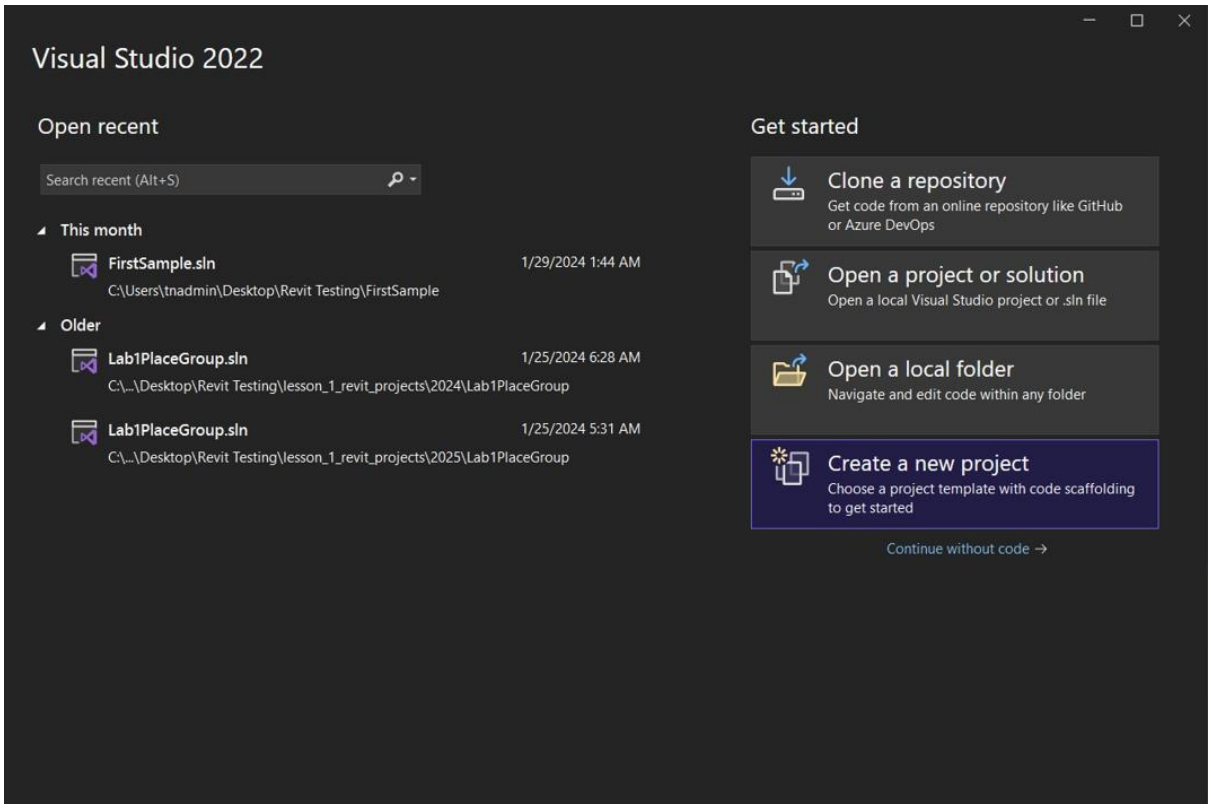
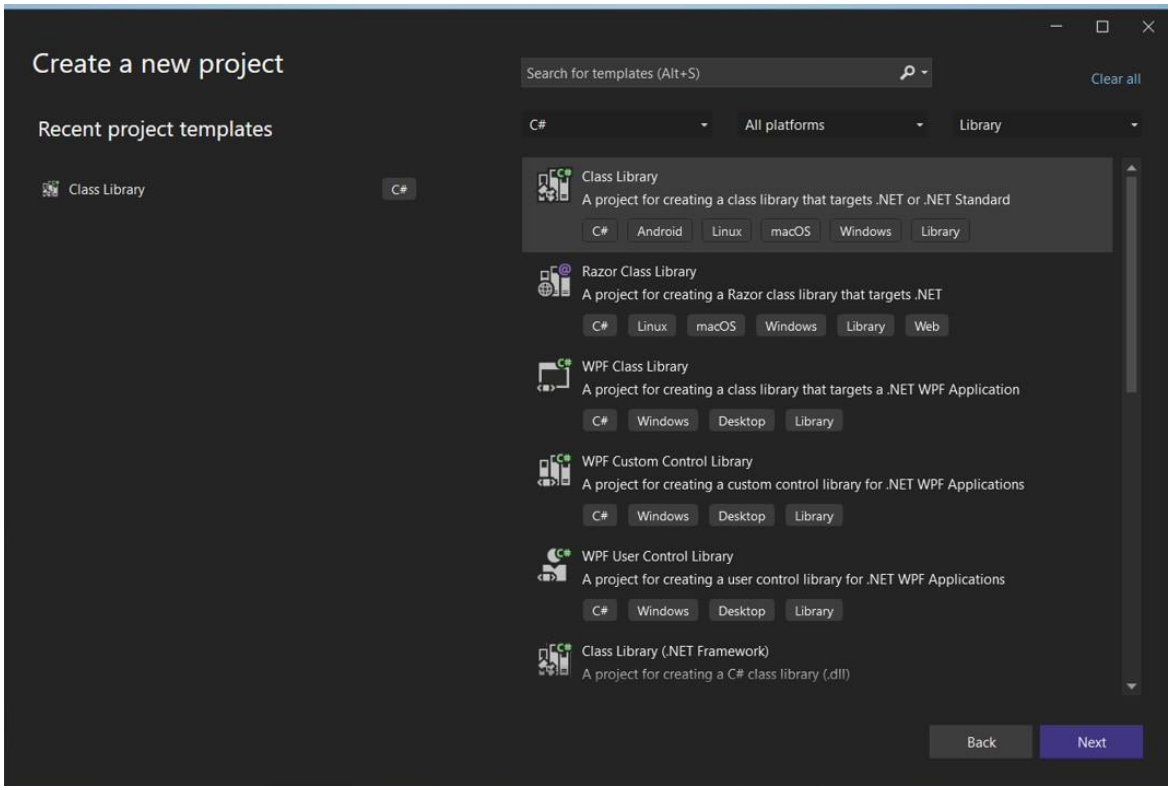
a. Open Visual Studio.
b. Navigate to the "Get started" section. Click on "Create a new project".

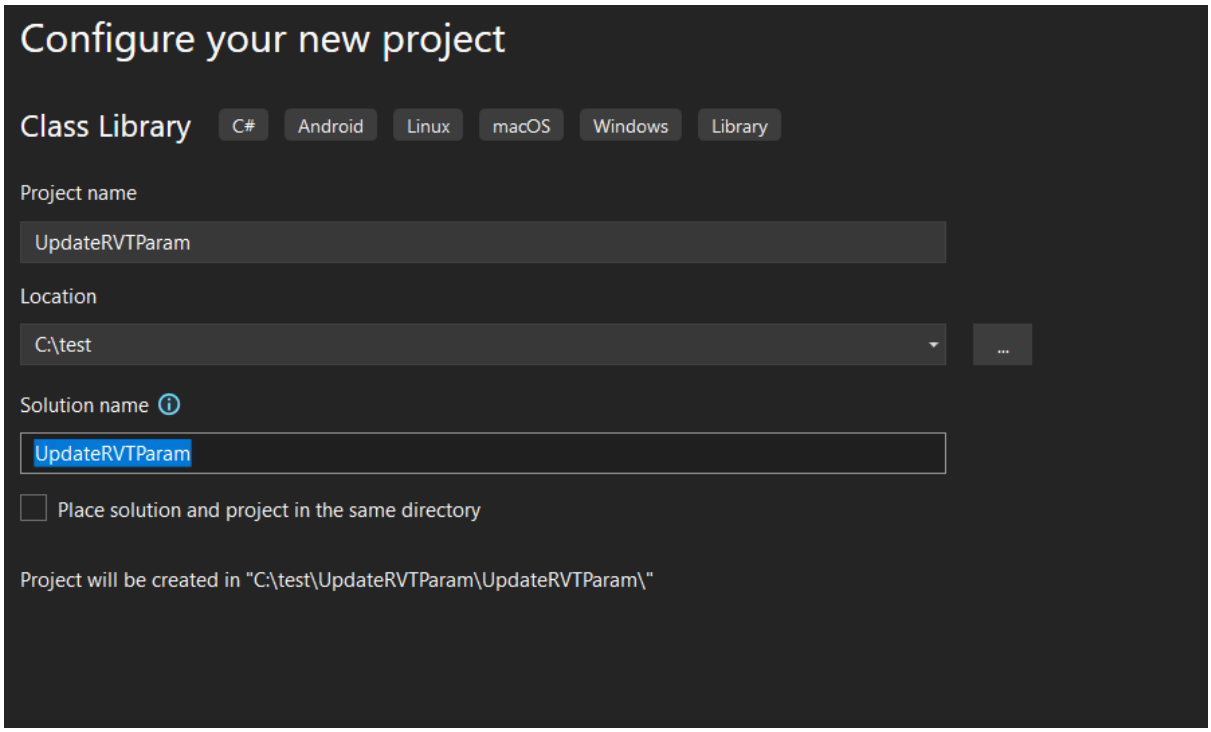


c. Select "C#" from languages.
d. Choose "All platforms".
e. Select "Library" as the project type.



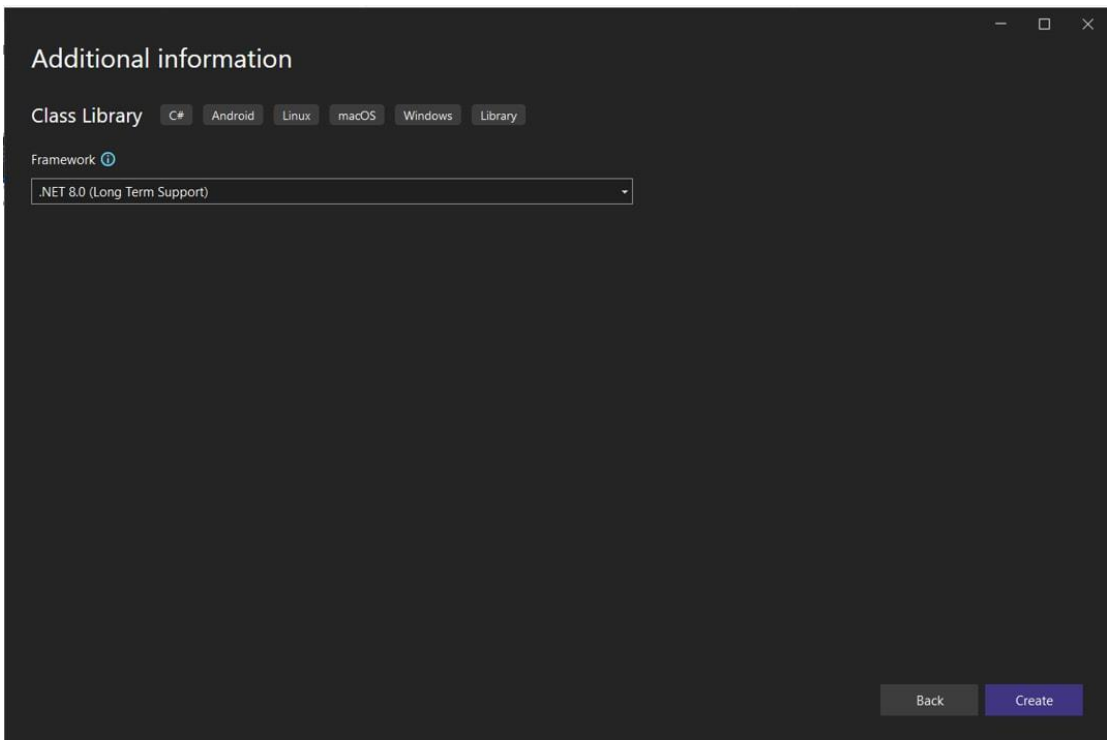
f. Now, choose "Class Library – A project for creating a class library that targets .NET or .NET Standard" from the available options. Click on the "Next" button.

g. Enter "UpdateRVTParam" as the Project Name.



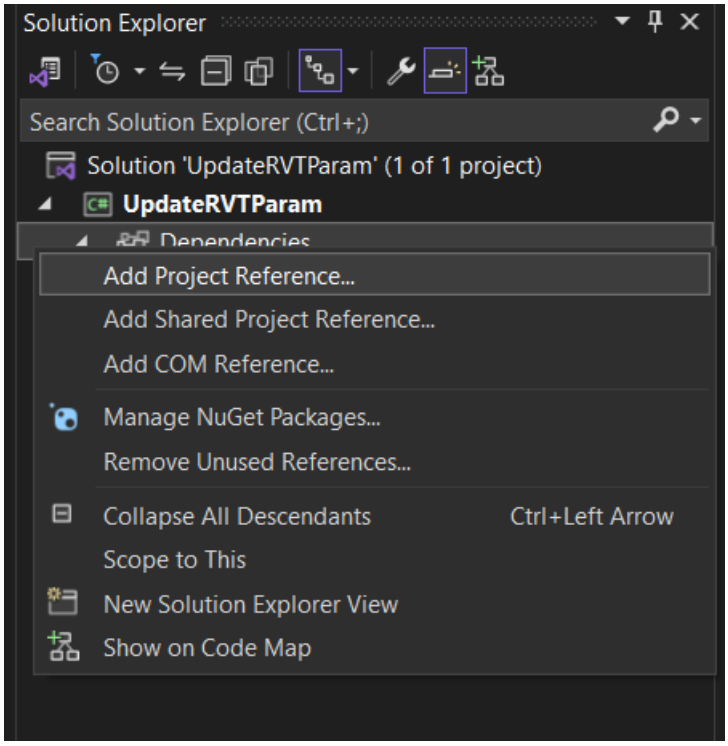
i. Click on the "Next" button.

j. In the "Framework" section, choose ".NET 8.0 (Long Term Support)". Finally, click on the "Create" button to create the project.

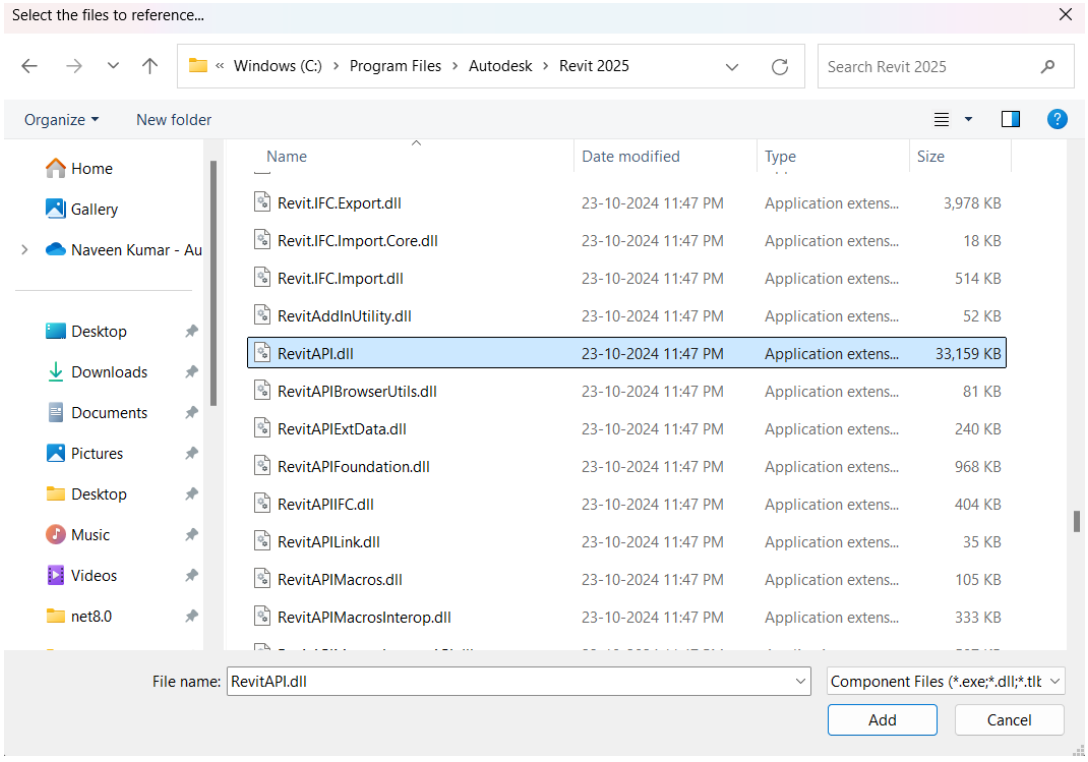


k. In the "View" menu, you'll see "Solution Explorer". Click on it. This action should open the Solution Explorer window.

Within the Solution Explorer, right-click "Dependencies" and select "Add Project Reference...".



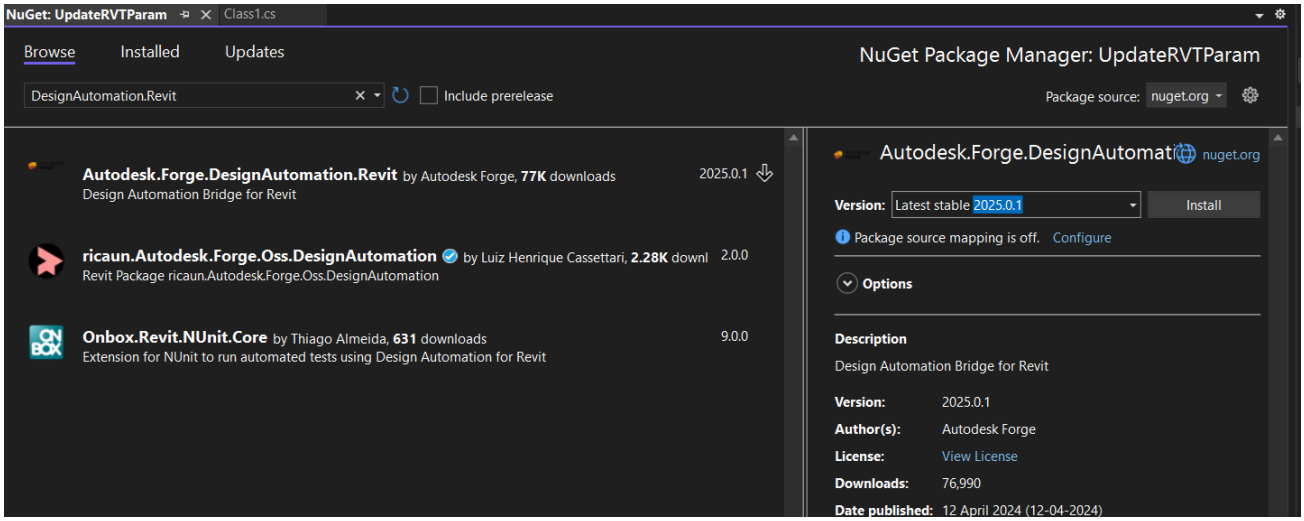
l. Click the Browse tab and in the Add Reference dialog and browse to the Revit product installation sub-folder. (The sub-folder path depends on where you have installed Revit 2025. The default path is C:\Program Files\Autodesk\Revit 2025\).



You will add one reference files from this folder. Select RevitAPI.dll and then click Add. In the Reference Manager Dialog, Click OK.

m. In the Solution Explorer window, right click RevitAPI under Assemblies node and choose properties. In the Properties window, click Copy Local property, and then click the drop-down list, select No.

n. In the Solution Explorer window, right-click on the project (UpdateRVTParam), go to **Manage NuGet Packages...** Under **Browse**, search for *DesignAutomation.Revit* and install *Autodesk.Forge.DesignAutomation.Revit* (choose the appropriate Revit version needed). At the time of creating this tutorial, the latest version available for Revit 2025 is "2025.0.1". Then, search for and install *Newtonsoft.Json* (which is used to parse input data in JSON format).



o. The project should contain a Class1.cs class, let's rename the file to Commands.cs (for consistency).

p. Now add the below code to **Commands.cs**

```
using Autodesk.Revit.ApplicationServices;
using Autodesk.Revit.Attributes;
using Autodesk.Revit.DB;
using DesignAutomationFramework;
using Newtonsoft.Json;
namespace Autodesk.Forge.Sample.DesignAutomation.Revit
{
    [Transaction(TransactionMode.Manual)]
    [Regeneration(RegenerationOption.Manual)]
    public class Commands : IExternalDBApplication
    {
        //Path of the project(i.e)project where your Window family files
are present
        string OUTPUT_FILE = "OutputFile.rvt";

        public ExternalDBApplicationResult
OnShutdown(ControlledApplication application)
        {
            return ExternalDBApplicationResult.Succeeded;
```

```
        }

        public ExternalDBApplicationResult OnStartup(ControlledApplication
application)
        {
            DesignAutomationBridge.DesignAutomationReadyEvent +=
HandleDesignAutomationReadyEvent;
            return ExternalDBApplicationResult.Succeeded;
        }
        private void HandleDesignAutomationReadyEvent(object? sender,
DesignAutomationReadyEventArgs e)
        {
            LogTrace("Design Automation Ready event triggered...");
            e.Succeeded = true;
            EditWindowParametersMethod(e.DesignAutomationData.RevitDoc);
        }

        private void EditWindowParametersMethod(Document doc)
        {
            // Deserialize input parameters from JSON file
            InputParams? inputParameters =
JsonConvert.DeserializeObject<InputParams>(File.ReadAllText("params.json")
);
            using (Transaction trans = new Transaction(doc))
            {
                trans.Start("Update window parameters");

                FilteredElementCollector windowCollector = new
FilteredElementCollector(doc).OfCategory(BuiltInCategory.OST_Windows).Wher
eElementIsNotElementType();
                ICollection<ElementId> windowIds =
windowCollector.ToElementIds();

                // Check if input parameters are valid and if any windows
are found
                if (inputParameters != null && windowIds.Count > 0)
                {
                    foreach (ElementId windowId in windowIds)
                    {

                        Element? window = doc.GetElement(windowId);
                        if (window is FamilyInstance famInst)
                        {
                            FamilySymbol? famSym = famInst.Symbol;

                            // Set the height and width parameters for the
window family symbol
                            SetElementParameter(famSym,
BuiltInParameter.WINDOW_HEIGHT, inputParameters.Height);
                            SetElementParameter(famSym,
BuiltInParameter.WINDOW_WIDTH, inputParameters.Width);
                        }
                    }
                }
```

```
                else
                {
                    // Log a message if no input parameters or windows are
found
                    if (inputParameters == null)
                        LogTrace("Error: Input parameters are null or
failed to deserialize.");
                    if (windowIds.Count == 0)
                        LogTrace("Warning: No windows found in the
document.");
                }

                // Commit the transaction to save all changes made to the
document
                trans.Commit();
            }
            //Save the updated file by overwriting the existing file
            ModelPath ProjectModelPath =
ModelPathUtils.ConvertUserVisiblePathToModelPath(OUTPUT_FILE);
            SaveAsOptions SAO = new SaveAsOptions();
            SAO.OverwriteExistingFile = true;

            //Save the project file with updated window's parameters
            LogTrace("Saving file...");
            doc.SaveAs(ProjectModelPath, SAO);
        }
        private void SetElementParameter(FamilySymbol FamSym,
BuiltInParameter param, double parameterValue)
        {
            FamSym.get_Parameter(param).Set(parameterValue);
        }
        public class InputParams
        {
            public double Width { get; set; }
            public double Height { get; set; }
        }
        private static void LogTrace(string format, params object[] args)
{ System.Console.WriteLine(format, args); }
    }
}
```

q. Inside Visual Studio, in the Build menu, click Build Solution to compile and build your plug-in. Build Success message shows in status bar of the Visual Studio window if the code is successfully built.

r. After building the solution, you will encounter numerous warnings displayed in the output window located at the bottom of the Visual Studio interface.

To suppress these warnings, follow these steps:

1. Navigate to the Solution Explorer.
2. Right-click on "UpdateRVTParam" and select "Properties."
3. In the properties window, under the "Build" tab, navigate to "General."
4. Change the "Platform target" to "x64."

5. Still in the "Build" tab, navigate to "Errors and Warnings."
6. Add ";MSB3277" to the existing list to suppress warnings effectively.

Proceed to build the solution as instructed in step “q”.

PackageContents.xml

In Solution Explorer, right-click "UpdateRVTParam," select "Add," and create a folder named “UpdateRVTParam.bundle”. Inside this folder, create a file named PackageContents.xml and copy the following content into it. Learn more at the **PackageContents.xml Format Reference**. This file instructs Revit to load our .addin plugin.

```
<?xml version="1.0" encoding="utf-8" ?>
<ApplicationPackage Name="RevitDesignAutomation" Description="Sample
Plugin for Revit" Author="tutorials.autodesk.io">
<CompanyDetails Name="Autodesk, Inc" Url="http://tutorials.autodesk.io"
Email="forge.help@autodesk.com"/>
<Components Description="Modify window parameters">
<RuntimeRequirements SeriesMax="R2025" SeriesMin="R2024" Platform="Revit"
OS="Win64"/>
<ComponentEntry LoadOnRevitStartup="True" LoadOnCommandInvocation="False"
AppDescription="Modify Window Parameters"
ModuleName="./Contents/Autodesk.Forge.Sample.DesignAutomation.Revit.addin"
Version="1.0.0" AppName="Modify Window Parameters"/>
</Components>
</ApplicationPackage>
```

Autodesk.Forge.Sample.DesignAutomation.Revit.addin

Under the UpdateRVTParam.bundle folder, create a subfolder named Contents, and inside this folder, create a new file called Autodesk.Forge.Sample.DesignAutomation.Revit.addin. This file tells Revit how to load the plugin. Copy the following content into the addin file.

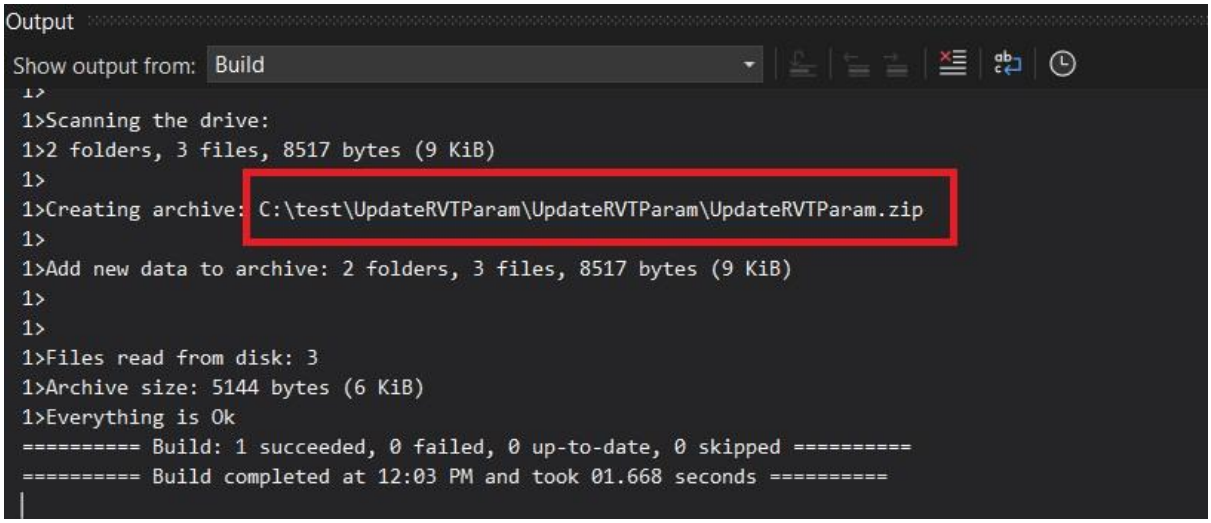
```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<RevitAddIns>
  <AddIn Type="DBApplication">
    <Name>Modify Window Parameters</Name>
<FullClassName>Autodesk.Forge.Sample.DesignAutomation.Revit.Commands</Full
ClassName>
    <Text>Revit for Design Automation</Text>
    <Description>Revit for Design Automation</Description>
    <VisibilityMode>AlwaysVisible</VisibilityMode>
    <Assembly>.\UpdateRVTParam.dll</Assembly>
    <AddInId>000BD853-36E4-461f-9171-C5ACEDA4E723</AddInId>
    <VendorId>ADSK</VendorId>
    <VendorDescription>Autodesk, Inc, www.autodesk.com</VendorDescription>
  </AddIn>
</RevitAddIns>
```

Create Appbundle Zip file

In **Solution Explorer**, right-click on "UpdateRVTParam" and select **Properties**. In the properties window, go to the **Build** tab and navigate to **Events**. Copy the following commands and paste them into the **Post-build event** field:

```
xcopy /Y /F "$(TargetDir)*.dll"
"$(ProjectDir)UpdateRVTParam.bundle\Contents\"
del /F
"$(ProjectDir)..\designAutomationSample\wwwroot\bundles\UpdateRVTParam.zip
"
"C:\Program Files\7-Zip\7z.exe" a -tzip "$(ProjectDir)UpdateRVTParam.zip"
"$(ProjectDir)UpdateRVTParam.bundle\" -xr0!*.pdb
```

Now, build the solution as outlined in Step q. This will generate the app bundle as a zip file.



In the root directory, create a folder named "bundles" and place the generated zip file inside it to ensure accessibility for Design Automation.