

The logo for NBIT is a rectangular graphic. The top portion is a solid blue color, and the bottom portion is a lighter, medium-blue color. The letters "NBIT" are centered in the blue area in a white, bold, sans-serif typeface.

NBIT

NopCommerce Plugin

nopCommerce plugins allow users to easily customize & enhance any **nopCommerce** based store site. This is a document on how nopCommerce plugin is developed and how it is implemented.

Ejan Shrestha

Table of Content

Table of Content 1

Plugin Structure In NopCommerce1

Steps to create a basic nopCommerce 4.20 plugin 2

Plugin Structure In NopCommerce

By default, all plugin projects with source code must be stored in \Plugins folder inside your nopCommerce application folder. It is a best practice to name a plugin project as “Nop.Plugin.{Group}.{Name}”.

A compiled plugin must be stored inside the \Presentation\Nop.Web\Plugins folder. This is the location where nopCommerce looks for any plugins to load in its application on runtime.

Each plugin must have a “plugin.json” file inside its directory which is used by nopCommerce application to determine whether a plugin is compatible with the current version of nopCommerce and also other details regarding plugin as described in the table below.

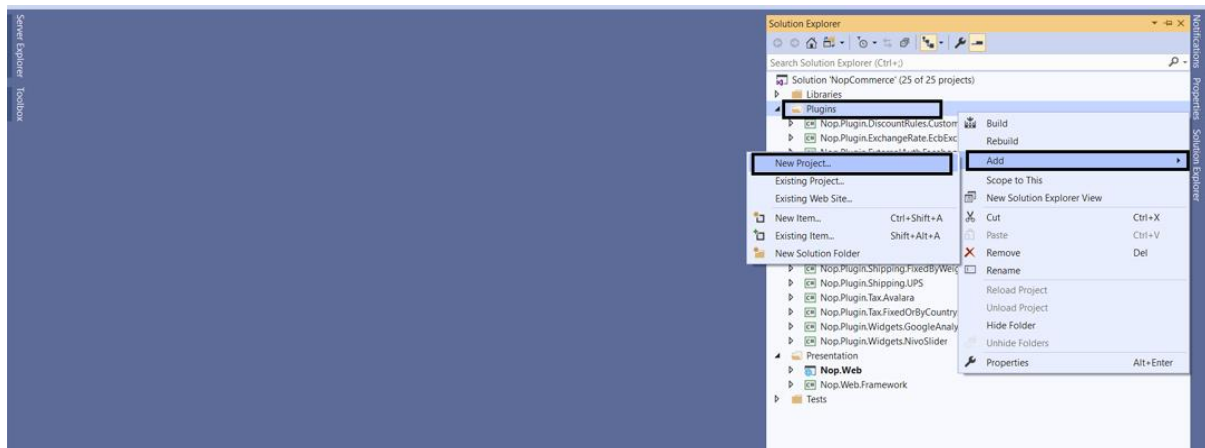
A “logo.jpg” file can also be supplied with the plugin which is used to display a plugin picture wherever needed.

Property Name	Description
Group	Group name is used to identify plugin groups by nopCommerce.It is used to filter plugins by group name on the plugins page in our admin panel.
FriendlyName	This is the name of the plugin that will be shown in the plugins list and will be used everywhere in the system to identify the plugin
SystemName	This should be unique and it is used by nopCommerce to load plugins by the System name
Version	This is the version of the plugin and it can be written in any format of our choice
SupportedVersions	This is an array and it can contain one or more nopCommerce supported versions of this plugin
Author	This is basically the writer of the plugin.
DisplayOrder	This is used to display plugins by an integer-order in the plugin list inside your administrator panel.
FileName	This is the name of your output file .Usually,it is the name of our plugin project with the .dll extension
Description	Here,we write about the plugin and what it does .

Steps to create a basic nopCommerce 4.20 plugin

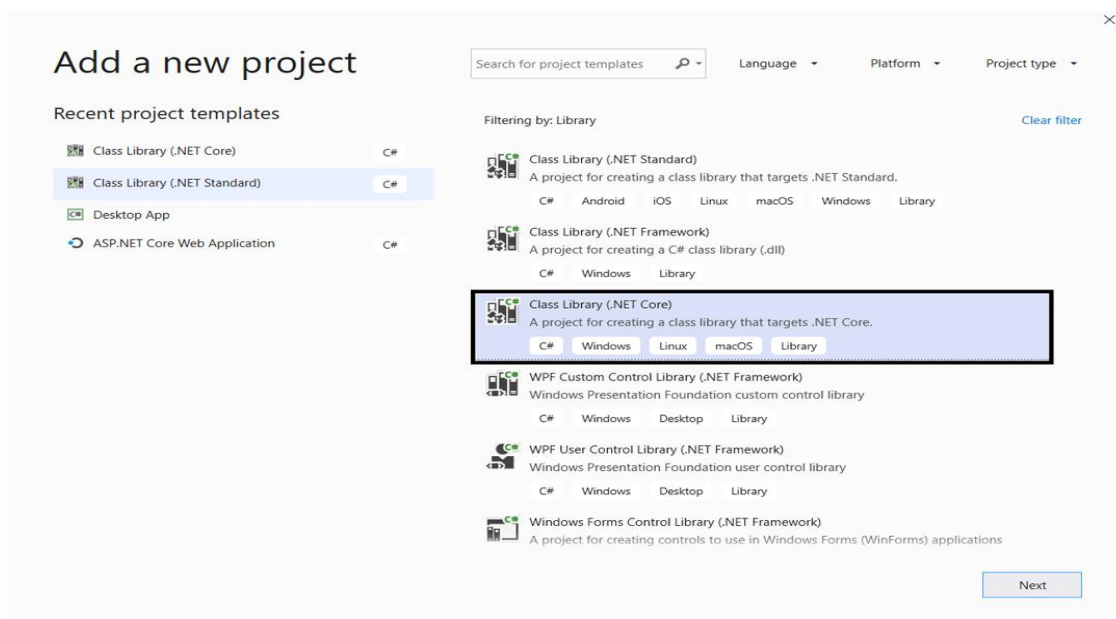
Step 1

Navigate to “Solution Explorer” and right-click on the “Plugins” folder. Go to, Add > New Project.



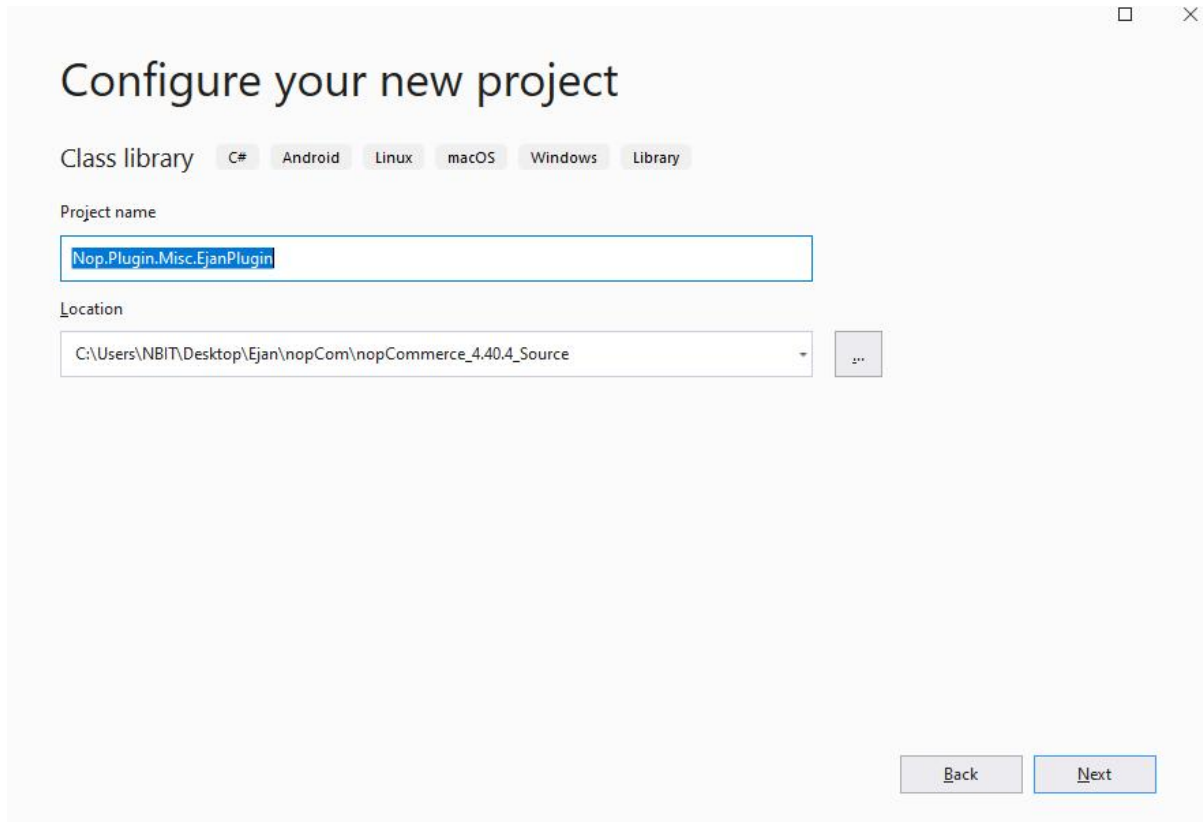
Step 2

In the “Add a new project” window, select “Class Library (.NET Core)” project and click on “Next”.



Step 3

Write a meaningful project name for NopCommerce Plugin Project. We will use “Nop.Plugin.Misc.EjanPlugin” for this sample project, as shown in the below image.



Configure your new project

Class library C# Android Linux macOS Windows Library

Project name

Nop.Plugin.Misc.EjanPlugin

Location

C:\Users\NB\IT\Desktop\Ejan\nopCom\nopCommerce_4.40.4_Source

Back Next

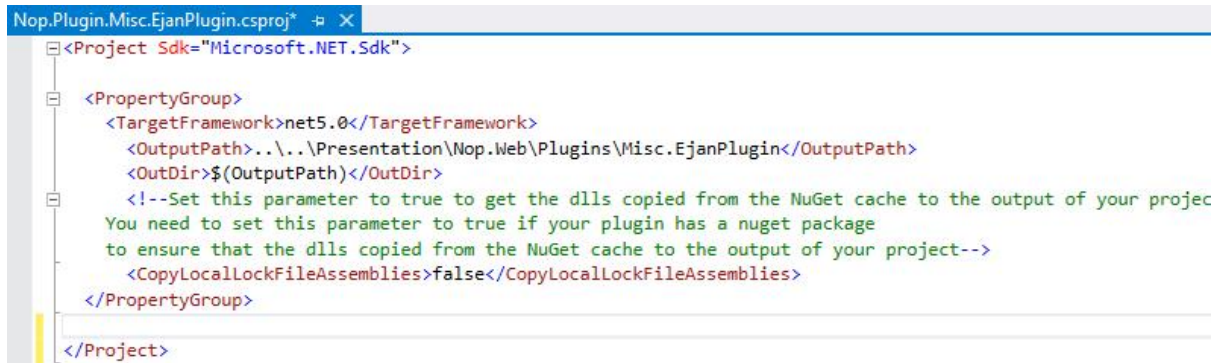
Note: Your plugin project must be located inside the \Plugins folder in your nopCommerce project directory.

Step 4

Open your plugin project .csproj file. Right-click on the plugin project you just created and go to “Edit project file”.

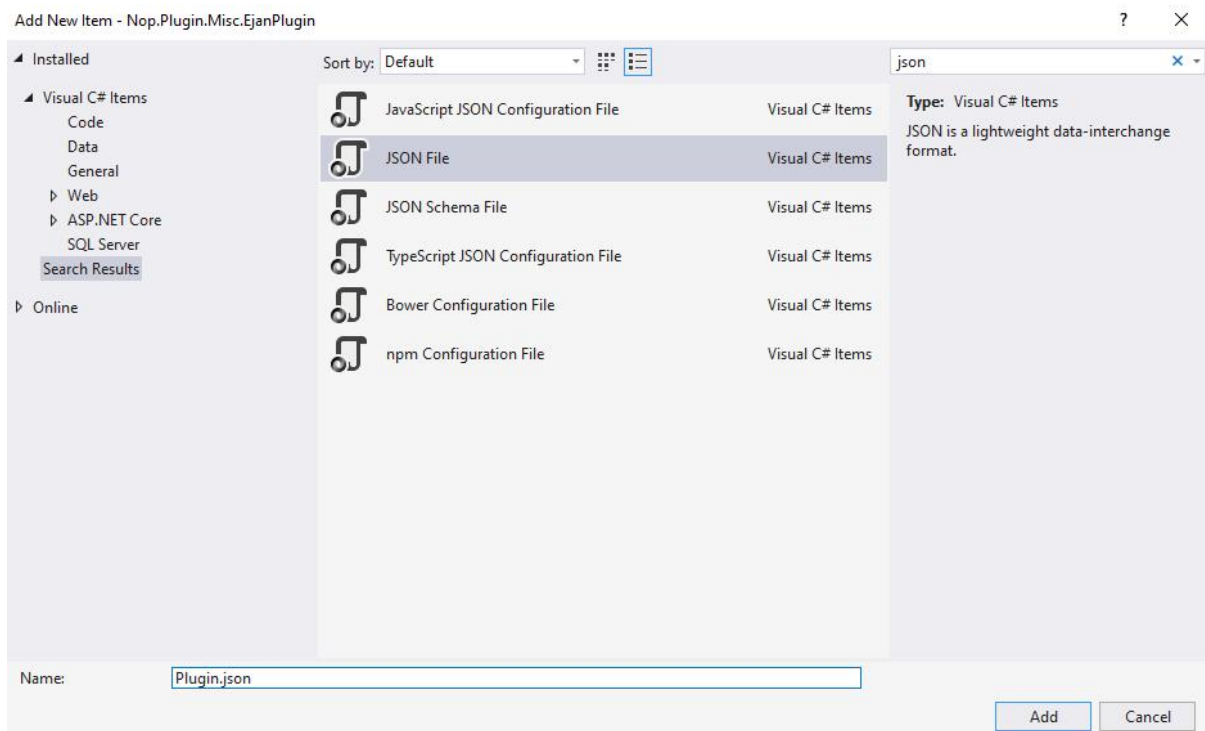
Step 5

In your .csproj file, set “OutputPath”, “OutDir” and “CopyLocalLockFileAssemblies”.



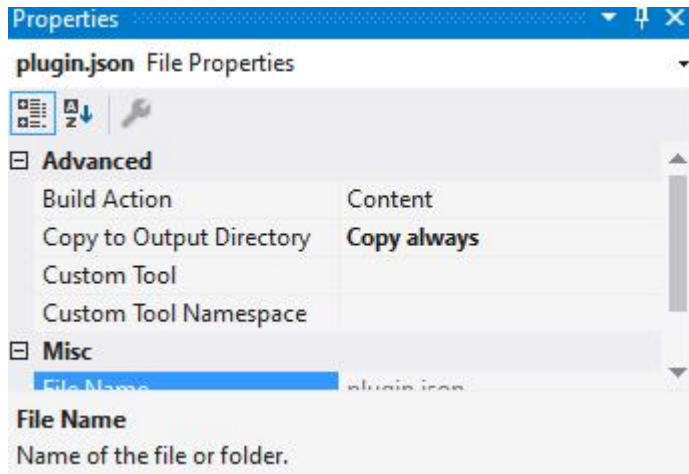
Step 6

- I. Right-click on your plugin project, go to Add > New item.
- II. Add a plugin.json file as shown in the below image.



Step 7

- I. Right-click on the plugin.json file that you just added and go to properties.
- II. In plugin.json file properties inside Visual Studio, set “Build Action” to “Content” and “Copy to Output Directory” to “Copy always”.



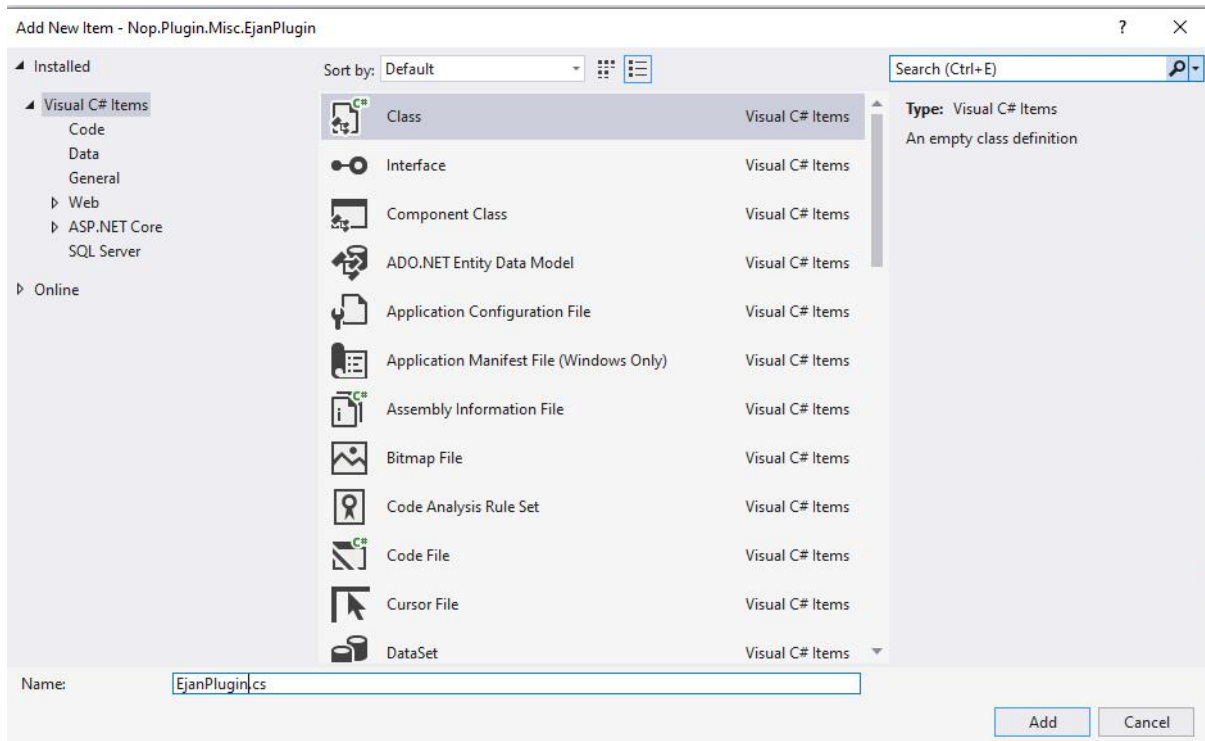
Step 8

Add contents to your newly added plugin.json file.



Step 9

- I. Right-click on your plugin project and go to Add > Class.
- II. You can choose to set a meaningful name for the new class. In our case, we choose to set “EjanPlugin.cs” and click on the Add button.



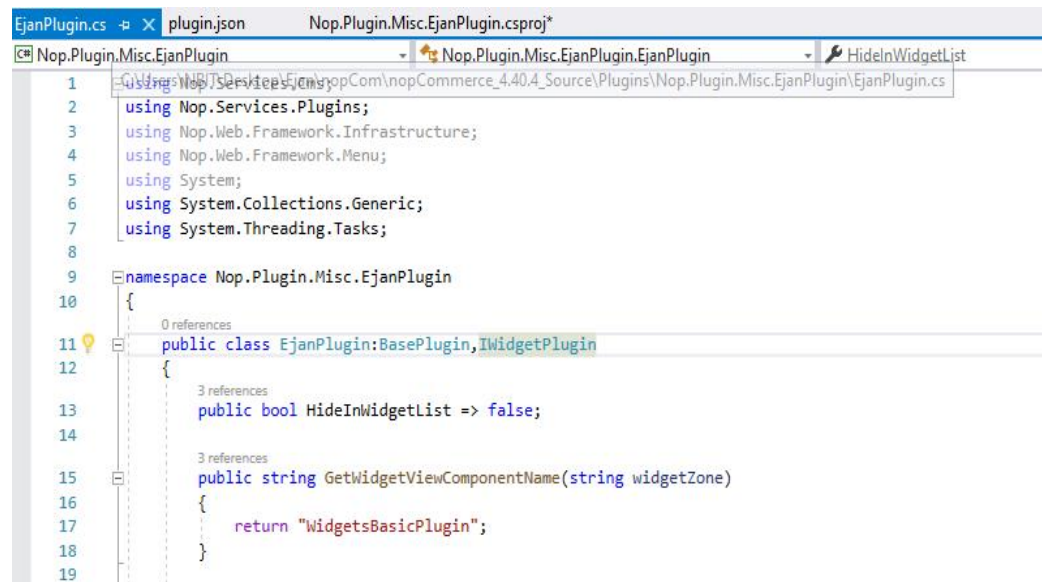
Step 10

- I. Right-click on “Dependencies” in your plugin project and click on “Add Reference”.
- II. In the “Reference Manager” window, go to Projects > Solution and select “Nop.Web.Framework” from the list of available projects as shown in the image below.
- III. Once added to Nop.Web.Framework in Dependencies, go to its properties and set “Copy Local” to “No”, as shown in the image below.

Note: We don’t need to add other projects separately after nopCommerce version 4.40 because the Nop.Web.Framework contains a reference to all other projects in the solution that come out of the box with the nopCommerce project.

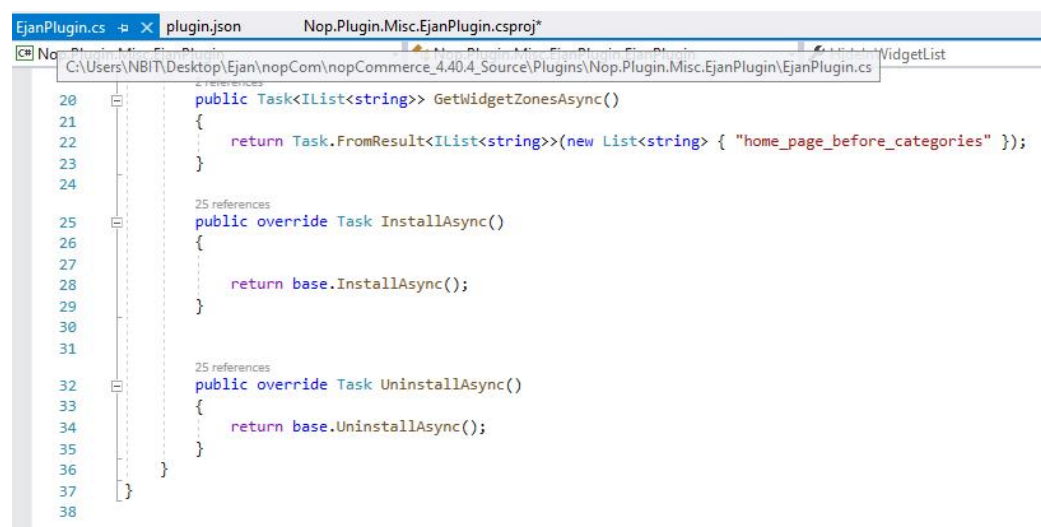
Step 12

Now We inherit the BasePlugin class and IWidgetPlugin interface in the EjanPlugin class , write override for install and uninstall methods and implements the IWidgetPlugin interface as shown below:



```

EjanPlugin.cs  plugin.json  Nop.Plugin.Misc.EjanPlugin.csproj*
Nop.Plugin.Misc.EjanPlugin
  Nop.Plugin.Misc.EjanPlugin.EjanPlugin
  HideInWidgetList
  C:\Users\NBIT\Desktop\Ejan\nopCom\nopCommerce_4.40.4_Source\Plugins\Nop.Plugin.Misc.EjanPlugin\EjanPlugin.cs
1  using Nop.Services.Plugins;
2  using Nop.Web.Framework.Infrastructure;
3  using Nop.Web.Framework.Menu;
4  using System;
5  using System.Collections.Generic;
6  using System.Threading.Tasks;
7
8
9  namespace Nop.Plugin.Misc.EjanPlugin
10 {
11     0 references
12     public class EjanPlugin:BasePlugin,IWidgetPlugin
13     {
14         3 references
15         public bool HideInWidgetList => false;
16
17         3 references
18         public string GetWidgetViewComponentName(string widgetZone)
19         {
20             return "WidgetsBasicPlugin";
21         }
22     }
  
```



```

EjanPlugin.cs  plugin.json  Nop.Plugin.Misc.EjanPlugin.csproj*
Nop.Plugin.Misc.EjanPlugin
  Nop.Plugin.Misc.EjanPlugin.EjanPlugin
  HideInWidgetList
  C:\Users\NBIT\Desktop\Ejan\nopCom\nopCommerce_4.40.4_Source\Plugins\Nop.Plugin.Misc.EjanPlugin\EjanPlugin.cs
20  public Task<IList<string>> GetWidgetZonesAsync()
21  {
22      return Task.FromResult<IList<string>>(new List<string> { "home_page_before_categories" });
23  }
24
25  25 references
26  public override Task InstallAsync()
27  {
28      return base.InstallAsync();
29  }
30
31  25 references
32  public override Task UninstallAsync()
33  {
34      return base.UninstallAsync();
35  }
36
37
38
  
```

Handling "Install" and "Uninstall" methods

Some plugins can require additional logic during plugin installation. For example, a plugin can insert new locale resources. So open your IPlugin implementation (in most case it'll be derived from BasePlugin class) and override the following methods:

Install. This method will be invoked during plugin installation. You can initialise any settings here, insert new locale resources, or create some new database tables (if required).

Uninstall. This method will be invoked during plugin uninstallation.

We have set the bool `HiddenInWidgetList` false because we want to show our plugin in the widgets list in the admin panel after we install the plugin.

`GetWidgetViewComponentName(string widgetZone)`

Get a name of a view component for displaying a widget and returns a ViewComponent name.

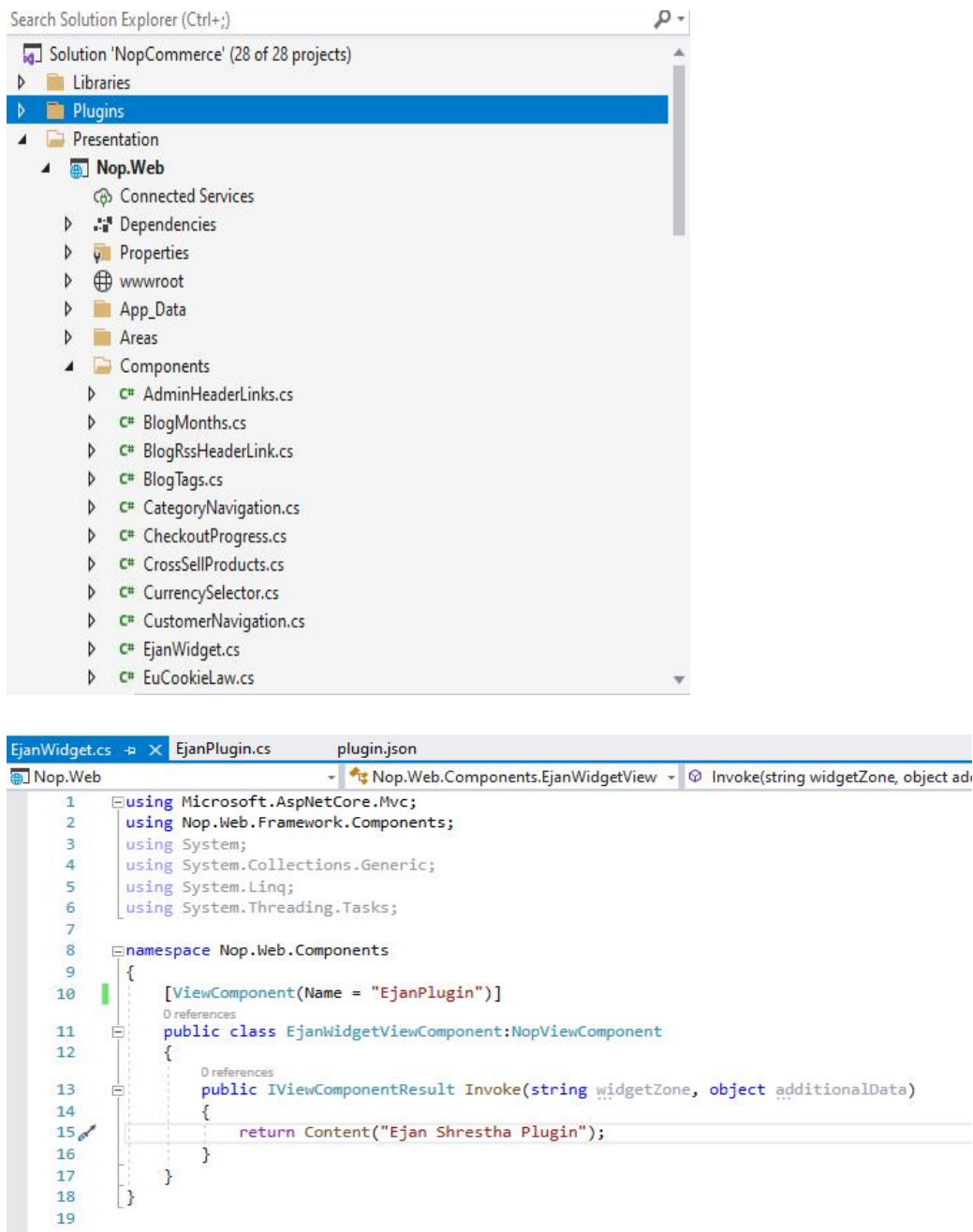
`GetWidgetZonesAsync()`

Gets widget zones where this widget should be rendered

And return a task that represent the asynchronous operation The Task result contains the widget zones

Step 13

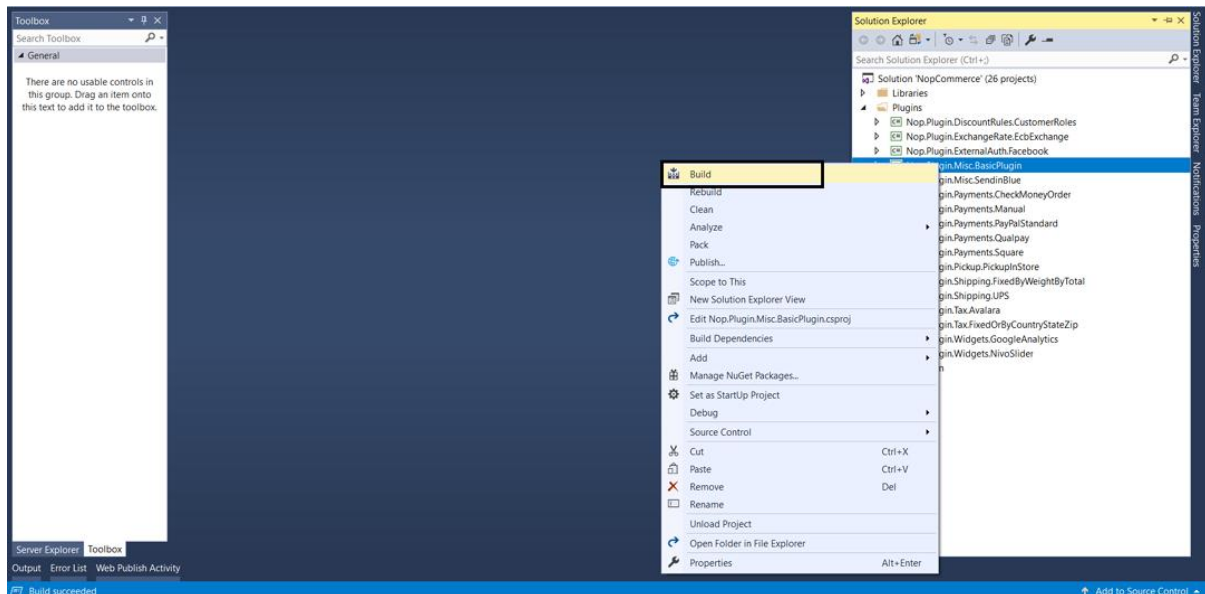
Create a new class in /Presentation/Nop.Web/Components. Here we have created a view component of EjanWidget.cs class where we inherit the NopViewComponent



ViewComponent attribute is used to change the name of the view component. For example, we have class named "EjanWidgetViewComponent" and want to set view component name to "EjanPlugin", this can be done using ViewComponent attribute.

Step 14

- I. Both, **plugin.json** and **logo.jpg**, files should be marked as “Content” for “Build Action” and “Copy always” for “Copy to Output Directory” properties for both files inside our Visual Studio.
- II. Right-click on the project(Nop.Plugin.Misc.EjanPlugin) and click on Build . After Successfully Build, a new folder will be created in the /Presentation/Nop.Web/Plugins with a SystemName of the project(Misc.EjanPlugin).



Step 15

- I. Run your project and login to /Admin area. Once you are logged in, go to Configuration > Local plugins
- II. On the local plugins page, navigate to the plugin that you just created and click on the “Install” button
- III. Once you have clicked on the “Install” button for your plugin, you will see buttons on top of the local plugins page. Click on “Restart application to apply changes” to complete your plugin installation and restart the server.
- IV. Now Enable the Plugin by clicking the edit button and check the enable button and click on save.