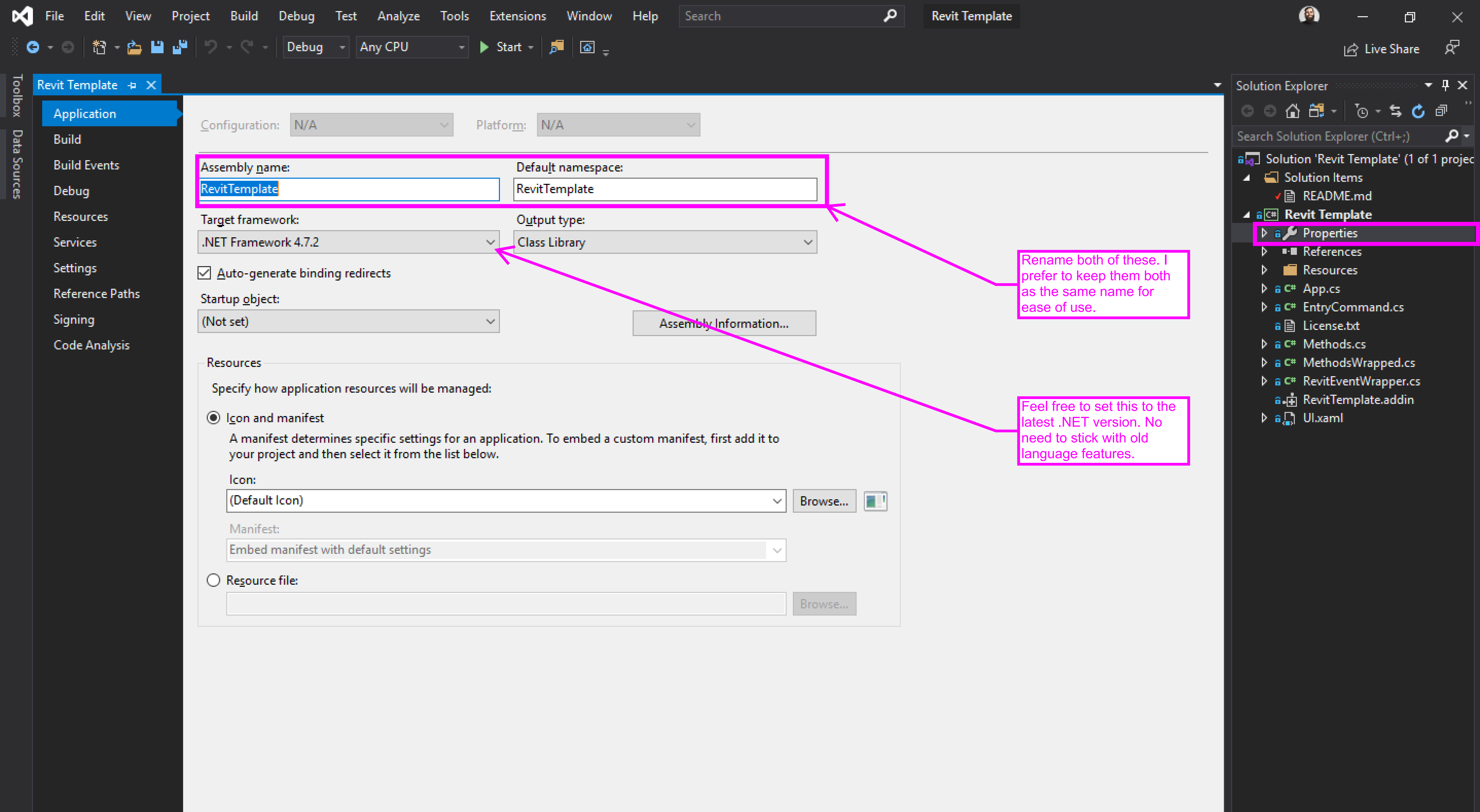


Configuration: N/A Platform: N/A

Assembly name: RevitTemplate
Default namespace: RevitTemplate

Target framework: .NET Framework 4.7.2
Output type: Class Library

☑ Auto-generate binding redirects

Startup object: (Not set)

Resources

Specify how application resources will be managed:

● Icon and manifest

A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.

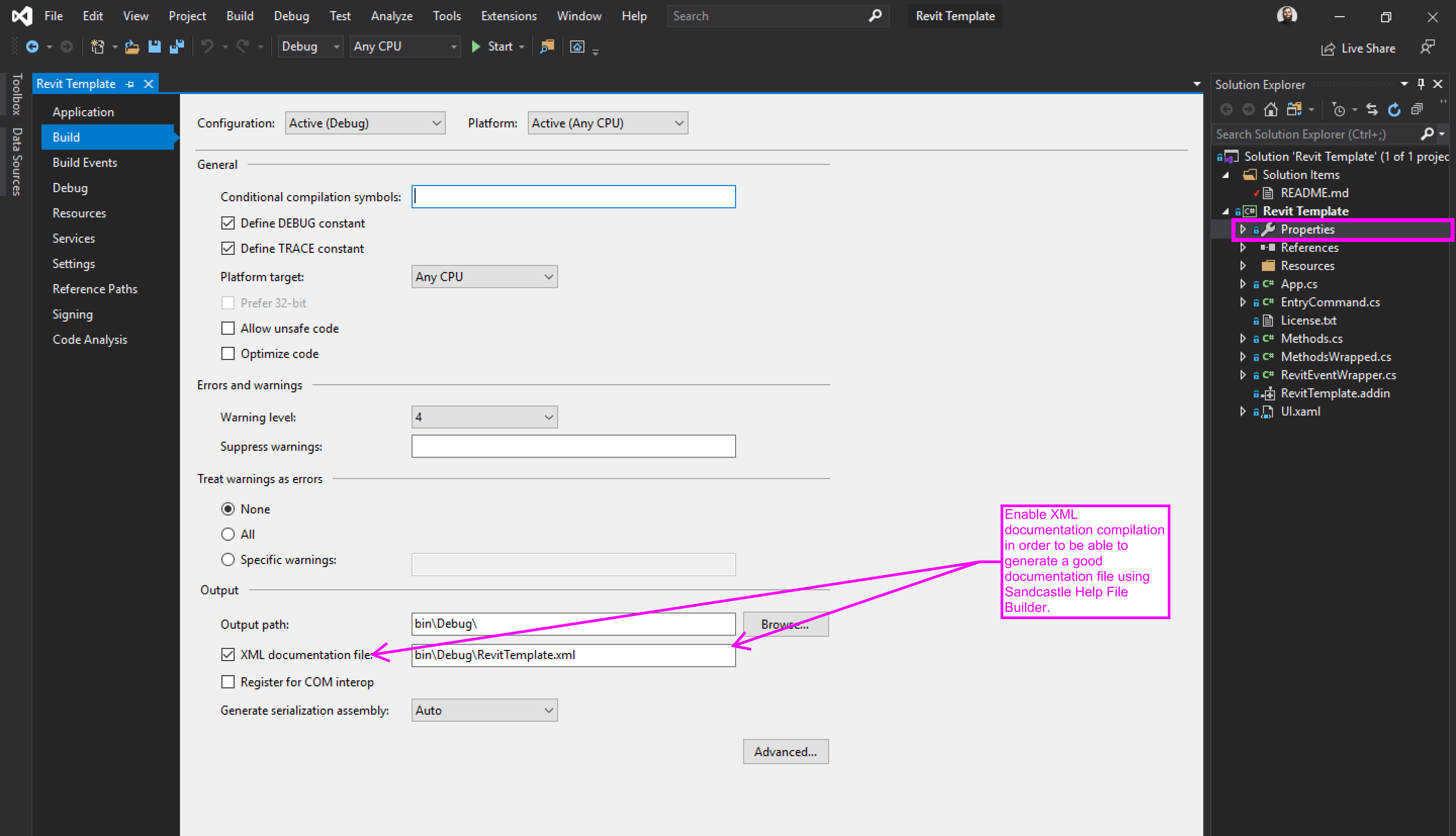
Icon: (Default Icon)

Manifest: Embed manifest with default settings

○ Resource file:

Rename both of these. I prefer to keep them both as the same name for ease of use.

Feel free to set this to the latest .NET version. No need to stick with old language features.



Configuration: Active (Debug)   Platform: Active (Any CPU)

General

Conditional compilation symbols:

- [✓] Define DEBUG constant
- [✓] Define TRACE constant

Platform target: Any CPU

- [ ] Prefer 32-bit
- [ ] Allow unsafe code
- [ ] Optimize code

Errors and warnings

Warning level: 4

Suppress warnings:

Treat warnings as errors

- [✓ None]
- All
- Specific warnings:

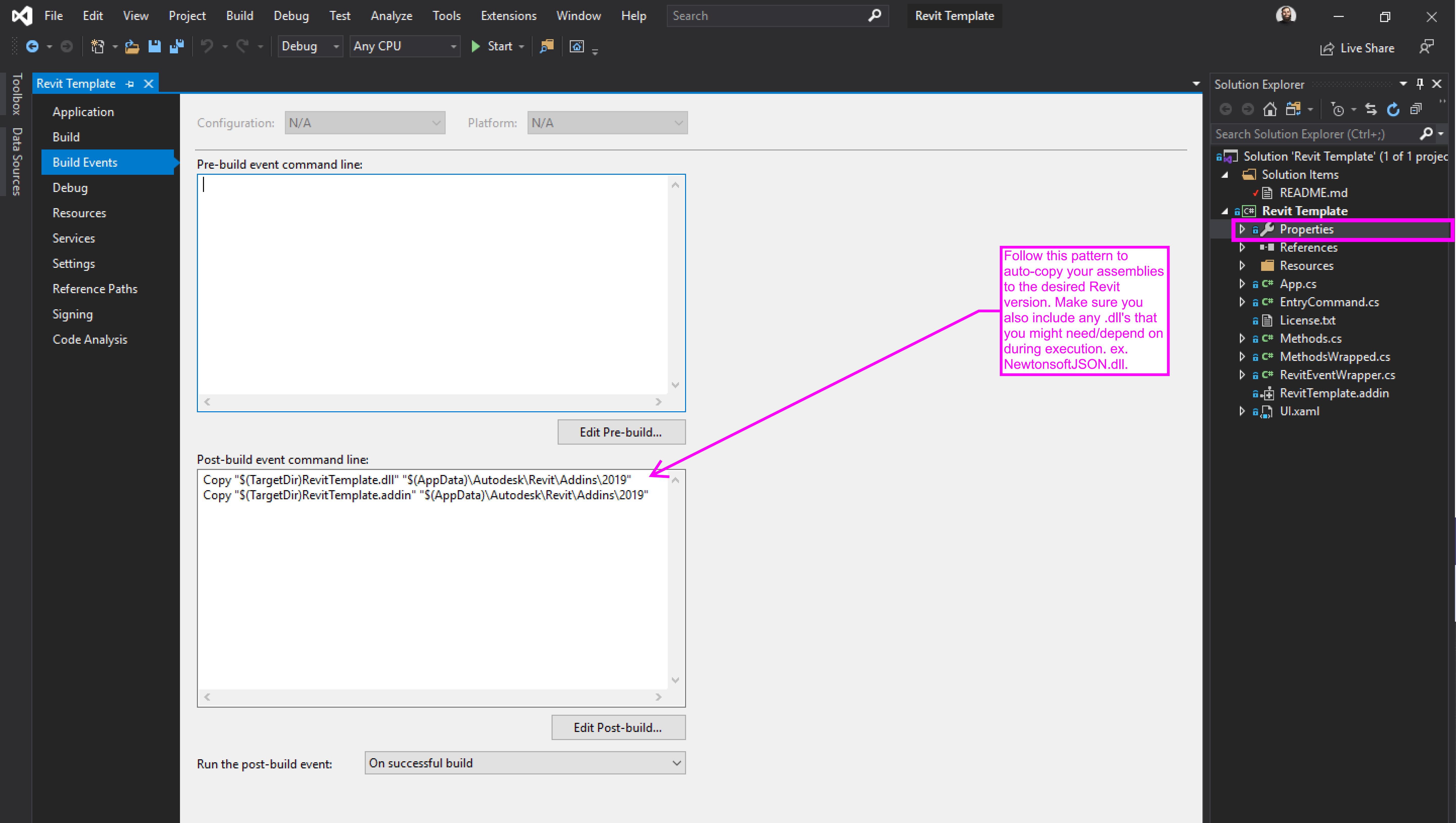
Output

Output path: bin\Debug\

- [✓] XML documentation file: bin\Debug\RevitTemplate.xml
- [ ] Register for COM interop

Generate serialization assembly: Auto

Enable XML documentation compilation in order to be able to generate a good documentation file using Sandcastle Help File Builder.



Configuration: N/A Platform: N/A

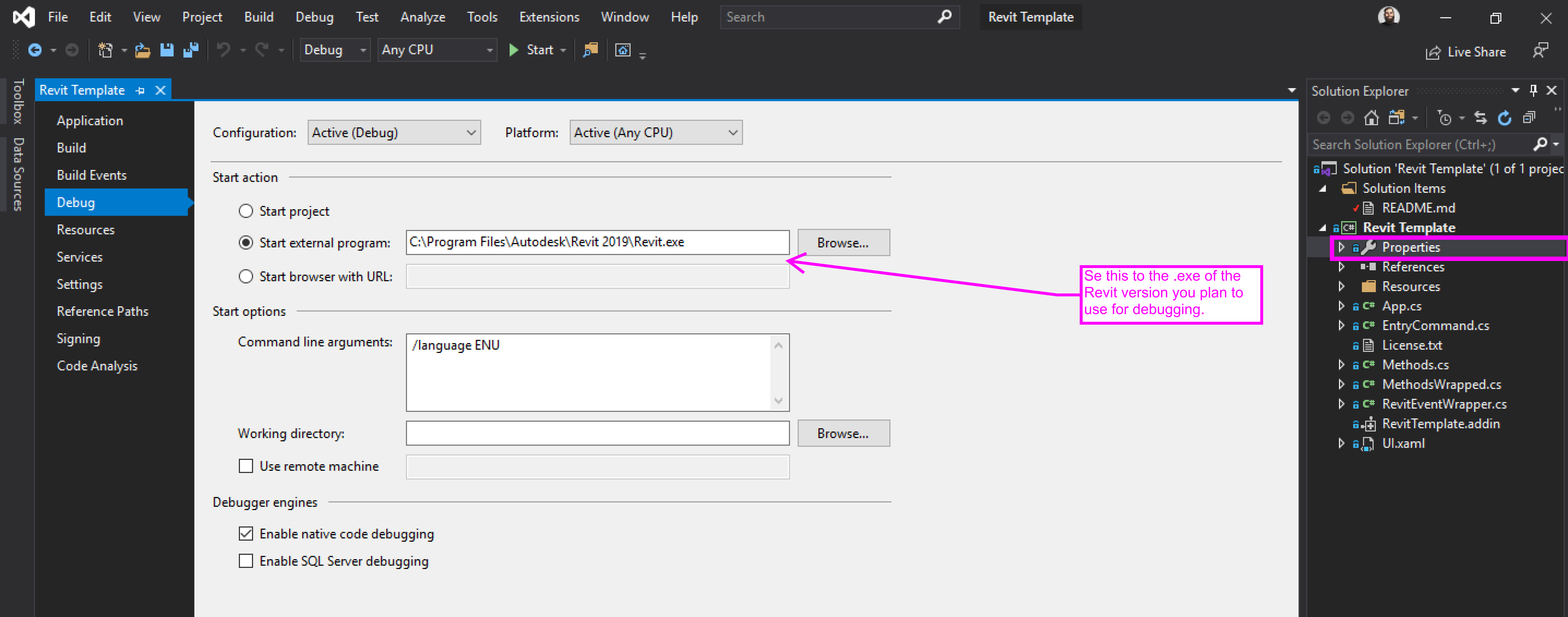
Pre-build event command line:

Post-build event command line:

```
Copy "$(TargetDir)RevitTemplate.dll" "$(AppData)\Autodesk\Revit\Addins\2019"
Copy "$(TargetDir)RevitTemplate.addin" "$(AppData)\Autodesk\Revit\Addins\2019"
```

Run the post-build event: On successful build

Follow this pattern to auto-copy your assemblies to the desired Revit version. Make sure you also include any .dll's that you might need/depend on during execution. ex. NewtonsoftJSON.dll.



Configuration: Active (Debug)   Platform: Active (Any CPU)

Start action

( ) Start project
(●) Start external program: C:\Program Files\Autodesk\Revit 2019\Revit.exe
( ) Start browser with URL:

Start options

Command line arguments: /language ENU

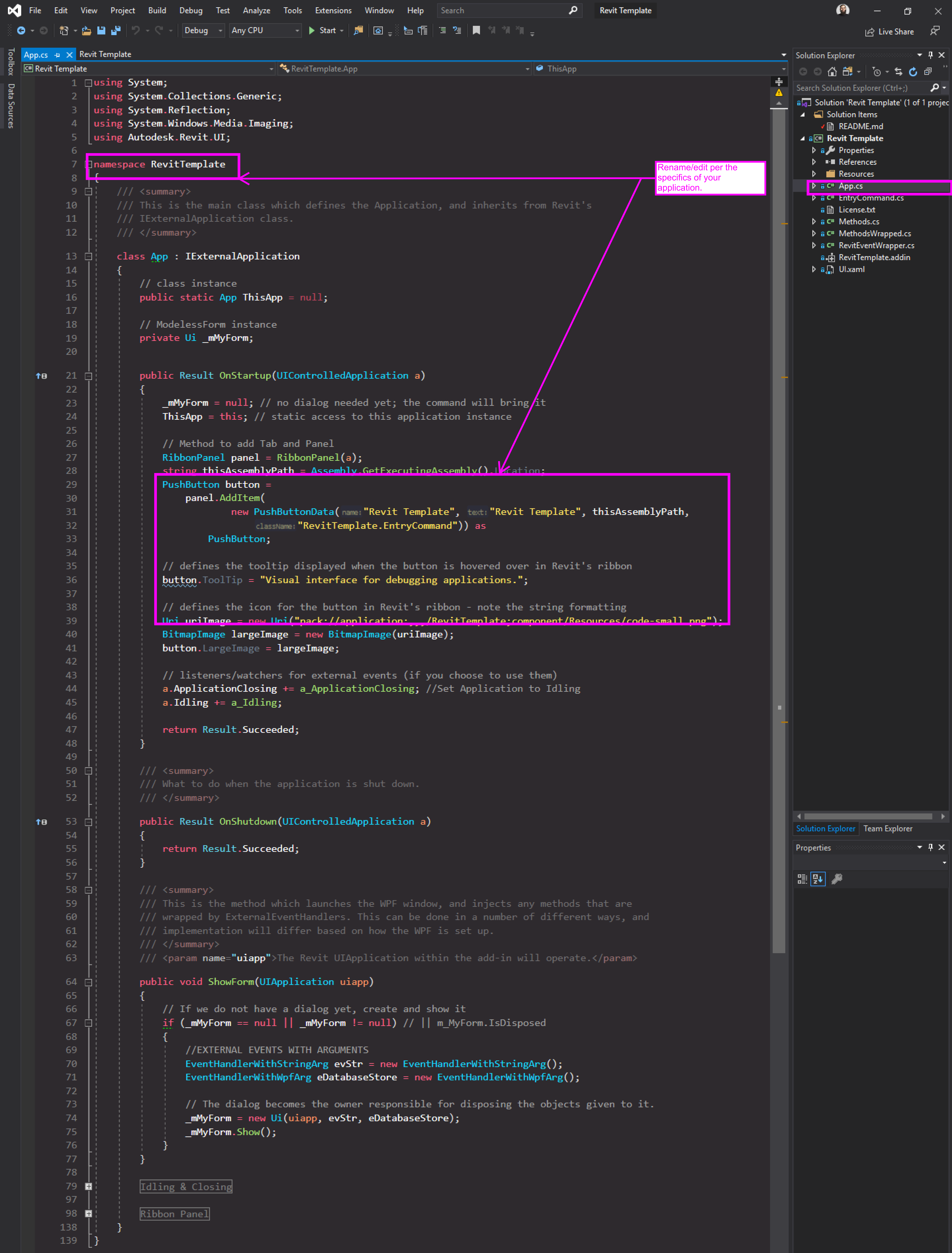
Working directory:

☐ Use remote machine

Debugger engines

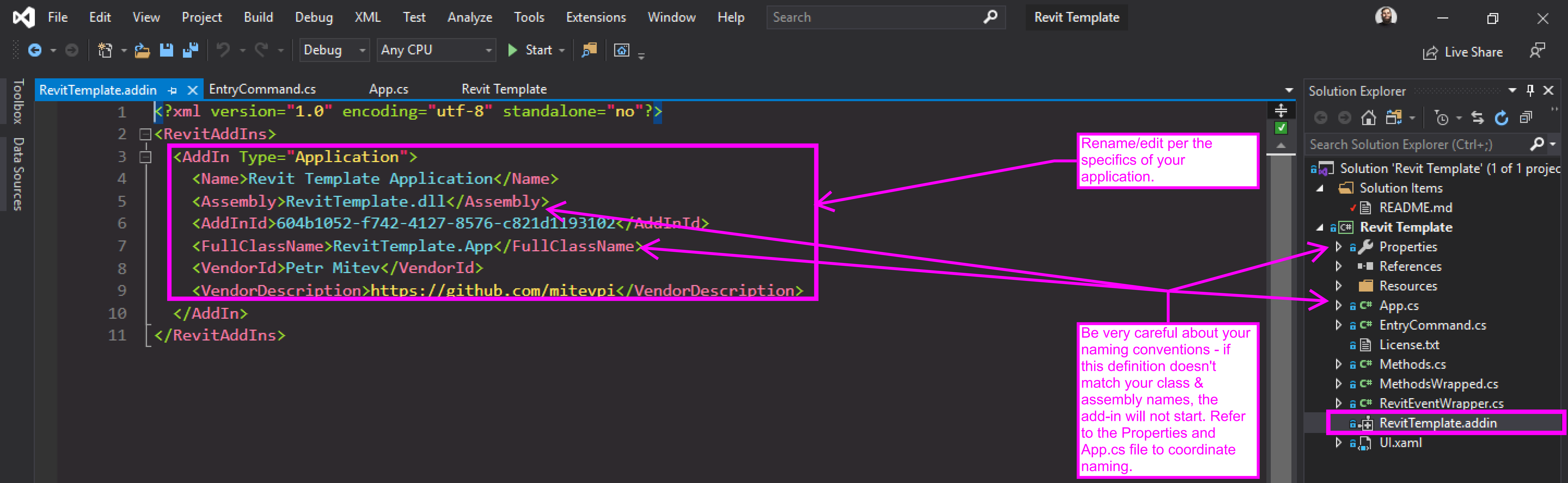
☑ Enable native code debugging
☐ Enable SQL Server debugging

Se this to the .exe of the Revit version you plan to use for debugging.



Rename/edit per the specifics of your application.

```csharp
using System;
using System.Collections.Generic;
using System.Reflection;
using System.Windows.Media.Imaging;
using Autodesk.Revit.UI;

namespace RevitTemplate
{
    /// <summary>
    /// This is the main class which defines the Application, and inherits from Revit's
    /// IExternalApplication class.
    /// </summary>

    class App : IExternalApplication
    {
        // class instance
        public static App ThisApp = null;

        // ModelessForm instance
        private Ui _mMyForm;


        public Result OnStartup(UIControlledApplication a)
        {
            _mMyForm = null; // no dialog needed yet; the command will bring it
            ThisApp = this; // static access to this application instance

            // Method to add Tab and Panel
            RibbonPanel panel = RibbonPanel(a);
            string thisAssemblyPath = Assembly.GetExecutingAssembly().Location;
            PushButton button =
                panel.AddItem(
                        new PushButtonData( name: "Revit Template",  text: "Revit Template", thisAssemblyPath,
                            className: "RevitTemplate.EntryCommand")) as
                    PushButton;

            // defines the tooltip displayed when the button is hovered over in Revit's ribbon
            button.ToolTip = "Visual interface for debugging applications.";

            // defines the icon for the button in Revit's ribbon - note the string formatting
            Uri uriImage = new Uri("pack://application:,,,/RevitTemplate;component/Resources/code-small.png");
            BitmapImage largeImage = new BitmapImage(uriImage);
            button.LargeImage = largeImage;

            // listeners/watchers for external events (if you choose to use them)
            a.ApplicationClosing += a_ApplicationClosing; //Set Application to Idling
            a.Idling += a_Idling;

            return Result.Succeeded;
        }

        /// <summary>
        /// What to do when the application is shut down.
        /// </summary>

        public Result OnShutdown(UIControlledApplication a)
        {
            return Result.Succeeded;
        }

        /// <summary>
        /// This is the method which launches the WPF window, and injects any methods that are
        /// wrapped by ExternalEventHandlers. This can be done in a number of different ways, and
        /// implementation will differ based on how the WPF is set up.
        /// </summary>
        /// <param name="uiapp">The Revit UIApplication within the add-in will operate.</param>

        public void ShowForm(UIApplication uiapp)
        {
            // If we do not have a dialog yet, create and show it
            if (_mMyForm == null || _mMyForm != null) // || m_MyForm.IsDisposed
            {
                //EXTERNAL EVENTS WITH ARGUMENTS
                EventHandlerWithStringArg evStr = new EventHandlerWithStringArg();
                EventHandlerWithWpfArg eDatabaseStore = new EventHandlerWithWpfArg();

                // The dialog becomes the owner responsible for disposing the objects given to it.
                _mMyForm = new Ui(uiapp, evStr, eDatabaseStore);
                _mMyForm.Show();
            }
        }

        Idling & Closing

        Ribbon Panel
    }
}
```



```xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<RevitAddIns>
  <AddIn Type="Application">
    <Name>Revit Template Application</Name>
    <Assembly>RevitTemplate.dll</Assembly>
    <AddInId>604b1052-f742-4127-8576-c821d1193102</AddInId>
    <FullClassName>RevitTemplate.App</FullClassName>
    <VendorId>Petr Mitev</VendorId>
    <VendorDescription>https://github.com/mitevpi</VendorDescription>
  </AddIn>
</RevitAddIns>
```

Rename/edit per the specifics of your application.

Be very careful about your naming conventions - if this definition doesn't match your class & assembly names, the add-in will not start. Refer to the Properties and App.cs file to coordinate naming.