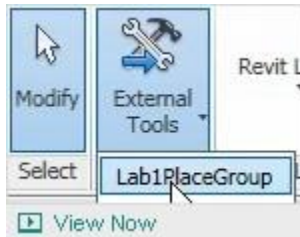


Lesson 1: The Basic Plug-in

In this lesson you will create your very first basic Autodesk Revit plug-in for copying groups selected by the user to a specified location.




Video: A Demonstration of Lesson 1 Steps to Create your First Plug-in

Provide Feedback: Please provide feedback about this Revit Training or this lesson via email: myfirstplugin@autodesk.com

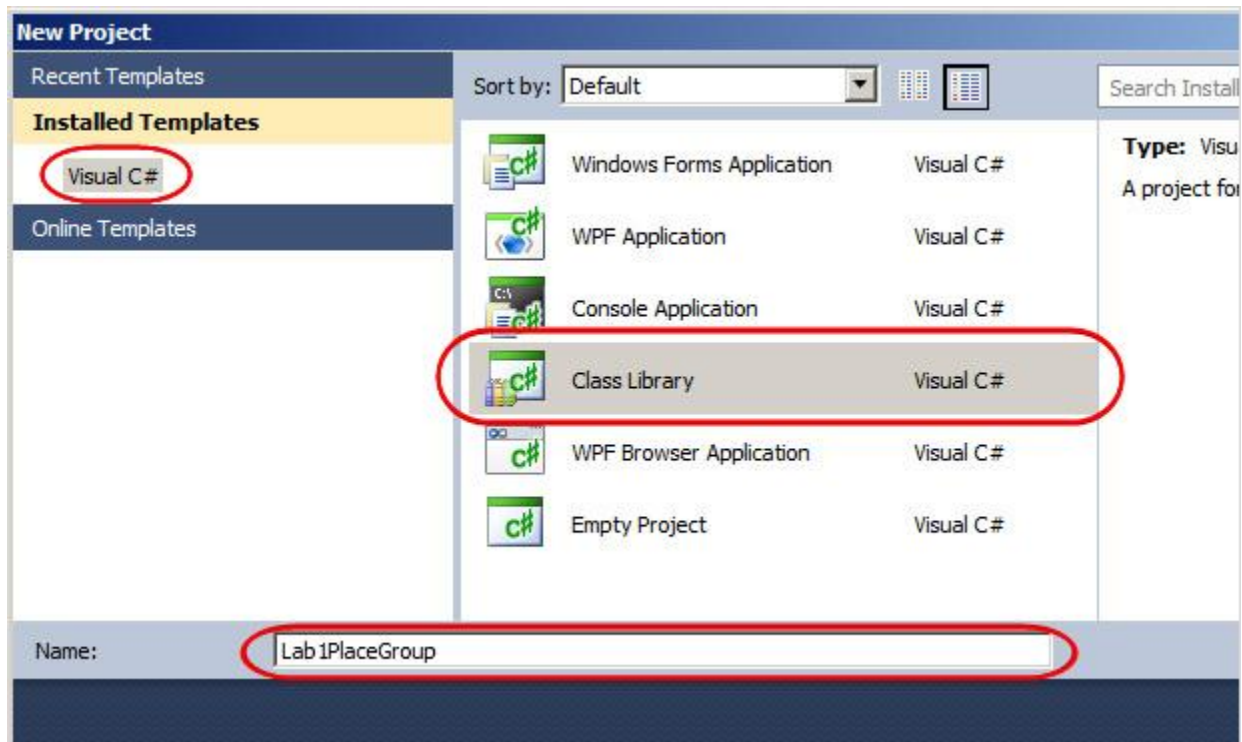
Lesson Downloads

 [lesson1_revit_2013_projects.zip](#) (zip - 28884Kb)

 [lesson1_revit_2012_and_earlier_project_files.zip](#) (zip - 7283Kb)

Steps to Create Your First Plug-in

1. **Launch the Visual C# Express development environment:**
Open Visual C# 2010 Express using the Windows **Start** menu, selecting **All Programs**, and then **Microsoft Visual Studio 2010 Express** and then the **Microsoft Visual C# 2010 Express** submenu-item.
2. **Create a class library project:**
Inside Visual C# Express, on the **File** menu, click **New Project**. In the **Installed Templates** tab in the left-hand window, click **Visual C#**. In the middle window, click **Class Library**. Enter **Lab1PlaceGroup** in the **Name** box. And then click **OK**.

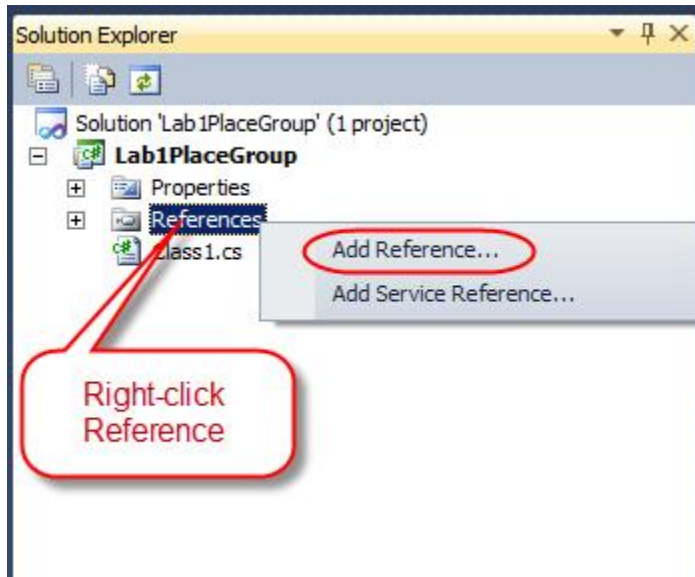


Visual C# Express will create a default code project for you and display the code in the code window.

3. **Save the project:**
On the **File** menu, click **Save All**. In the display window type **C:\test** in the **Location** box, and then click **Save**.

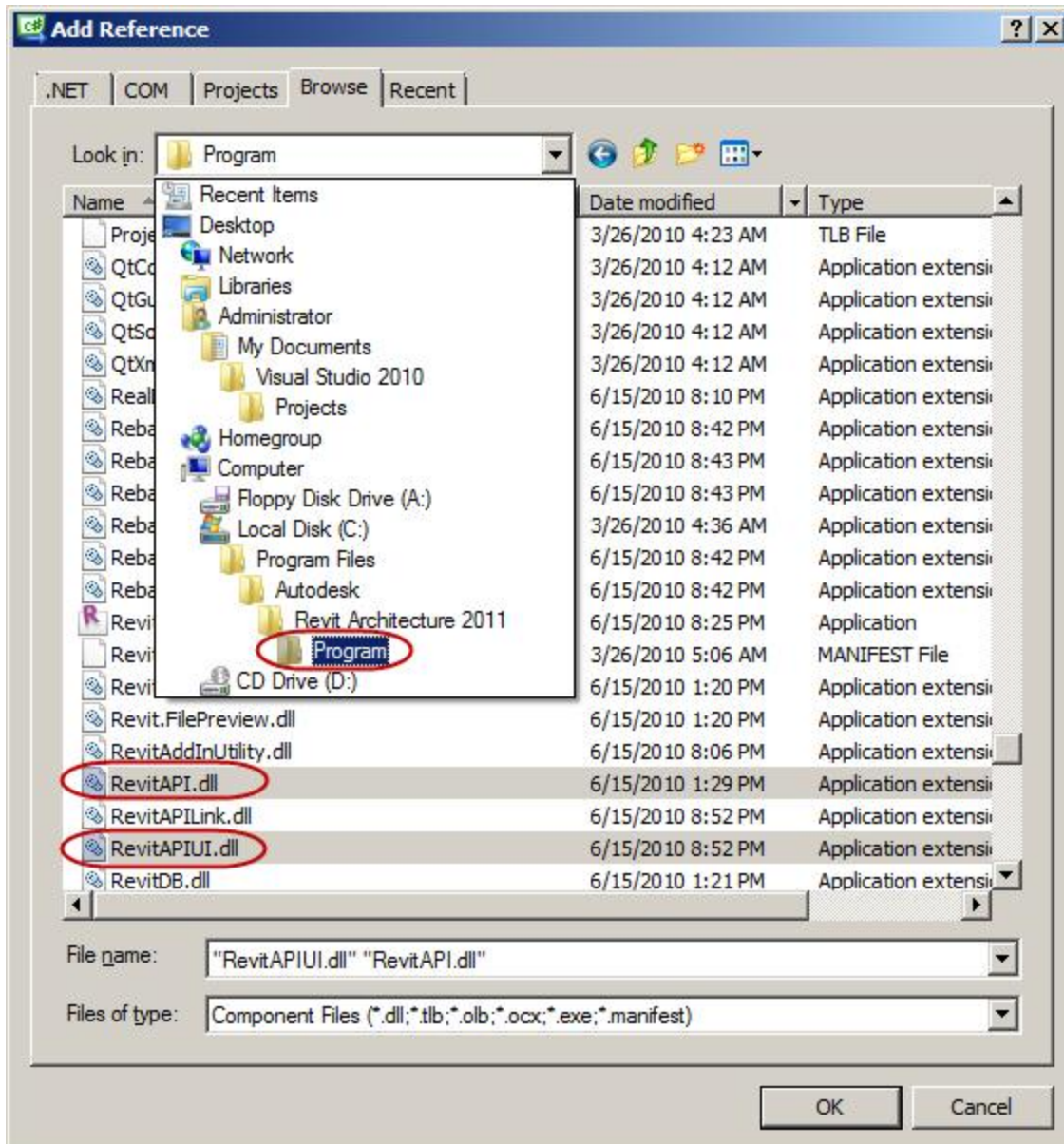
4. **Add references:**

In the **Solution Explorer** window on the right-hand side of the Visual C# Express window, right-click **References** and click **Add Reference...**



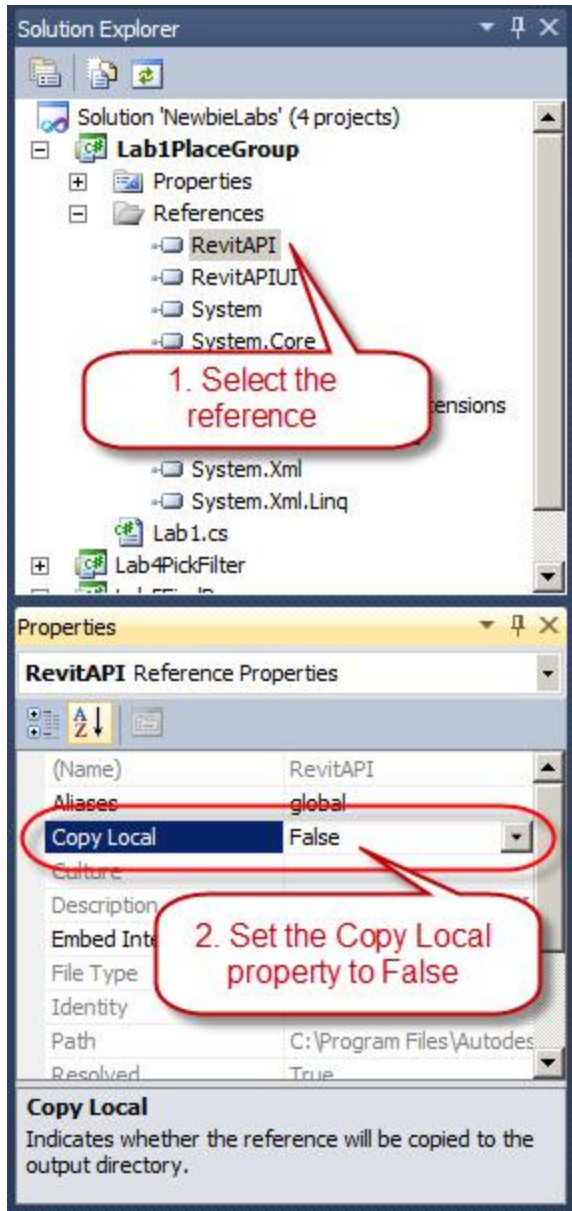
5. Click the **Browse** tab in the **Add Reference** dialog and browse to the Revit product installation sub-folder. (The sub-folder path depends on where you have installed Revit Architecture 201x. The default path is *C:\Program Files\Autodesk\Revit Architecture 201x\Program**).

** The path may vary depending on the flavour of Autodesk Revit you are using.*



You will add two reference files from this folder. Select **RevitAPI.dll**, hold the Ctrl key and select **RevitAPIUI.dll**, and then click **OK**. Now the two interface DLL files are referenced in your project. All the Revit APIs are exposed by these interface files and your project can use all of those available APIs from them.

6. Set the referenced files' Copy Local property value:

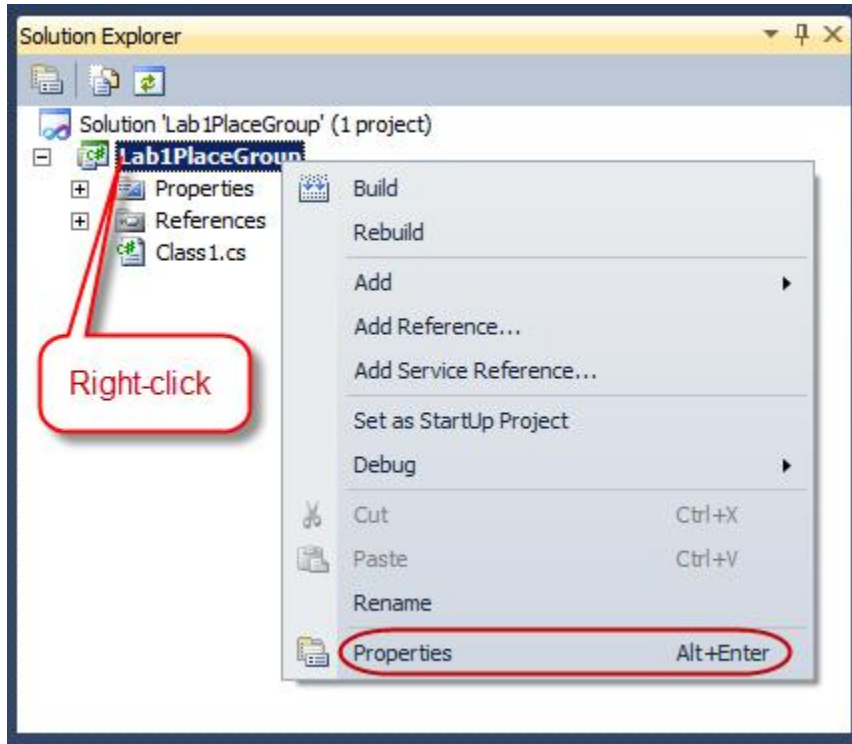


In the **Solution Explorer** window you saw in step 5, click **RevitAPI** under **Reference** node. In the **Properties** window, click **Copy Local** property, and then click the drop-down list, select **False**. Repeat the same steps to change **RevitAPIUI**'s **Copy Local** property value to **False**.

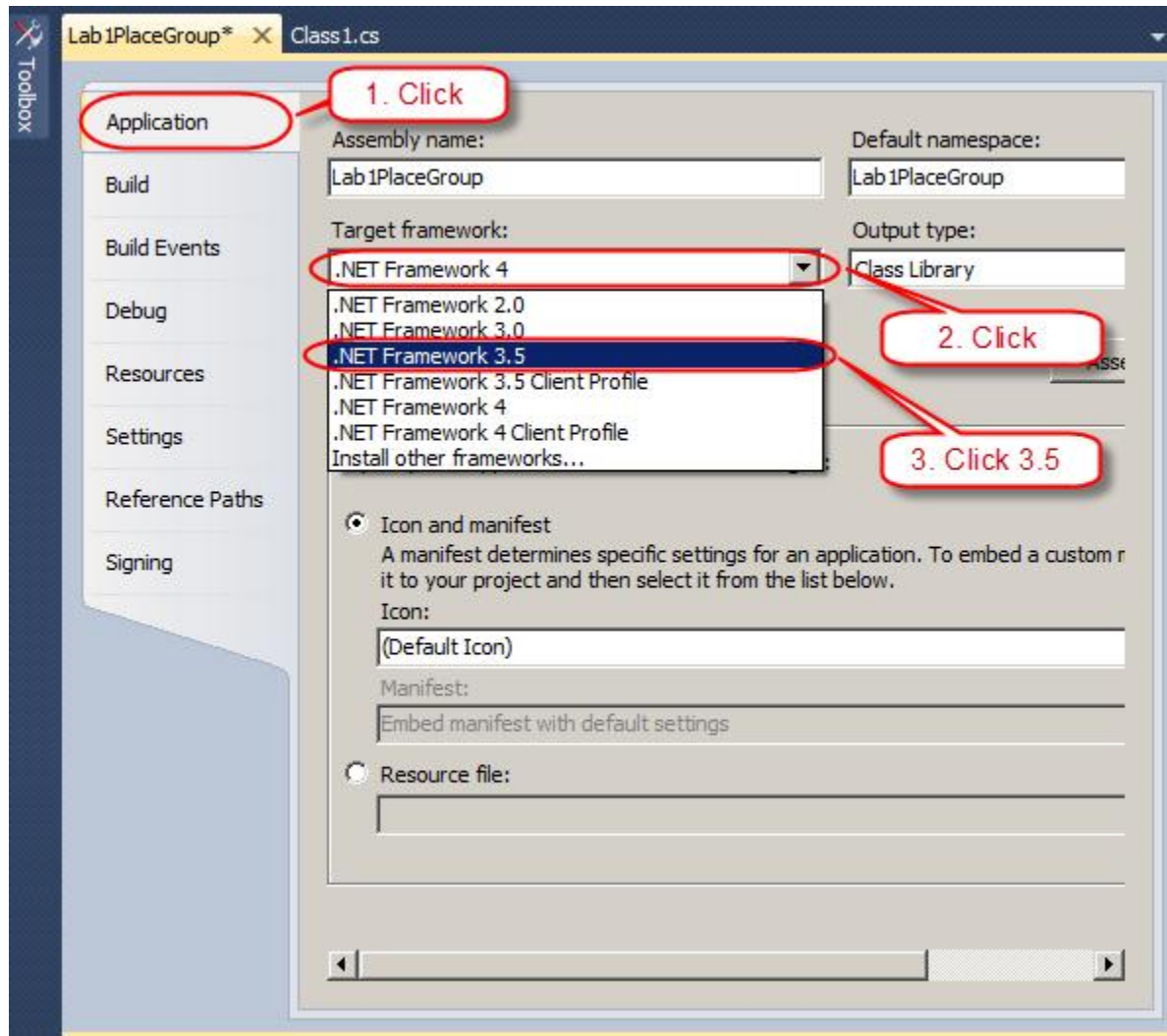
7. **Set target .NET Framework:**

Attention: Autodesk Revit 2011 supports the use of .NET Framework 3.5. Autodesk Revit 2012 and higher supports the use of .NET Framework 4.0, which Visual C# 2010 Express uses by default. The following step is needed in order to use the correct version. If the Revit Architecture version that you are using supports .NET Framework 4.0, you can skip step 7, 8 and 9.

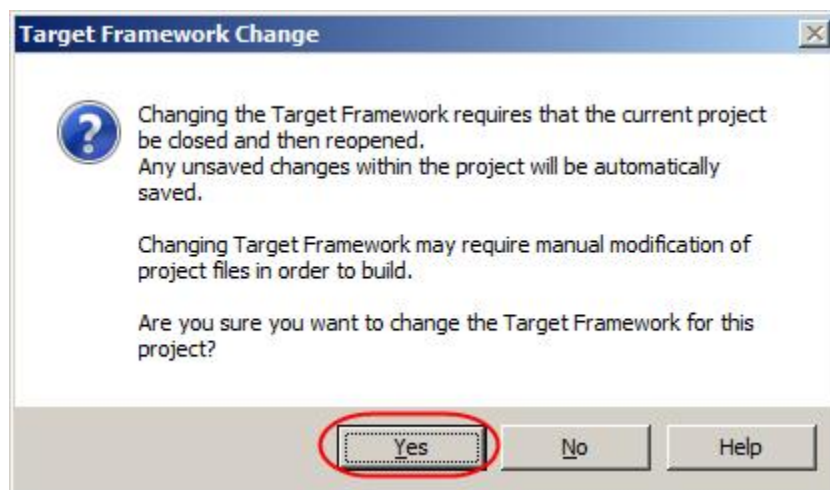
In the **Solution Explorer** window, right-click **Lab1PlaceGroup** project, and click **Property**.



8. In the displaying Project Property form, click **Application** tab on the left-hand side of the window, click the drop-down list below **Target framework**, and then click the **.NET Framework 3.5** option in the list.



9. The following message box appears to ask your confirmation. Click **Yes** to confirm the change.



10. **Add the code:**

Double click **Class1.cs** in the **Solution Explorer** window to show the code-editing window. Delete everything in this window and then type the following C# code. To get the full experience of developing with Visual C# Express – including the use of features such as IntelliSense – we recommend you type the code from this guide rather than copying and pasting it. That said, if constrained for time you can also copy and paste into the Visual C# Express code window: although this reduces the experience you gain from working with the code directly.

```
11.     using System;
12.     using System.Collections.Generic;
13.     using System.Linq;
14.
15.     using Autodesk.Revit.DB;
16.     using Autodesk.Revit.DB.Architecture;
17.     using Autodesk.Revit.UI;
18.     using Autodesk.Revit.UI.Selection;
19.     using Autodesk.Revit.ApplicationServices;
20.     using Autodesk.Revit.Attributes;
21.
22.     [TransactionAttribute(TransactionMode.Manual)]
23.     [RegenerationAttribute(RegenerationOption.Manual)]
24.     public class Lab1PlaceGroup : IExternalCommand
25.     {
26.         public Result Execute(
27.             ExternalCommandData commandData,
28.             ref string message,
29.             ElementSet elements)
30.         {
31.             //Get application and document objects
32.             UIApplication uiApp = commandData.Application;
33.             Document doc = uiApp.ActiveUIDocument.Document;
34.
35.             //Define a Reference object to accept the pick result.
36.             Reference pickedRef = null;
37.
38.             //Pick a group
39.             Selection sel = uiApp.ActiveUIDocument.Selection;
40.             pickedRef = sel.PickObject(ObjectType.Element, "Please select a
group");
41.             Element elem = pickedRef.Element;
42.             Group group = elem as Group;
43.
44.             //Pick a point
45.             XYZ point = sel.PickPoint("Please pick a point to place group");
46.
47.             //Place the group
48.             Transaction trans = new Transaction(doc);
49.             trans.Start("Lab");
50.             doc.Create.PlaceGroup(point, group.GroupType);
51.             trans.Commit();
52.
53.             return Result.Succeeded;
54.         }
55.     }
```

Don't worry about the details of the code for now, you'll come back to this shortly in the next couple of lessons.

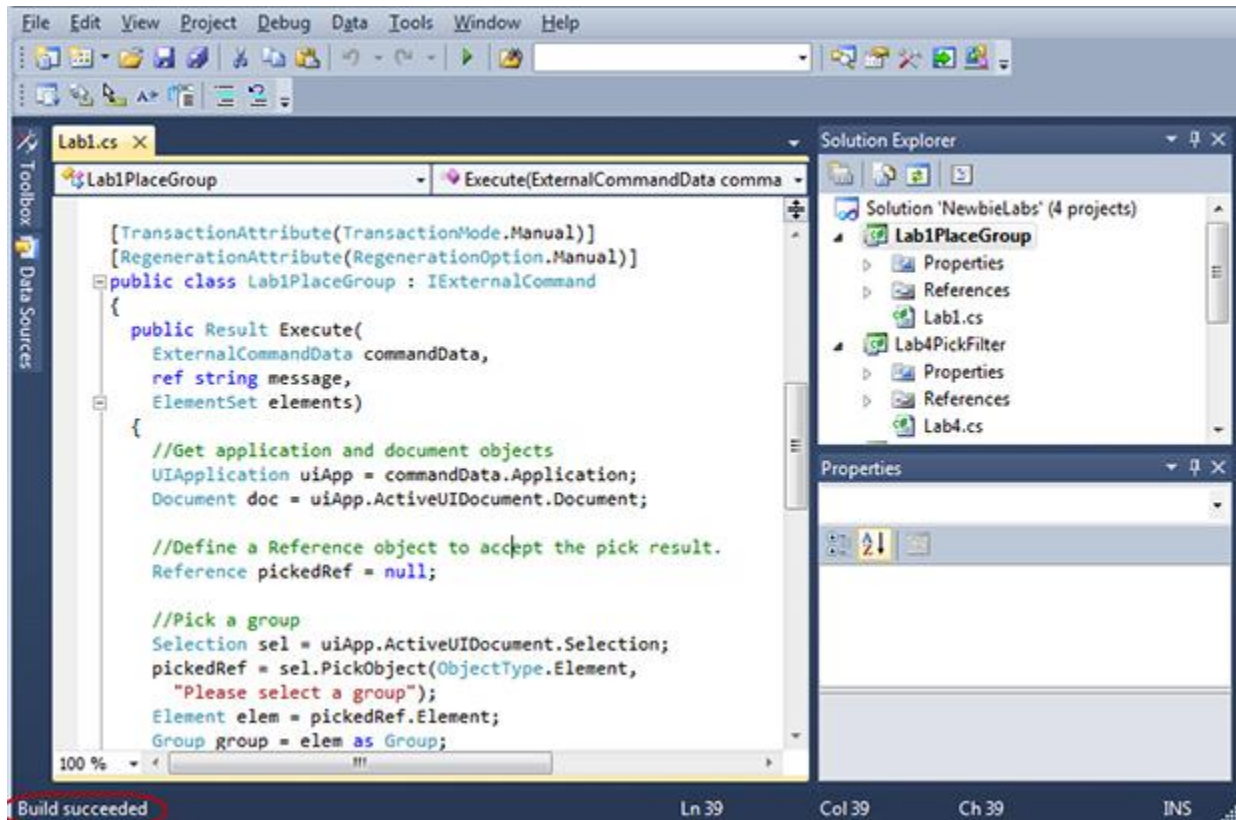
11. **Save the file:**

On the **File** menu, click **Save All**.

12. **Build the project:**

The code you have written is in human readable form. To make the code readable by a computer, you will need to translate it or “build” it.

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



Note: If you are working with Revit 2012 API, you will see a warning stating that 'Autodesk.Revit.DB.Reference.Element' is obsolete. At this point, don't worry about this warning. We will address what the code needs to be changed to, in Lesson 3.

If you are working with Revit 2013 and higher API, you will need to replace the following line of code:

```
Element elem = pickedRef.Element;
```

with the following:

```
Element elem = doc.GetElement(pickedRef);
```

That's it! You have just written your first plug-in for Autodesk Revit.

Before you actually work with the plug-in in Revit, you will need to do one more step, which is to write an AddIn manifest.

Writing an AddIn Manifest

An AddIn manifest is a file located in a specific location checked by Revit when the application starts. The manifest includes information used by Revit to load and run the plug-in.

1. **Add the manifest code:**

Start **Notepad.exe** from the Windows **Start** menu. Copy and paste the following plug-in load settings to the Notepad editor.

```
2. <?xml version="1.0" encoding="utf-8"?>
3. <RevitAddIns>
4.   <AddIn Type="Command">
      <Assembly>
        C:\test\Lab1PlaceGroup\Lab1PlaceGroup\bin\Release\Lab1PlaceGroup.dll
      </Assembly>
5.   <ClientId>502fe383-2648-4e98-adf8-5e6047f9dc34</ClientId>
6.   <FullClassName>Lab1PlaceGroup</FullClassName>
7.   <Text>Lab1PlaceGroup</Text>
8.   <VendorId>ADSK</VendorId>
9.   <VisibilityMode>AlwaysVisible</VisibilityMode>
10.  </AddIn>
    </RevitAddIns>
```

Depending on what version you are using you may need to change the path here to match your Lab1PlaceGroup.dll location on your computer:

C:\test\Lab1PlaceGroup\Lab1PlaceGroup\bin\Release\Lab1PlaceGroup.dll

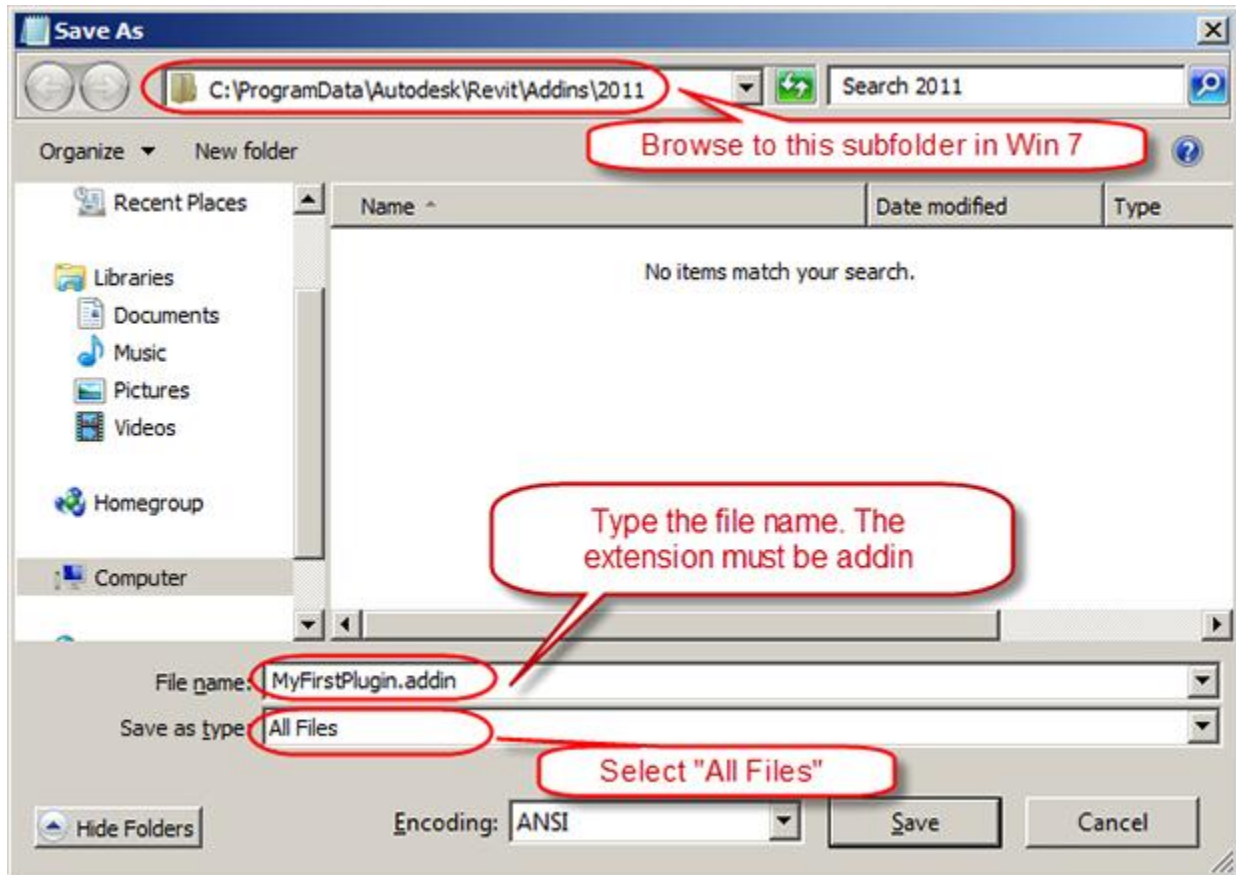
11. **Save the file:**

On **Notepad's File** menu, click **Save**. Enter **MyFirstPlugin.addin** in the **File name** box. Change **Save as type** to the **All Files** option (the file name is up to you; however the file extension must be ".addin"). Browse to the following subfolder, and then click the **Save** button.

- For Windows XP* - C:\Documents and Settings\All Users\Application Data\Autodesk\Revit\Addins\2011\
- For Windows Vista/Windows 7* - C:\ProgramData\Autodesk\Revit\Addins\2011\ (The ProgramData folder is hidden by default)

** The path may vary depending on the flavour of Autodesk Revit you are using.*

For example, here is the setting in Save As dialog in Windows 7 for Revit Architecture 2011.



Let's run the plug-in to see what it does in Revit.

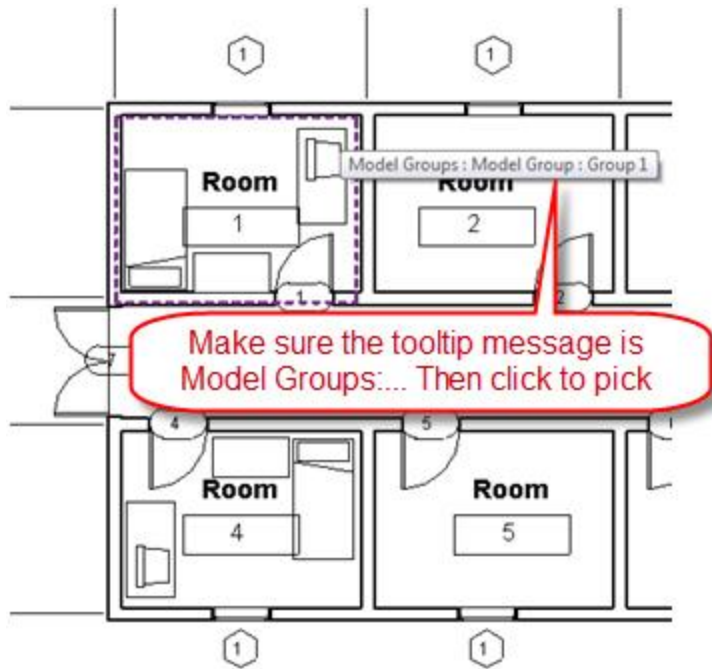
Running the Plug-in

1. Start Autodesk Revit Architecture.
2. Open the [Hotel.rvt](#) Project file.
3. **Load your plug-in into Revit and allow the plug-in to communicate with Revit:** Inside Revit on the **Add-Ins** ribbon tab, click the **External Tools** drop-down list, then click **Lab1PlaceGroup**. This will start your plug-in.

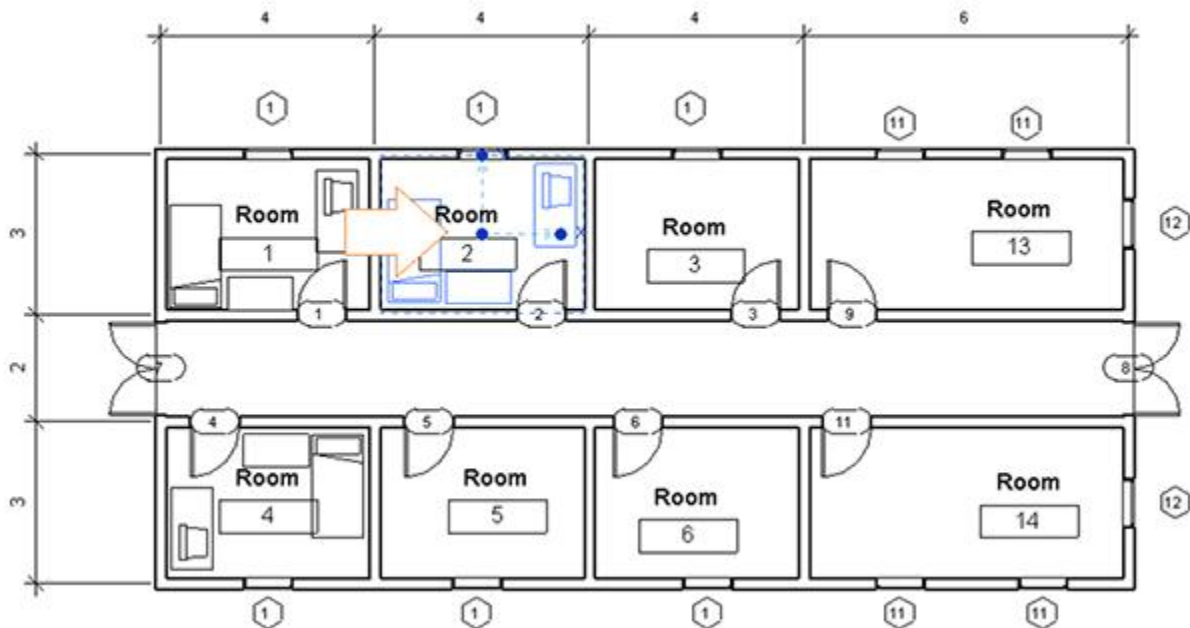


4. **Work with the plug-in:** Move the cursor over **Room1** in the Revit building model. When the cursor is hovering over the furniture group, its

bounding box should be highlighted as per the below picture, with a tooltip showing **Model Groups : Model Group : Group 1**. Click to select this furniture group. (Note: when highlighted the room looks very similar to the group. Please carefully select the group according to the message in the tooltip. If the room is selected, you will not see the expected result after the following step.)



- Pick a point in another room, for example in **Room 2**. You should see the group copied to this location. The center of the new group is the point you selected.



Congratulations! You have just written your first plug-in for Autodesk Revit. You will be reviewing the code in detail in [Lesson 3](#).

Before you move on to the next lessons, let us go back to some of the things we skipped over earlier, starting with basics concept about programming, and the benefits it can bring to your day-to-day work.

Additional Topics

Introduction to Programming

The C# code you have just executed to copy a group is only 30 lines long. Here you see a small amount of code working in a similar way to the internal Revit command, **Create Similar**. Software programming allows you to capture the logic of a particular functionality once and then reap the benefits over and over again, every time you want to perform this functionality.

What is Programming?

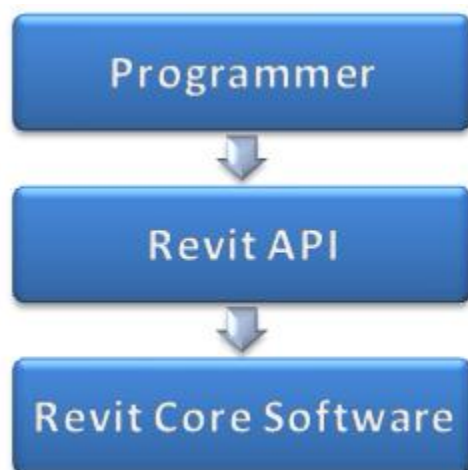
A simple answer to this question is: Computer programming is the process of creating a sequence of instructions to tell the computer to do something. You can look at your program as a sequence of instructions. During the course of the upcoming lessons, you will look at the various lines and blocks of code and look at them all in the context of being instructions for a computer.

If you were to explain what computers are to a young child, you might say: a computer is a tool which follows instructions you provide. Programming is one way of giving instructions to the computer. Internally, a computer sees these instructions encoded as a series of numbers (also called machine code). The human-readable instructions you saw at the beginning of this lesson is called source code and the computer converts these instructions to machine code which it can then read and execute. A sequence of such instructions (or code), written to perform a specific task, is called a program and a collection of such programs and related data is called a software. Autodesk Revit is one such software product.

Source code can be written in different languages, just as humans use different languages to communicate between ourselves. The language you will be using in this guide is called C# (pronounced “C-Sharp”).

What is an API?

API is the acronym for Application Programming Interface: the way a software programmer can communicate with a software product. For instance, the Revit API is the way programmers can work with Revit, and it establishes what functionality a software programmer can use within Revit. Such as the Revit API allows you to write instructions for Revit to execute one after the other.



To put this slightly differently: commercial software companies, such as Autodesk, often distribute a set of libraries which you can use in your own program to interact with a particular software product, such as Autodesk Revit, and extend its functionality. This set of libraries is known as the software product's API.

The type of program you write to interact with a software product and extend its functionality will depend upon how the API has been designed and what has been exposed (through APIs) for you to work with.

What is a Plug-in?

A software plug-in is a type of program module (or file) that adds functionality to a software product, usually in the form of a command automating a task or some customization of the product's behavior. When you talk about a plug-in for Revit – and you will also hear the term Add-In used for this product – we mean a module containing code that makes use of the Revit API. Revit loads such plug-ins and uses them to adjust its behavior under certain conditions, such as when a particular command is executed by the user of the plug-in.