How to create an Add-In extension .dll file and make it available from Robot pull down menu. (language C#)

August 28, 2018

1. Introduction

Robot Structure Analysis is equipped with appropriate interfaces that allow you to extend the functionality in a fairly simple way by including external components called directly from its pull down menu.

This document was created to help you quickly implement such an add-in extension by using Visual Studio C# project. The points below are in fact step by step instruction you should utilize to implement your own add-in extension and have possibility to call it from Robot Structure Analysis pull down menu.

2. Visual Studio C# template project

If you want to create add-in extension use project template attached to Robot Structure Analysis SDK (its name is MyAddin) or modify your own project basing on this project using the information contained in this short manual.

The sample project template attached to Robot Structure Analysis SDK is the complete add-in but in fact it is doing nothing except showing simple window. This is only example but by adding your own code into proper places you will be able to obtain what you intent.

3. References to RobotOM  (Robot Object Model) library

In your project set references to RobotOM library (Interop.RobotOM.dll in the case of C# project).

In the template project attached to RSA SDK references to RobotOM library are set to...\MyAddin\bin\Debug\Interop.RobotOM.dll
4. Project configuration

The following screen shots briefly summarize the necessary project settings. Make sure that your add-in assembly is COM visible.
5. Implementation of IRobotAddIn interface

Below you can see the simplest implementation of the IRobotAddIn interface. The DoCommand method will be executed after clicking on the appropriate pull down menu item. Therefore, it must contain instructions responsible for the appropriate functionalities of your add-in extension. The commands can be distinguished using the cmd_id parameter. By means of the InstallCommands method individual commands are added to the Robot Structure Analysis pull down menu. This is done using cmd_list parameter being RobotCmdList type.

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using RobotOM;

namespace MyAddin
{
    // ---------
    [System.Runtime.InteropServices.ComVisibleAttribute(true)]
    public class Class1 : IRobotAddIn
    {
        private IRobotApplication iapp = null;

        public bool Connect(RobotApplication robot_app, int add_in_id, bool first_time)
        {
            iapp = robot_app;
            return true;
        }

        public bool Disconnect()
        {
            iapp = null;
            return true;
        }

        public void DoCommand(int cmd_id)
        {
            //exemplary implementation

            // or execute any of your command for e.g. new Form1().Show();
        }

        public double GetExpectedVersion()
        {
            return 10;
        }

        public int InstallCommands(RobotCmdList cmd_list)
        {
            //exemplary implementation
            cmd_list.New(1, "My Command 1"); // Text in Robot menu
            return cmd_list.Count;
        }
    }
    // ---------
}
6. Setting new GUID for IRobotAddIn implementation and COM visible attribute for all forms

... COM visibility attribute must be set to FALSE for all forms in your add-in.
7. Add your own functionality code and build project

Implement DoCommand method with instructions responsible for the appropriate functionalities of your add-in extension.
Implement InstallCommands to add individual commands to the Robot Structure Analysis pull down menu.
Build project.

8. Preparing add-in .dll file to run with Robot Structure Analysis

- **Create .tlb file**
  Go to folder where your add-in .dll file is generated (\MyAddin\bin\Debug in our example)
  Run command:
  c:\Windows\Microsoft.NET\Framework64\v4.0.30319\regasm.exe /tlb /codebase MyAddin.dll
  or
  c:\Windows\Microsoft.NET\Framework64\v2.0.50727\RegAsm.exe /tlb /codebase MyAddin.dll

- **Add created .tlb library to add-in .dll file:**
  From the File menu \Open\File -> open created add-in .dll file (\MyAddin\bin\Debug\MyAddin.dll in our example) and add created .tlb library to .dll file (right hand mouse click menu). Resource type should be named as TYPELIB.
• **Change TYPELIB number**
  Change number to e.g. 1.0 using Properties (right hand mouse click menu)

![Image of Properties window]

• **Close Visual Studio and save changes to .dll file**

9. **Registration of created add-in .dll file to enable its visibility in Robot Structure Analysis**

Register add-in .dll file after its locating in target folder on any computer you want it to use as described below:

• **Open Command Prompt window as Admin**

![Image of Command Prompt window]
• Go to the folder where the add-in .dll file is located and register it by commands:
  c:\Windows\Microsoft.NET\Framework64\v4.0.30319\regasm.exe /tlb /codebase MyAddin.dll
  or
  c:\Windows\Microsoft.NET\Framework64\v2.0.50727\RegAsm.exe /tlb /codebase MyAddin.dll

9. Making new option available in Robot Structure Analysis pull down menu

  Start RSA, select any structure type, then from Add-ins menu start Add-ins Manage and using “…”
  button show path to add-in .dll file, then press Add button.
How to create an Add-In extension .dll file and make it available from Robot pull down menu. (language VB Net)
1. Visual Studio VBNet template project

Create a class library project:
Inside Visual Studio, on the File menu, click New Project. In the Installed Templates tab in the left-hand window, click Visual Basic. In the middle window, click Class Library.

Enter MyAddinVBNet in the Name box and enter D:\Add-in in Location box then click OK.

Visual Studio will create a default code project for you and display the code in the code window.
2. Add references:
In the Solution Explorer window on the right-hand side of the Visual Studio window, right-click References and click Add Reference...

Click the **Browse** tab and in the Add Reference dialog and browse to the Autodesk Robot Structural Analysis Professional product installation sub-folder. (The sub-folder path depends on where you have installed RSA 201x. The default path is `C:\Program Files\Autodesk\Autodesk Robot Structural Analysis Professional 201x\*`).

You will add reference file from this folder. Select **robotom.tlb**, and then click OK. Now the interface DLL file is referenced in your project. All the RSA APIs are exposed by these interface files and your project can use all of those available APIs from them.
3. Project configuration

The following screen shots briefly summarize the necessary project settings.
4. Implementation of IRobotAddIn interface

Below you can see the simplest implementation of the IRobotAddIn interface. The DoCommand method will be executed after clicking on the appropriate pull down menu item. Therefore, it must contain instructions responsible for the appropriate functionalities of your add-in extension. The commands can be distinguished using the cmd_id parameter. By means of the InstallCommands method individual commands are added to the Robot Structure Analysis pull down menu. This is done using cmd_list parameter being RobotCmdList type.

Imports RobotOM

<ComClass(Class1.ClassId)>
Public Class Class1
    Implements RobotOM.IRobotAddIn

    ' This GUID provides the COM identity for this class
    ' and its COM interfaces. If you change it, existing
    ' clients will no longer be able to access the class.
    Public Const ClassId As String = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

    Private robotApp As RobotOM.IRobotApplication

    ' A creatable COM class must have a Public Sub New()
    ' with no parameters, otherwise, the class will not be
    ' registered in the COM registry and cannot be created
    ' via CreateObject.
    Public Sub New()
        MyBase.New()
    End Sub

    Public Sub DoCommand(cmd_id As Integer) Implements IRobotAddIn.DoCommand
        Select Case cmd_id
        Case 1
            ' Clicking on first command in menu user gets message
            MsgBox("Message From AddinVBNet")
        Case 2
            ' Clicking on second command in menu user gets dialog
            Dim frm As New Form1()
            Dim value As Integer
            frm.Init(value)
            frm.ShowDialog()
        End Select
    End Sub

    Public Function Connect(robot_app As RobotApplication, add_in_id As Integer, first_time As Boolean) As Boolean Implements IRobotAddIn.Connect
        robotApp = robot_app
        Return True
    End Function

    Public Function Disconnect() As Boolean Implements IRobotAddIn.Disconnect
        robotApp = Nothing
        Return True
    End Function
Public Function GetExpectedVersion() As Double Implements IRobotAddIn.GetExpectedVersion
    Return 19.0
End Function

Public Function InstallCommands(cmd_list As RobotCmdList) As Integer Implements IRobotAddIn.InstallCommands
    cmd_list.[New](1, "Command 1 from AddinVBNet")
    cmd_list.[New](2, "Command 2 from AddinVBNet")
    Return cmd_list.Count
End Function
End Class

5. Setting new GUID for IRobotAddIn implementation

Replace xxxx.. with new GUID.

6. Preparing add-in .dll file to run with Robot Structure Analysis

- Create .tlb file
  Go to folder where your add-in .dll file is generated (...\MyAddinVBNet\bin\Debug in our example)
  Run command:
c:\Windows\Microsoft.NET\Framework64\v4.0.30319\regasm.exe /tlb /codebase MyAddinVBNet.dll
or
c:\Windows\Microsoft.NET\Framework64\v2.0.50727\RegAsm.exe /tlb /codebase MyAddinVBNet.dll

- **Add created .tlb library to add-in .dll file:**
  From the File menu > Open -> open created add-in .dll file 
  (...\MyAddinVBNet\bin\Debug\MyAddinVBNet.dll in our example) and add created .tlb library to .dll file (right hand mouse click menu).
  Resource type should be named as TYPELIB.

- **Change TYPELIB number**
  Change number to e.g 1.0 using Properties (right hand mouse click menu)
• Close Visual Studio and save changes to .dll file

7. Registration of created add-in .dll file to enable its visibility in Robot Structure Analysis

Register add-in .dll file after its locating in target folder on any computer you want it to use as described below:

• Open Command Prompt window as Admin

• Go to the folder where the add-in .dll file is located and register it by commands:
8. Making new option available in Robot Structure Analysis pull down menu

Start RSA, select any structure type, then from Add-ins menu start Add-ins Manage and using "..." button show path to add-in .dll file, then press Add button.
... after closing manager the new option should be available in robot menu as it is showed above.