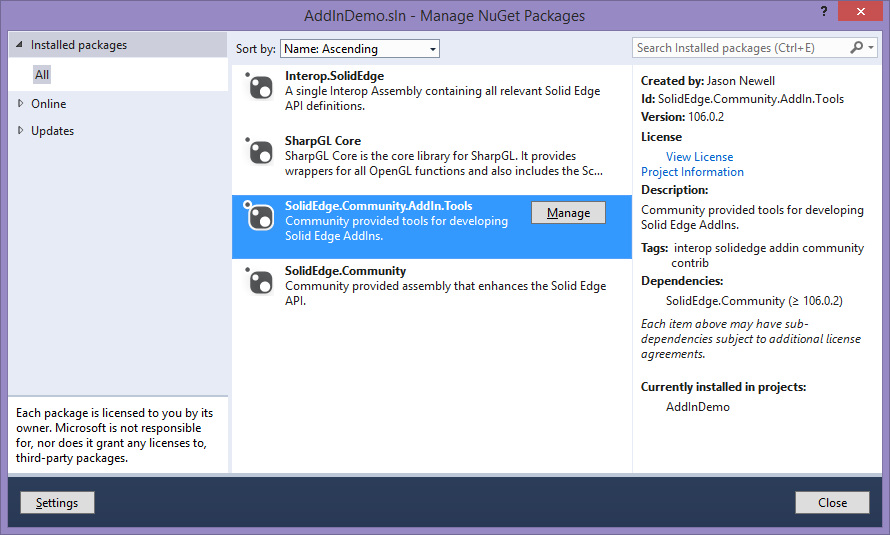
Solid Edge AddIn Demo

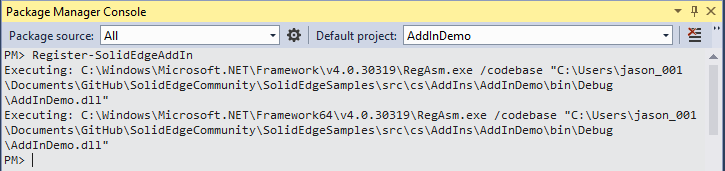
# NuGet

This project makes use of open source projects via [NuGet](https://www.nuget.org/). The NuGet packages used are shown below. Of particular interest is the [SolidEdge.Community.AddIn.Tools](https://www.nuget.org/packages/SolidEdge.Community.AddIn.Tools) package. It has dependencies on [Interop.SolidEdge](https://www.nuget.org/packages/Interop.SolidEdge) and [SolidEdge.Community](https://www.nuget.org/packages/SolidEdge.Community). It also installs package manager console commands and post build project events that enable you to embed images as native resources into the assembly.

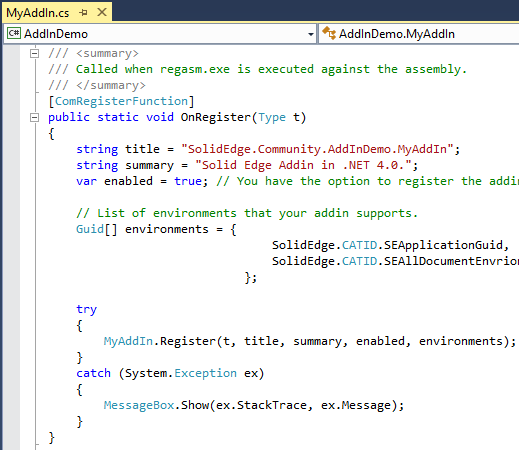


# Registration

Registering the addin has never been easier due to the [SolidEdge.Community.AddIn.Tools](https://www.nuget.org/packages/SolidEdge.Community.AddIn.Tools) package. In Visual Studio 2012\2013, navigate to Tools -> NuGet Package Manager -> Package Manager Console. Type ***Register-SolidEdgeAddIn*** and hit enter. This command will execute the x86 & x64 version of regasm.exe against your assembly.

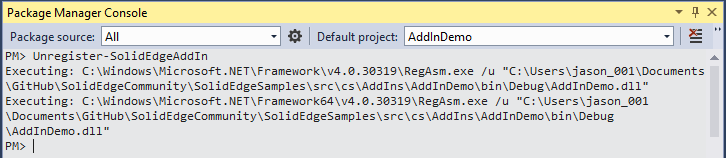


When regasm.exe is executed against the assembly, the OnRegister method will get called. This is where you can control the registration process of your addin.

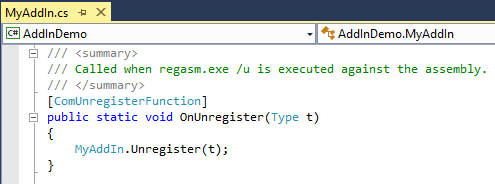


# Unregistration

Unregistering the addin is just as easy. In Visual Studio 2012\2013, navigate to Tools -> NuGet Package Manager -> Package Manager Console. Type **Unr*egister-SolidEdgeAddIn*** and hit enter. This command will execute the x86 & x64 version of regasm.exe /u against your assembly.

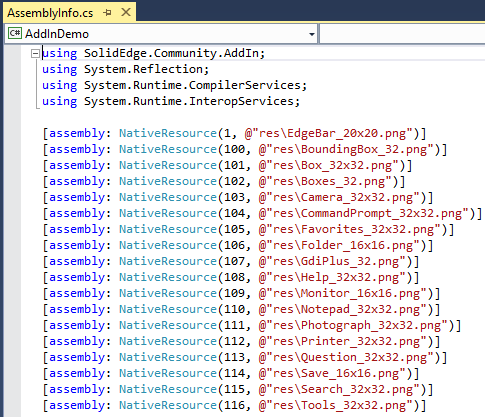


When regasm.exe /u is executed against the assembly, the OnUnregister method will get called. This is where you can control the unregistration process of your addin.

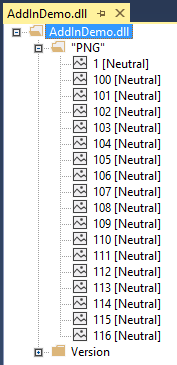


# Native Win32 Resources

Embedding native Win32 resources for your EdgeBar and Ribbon is made possible by the [SolidEdge.Community.AddIn.Tools](https://www.nuget.org/packages/SolidEdge.Community.AddIn.Tools) package. Add your BMP\PNG resources to a folder under the project directory and modify the AssemblyInfo.(cs|vb) with the NativeResource attribute as shown below. After you build the solution, the resources will automatically get embedded into your .dll.



You can open your .dll in Visual Studio, Resource Hacker, etc. to verify that the resources got embedded during the post build event. If you do open your .dll in this manner, be sure to close it before attempting another build or you will get a build error due to the file being in use.

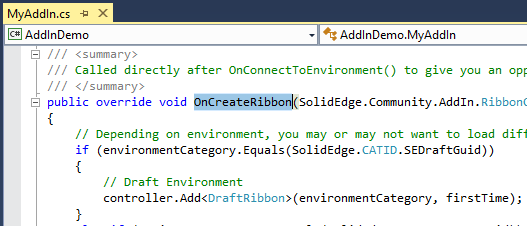


# Ribbons

You can now completely define your Ribbon in XML as shown below. The imageId attribute for each control points to a NativeResource that you previously defined in AssemblyInfo.(cs|vb).



The OnCreateRibbon method of your addin will get called when it’s time to create the Ribbon.



In the following example, an instance of our PartRibbon class get created. In the constructor, we load the Ribbon XML to configure the ribbon.

