on the Solutions Platform drop-down and select “Configuration Manager…”.



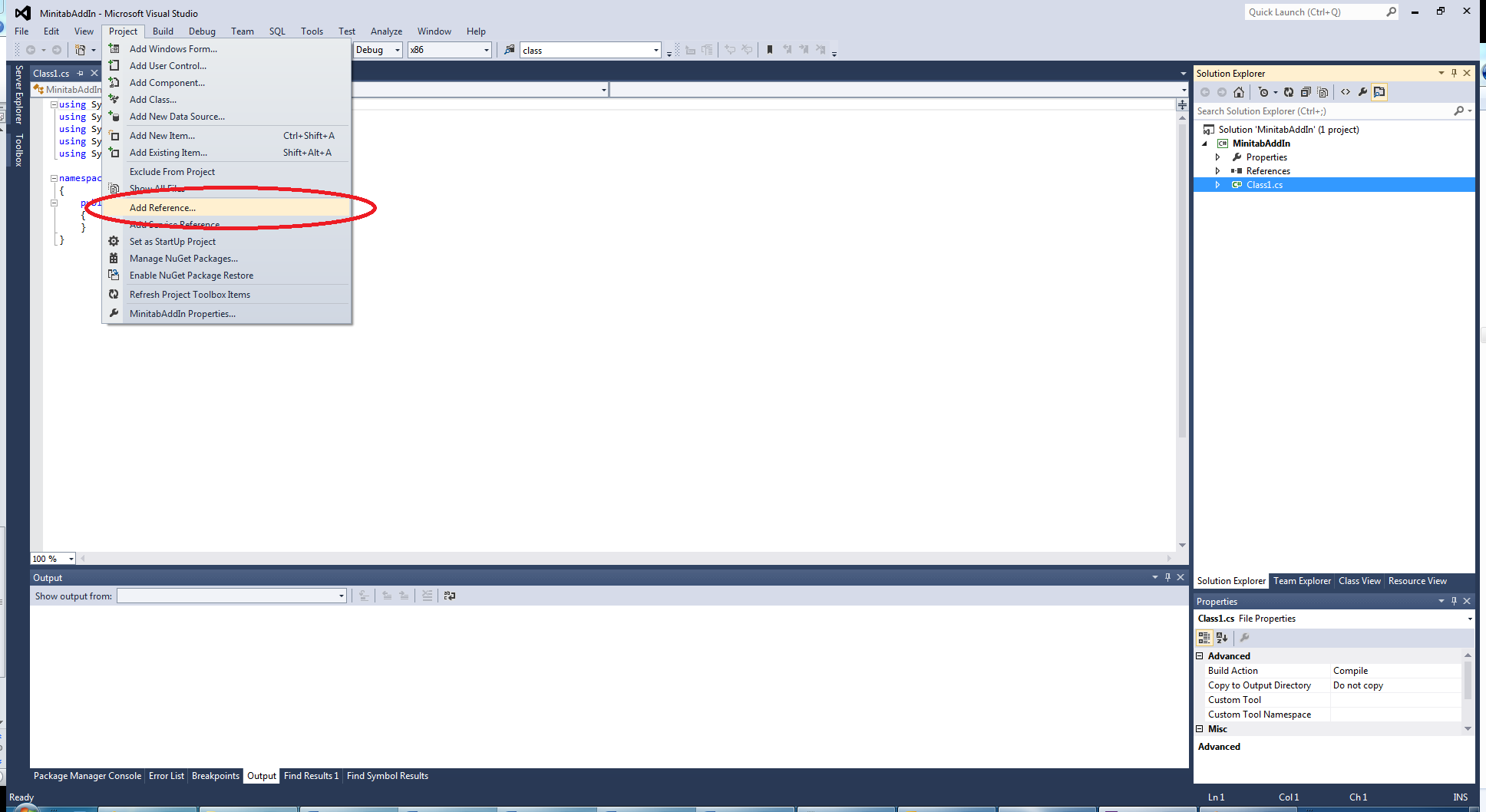
In the Active solution platform down-down, select “<New…>”.

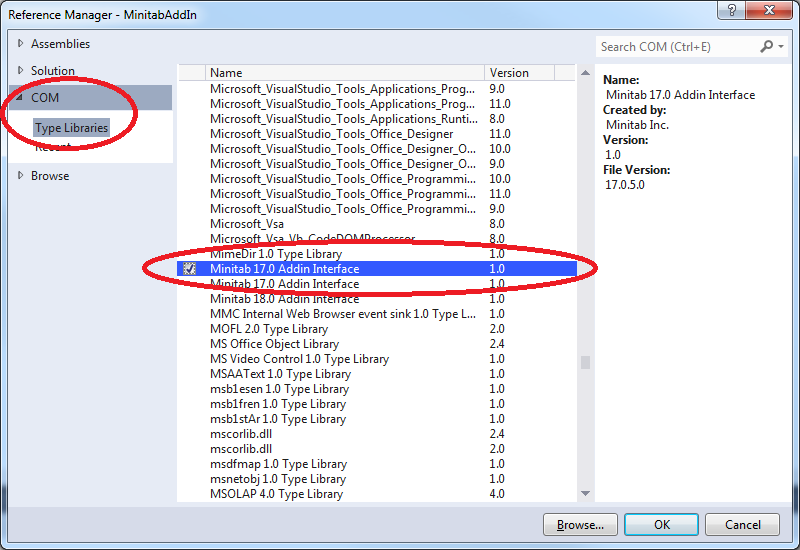


Choose “x86” as the new platform:



1. Add a COM reference to the Minitab 17.0 Addin Interface (or Minitab 16.0 Addin Interface if you are using Minitab 16).





1. Add the following using directive to your .cs file:

using System.Runtime.InteropServices;

1. Add a public class implementing the MinitabAddinTLB.IMinitabAddin interface. Your class will need to implement the following 9 methods:

public void OnConnect(int iHwnd, object pApplication, ref int iFlags)

public void OnDisconnect()

public string GetName()

public string GetDescription()

public void GetMenuItems(ref string sMainMenu, ref System.Array saMenuItems, ref

int iFlags)

public string OnDispatchCommand(int iMenu)

public void OnNotify(MinitabAddinTLB.AddinNotifyType eType)

public bool QueryCustomCommand(string sName)

public void ExecuteCustomCommand(string sName, ref System.Array saArgs)

See Minitab Automation Help for descriptions of what these methods should contain.

1. Add these 4 attributes to your class:

[ComVisible(true)]

[Guid("*EB4B83A5-4376-4839-80FA-D3081C58BA73*")]

[ClassInterface(ClassInterfaceType.None)]

[ProgId("*namespace*.*class*")]

Change the GUID above to a new unique GUID and change “*namespace*.*class*” to be your namespace and class name.

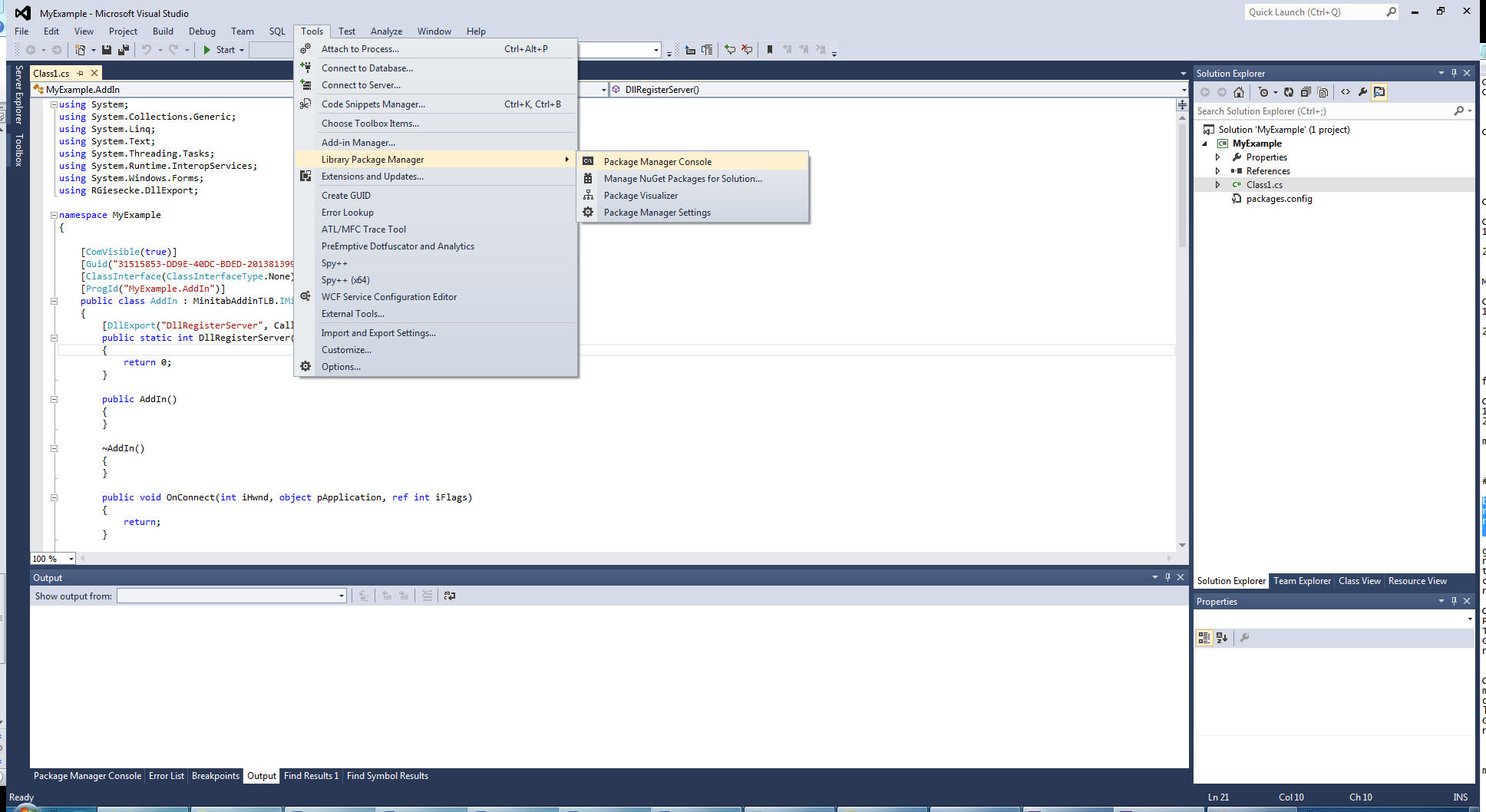
1. In the OnDisconnect method, make sure to perform garbage collection:

GC.Collect();

1. Minitab’s add-in framework was designed for COM dll’s. To have a .Net dll work, it needs to export a DllRegisterServer method. This method doesn’t need do anything (although it can be used to create appropriate registry entries for the add-in) but Minitab needs to be able to call it. This step explains how to have your dll export a static method.

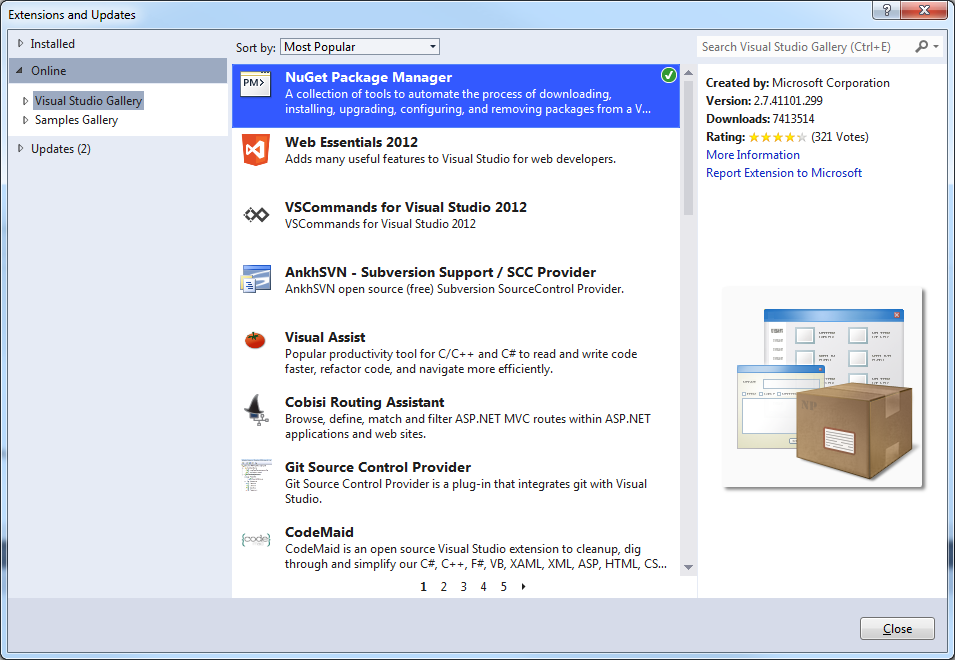
Make sure you have the NuGet Package Manager installed in Visual Studio.

If you do, you will have a Tools > Library Package Manager > Package Manager Console menu item in Visual Studio:



If it is not installed, choose Tools > Extensions and Updates …

Choose the “Online” choice on the left and then “Visual Studio Gallery”. “NuGet Package Manager” should be at (or near) the top of the list. Select it and then click “Download”. Follow any instructions.



(In the screenshot above, the NuGet Package Manager has already been installed. If it is not installed, there will be a “Download” button in place of the green circled checkmark.)

Now select Tools > Library Package Manager > Package Manager Console…

At the PM> prompt, type

Install-Package UnmanagedExports

And press Enter.

NOTE: if you are using the 64-bit version of Visual Studio, you want to do this instead:

Install-Package UnmanagedExports –Version 1.2.3-Beta

After a minute or so, you should see:

Successfully installed 'UnmanagedExports 1.2.6'.

Successfully added 'UnmanagedExports 1.2.6'

Your project should now have a new reference to

RGiesecke.DLLExport.Metadata

Add the following using directive to your .cs file:

using RGiesecke.DllExport;

Add the following method to your class:

[DllExport("DllRegisterServer", CallingConvention = CallingConvention.StdCall)]

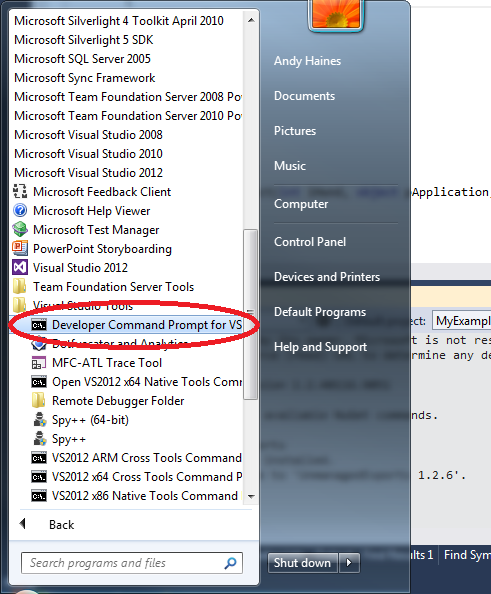
public static int DllRegisterServer()

{

return 0;

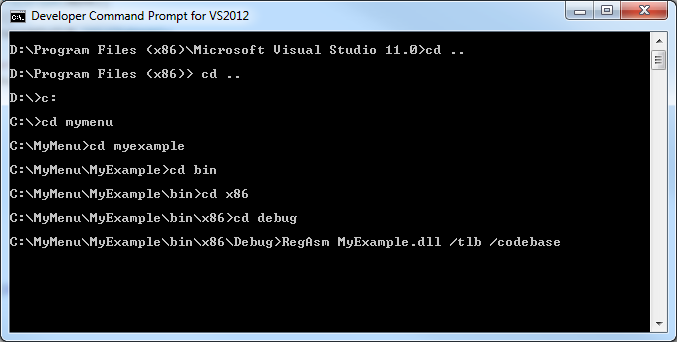
}

1. Now build the solution.
2. Open the Developer Command Prompt for VS201x:



(Make sure to run the prompt as an administrator)

Using RegAsm with the /tlb and /codebase switches to create a type library file and to add the proper flags to the registry.



1. The remainder of the steps will allow us to embed the type library inside the dll. Create a new text file with the extension .rc that contains the following line:

1 typelib "C:\MyMenu\CSharpExample\bin\\x86\Debug\CSharpExample.tlb"

Change the path and name of the file to whatever is correct for your .tlb file.

1. Create a compiled resource file from the above .rc file. To do this, in the Developer Command Prompt, run

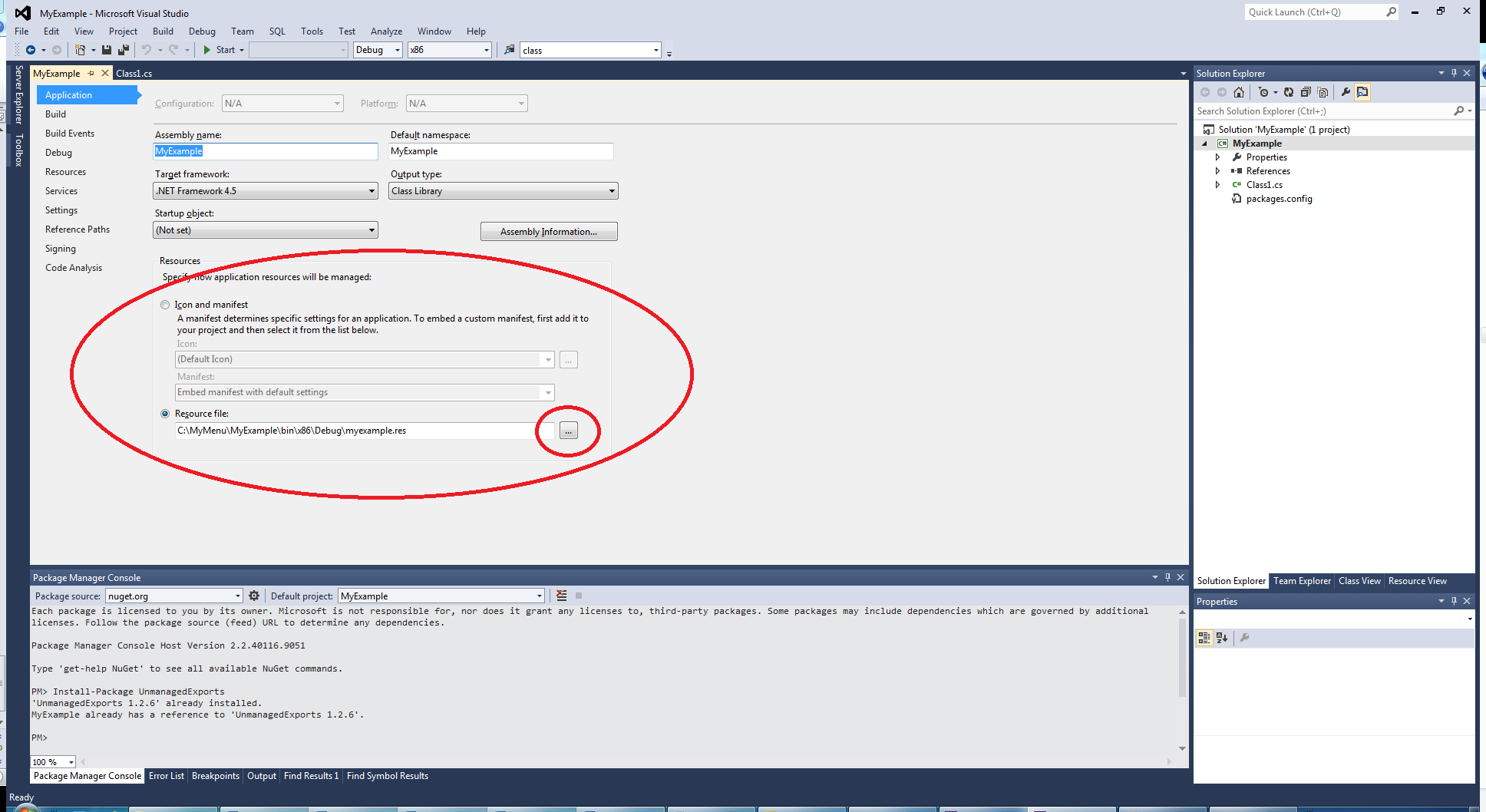
rc *yourfile.rc*

changing “*yourfile*” to whatever is appropriate. This will create a

*yourfile*.res

file.

1. In your application properties, change from “Icon and manifest” resources to the compiled resource file you just created.



Choose the Resource file radio button and then click the “…” button and navigate to the .res file.

1. Rebuild the solution (make sure to rebuild and not just build)