Create a C# Template for AutoCAD with Visual Studio 2015

The goal of this tutorial is to show how to create a C# ProjectTemplate for AutoCAD in Visual Studio 2015, template allowing to automatically launch AutoCAD by loading the DLL from Visual Studio in Debug mode.

The example shown uses Visual Studio Community 2015 but it is easily transferable to other versions.

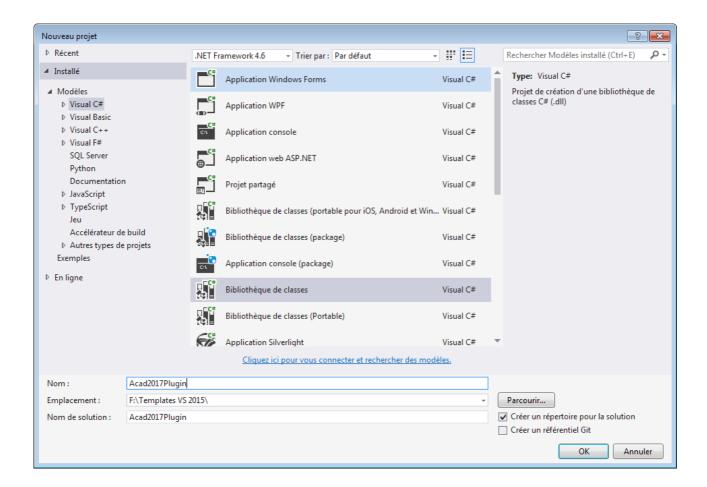
It will be created a template for AutoCAD 2017 but there again it is possible to transpose to target other versions of AutoCAD.

The path to the output directory for the Debug mode used in the example is the default (.\bin\Debug).

Start a new project

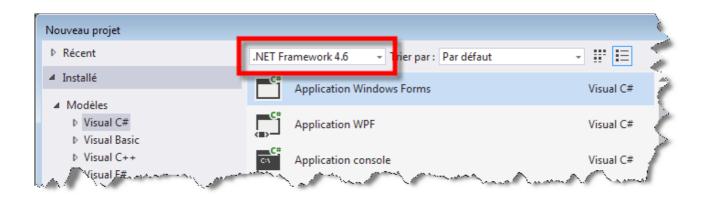
In Visual Studio, select the *Visual C#* language and the type of project: *Class Library*.

Rename the project: Acad2017Plugin for example, and change the path to the location as desired.



Specify the target Framework

In the New Project dialog box, select the targetted Framework drop-down list.



It is best to target the version of the Framework corresponding to that installed by the version of AutoCAD for the what the template is created:

AutoCAD Version	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
Re lease	R17.0	R17.1	R17.2	R18.0	R18.1	R18.2	R19.0	R19.1	R20.0	R20.1	R21.0	R22.0
DWG Format	AC1021	AC1021	AC1021	AC1024	AC1024	AC1024	AC1027	AC1027	AC1027	AC1027	AC1027	AC1032
Installed .NET Framework	2.0	2.0	2.0	3.5	3.5	4.0	4.0	4.0	4.5	4.5	4.6	4.6
AcCore Mgd.dll require d												
	•											
Visual Studio 2005 (8.0)												
Visual Studio 2008 (9.0)												
Visual Studio 2010 (10.0)												
Visual Studio 2012 (11.0)												
Visual Studio 2013 (12.0)												
Visual Studio 2015 (14.0)												
Visual Studio 2017 (15.0)												

Add AutoCAD libraries

In Solution Explorer, select *References* then right click and *Add a reference*... In the *Reference Manager* dialog, click *Browse*...

References to add are in the version targeted for AutoCAD installation directory, or better, in the ObjectARX20XX\inc folder corresponding to the version of AutoCAD targeted (inc-win32 or inc-x64 according to the versions of AutoCAD and the platform of the station used).

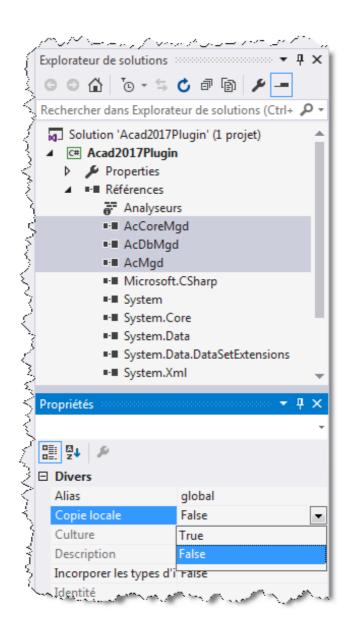
You can download the latest versions of the ObjectARX on this page:

http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=773204

Choose commonly used libraries (it will be easy to add others in the same way if the creation of the new project requires).

- AutoCAD 2007 to 2012: AcDbMgd.dll and AcMgd.dll
- AutoCAD 2013 to 2017: AcCoreMgd.dll, AcDbMgd.dll and AcMgd.dll

Gilles Chanteau



Add a draft code

Rename the class created by Visual Studio: *Class I* with a more meaningful name, for example, *Commands* from the solution Explorer. A dialog box offers to rename all references to *Class I* in the project, answer *Yes*.

Open the Commands class in the editor of code by double click on Commands in solution Explorer.

Add the *using* statements to import the most widely used AutoCAD namespaces and, possibly, an alias for the Autodesk.AutoCAD.ApplicationServices.Application class.

Add a few lines of code common to define an AutoCAD command:

```
using System;
using System.Collections.Generic;
using System.Ling;
using System.Text;
using Autodesk.AutoCAD.ApplicationServices;
using Autodesk.AutoCAD.DatabaseServices;
using Autodesk.AutoCAD.EditorInput;
using Autodesk.AutoCAD.Geometry;
using Autodesk.AutoCAD.Runtime;
using AcAp = Autodesk.AutoCAD.ApplicationServices.Application;
namespace Acad2017Plugin
{
    public class Commands
        [CommandMethod("TEST")]
        public void Test()
            var doc = AcAp.DocumentManager.MdiActiveDocument;
            var db = doc.Database;
            var ed = doc.Editor;
            using (var tr = db.TransactionManager.StartTransaction())
                tr.Commit();
            }
        }
    }
}
```

Build the solution to control the absence of error (F6).

Add a script to load the application at startup of AutoCAD

From Solution Explorer, right-click on the Acad2017Plugin project and then *Add* and *New Item...* (Ctrl+Shift+A).

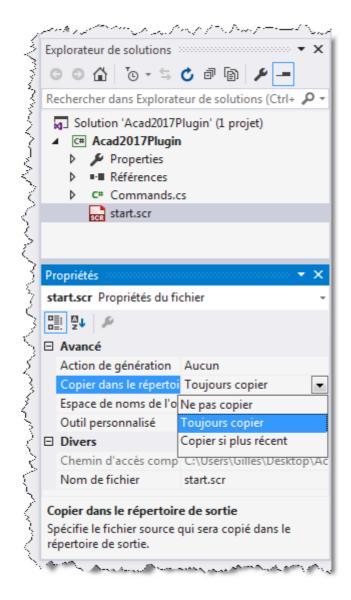
In the Add New Item dialog box, choose: Text File. Rename the file: start.scr and click on Add.

The file opens in the Editor window, add the following statement including a space at the end.

```
netload "Acad2017Plugin.dll"
```

Save the start.scr file.

In the start.scr properties (select start.scr in Solution Explorer), put the *Copy in the Output Directory* property to Always Copy.



Note: with AutoCAD 2016 and 2017, it is imperative that the LEGACYCODESEARCH system variable is set to 1 for AutoCAD integrates the output directory (.\Acad2017Plugin\bin\Debug) in the search paths and so found the DLL and the script file.

Change the MSBuild project file to launch AutoCAD for debugging

MSBuild project files (.csproj) are xml files that describe and control the process of generation of applications.

It is in this file that should be added the instructions that allow to launch AutoCAD since Visual Studio in debug mode and load the DLL to start AutoCAD.

Build the solution and close Visual Studio before you can edit the Acad2017Plugin.csproj file.

Open the .\Acad2017Plugin\Acad2017Plugin\Acad2017Plugin.csproj file with a text editor (Notepad, notepad ++, etc.).

Add the following nodes at the end of the *PropertyGroup* node corresponding to the generation in debug (the second *PropertyGroup* node in the file), after changing, if necessary, the acad.exe file path corresponding to the version of the file start.scr (the output for the Debug mode directory) and targeted AutoCAD.

```
<StartAction>Program</StartAction>
<StartProgram>C:\Program Files\Autodesk\AutoCAD 2017\acad.exe</StartProgram>
<StartArguments>/nologo /b "start.scr"</StartArguments>
```

StartAction indicates that the generation of the project should start a program (which is usually not the case with a DLL).

StartProgram specifies the program to start here AutoCAD.

StartArguments contains the arguments that are going to run the script that will load the DLL.

The *PropertyGroup* node should look like this:

In the *ItemGroup* node are added to the project references.

If the solution directory is on the same drive as the one in the what are referenced DLLs (the directory of installation of AutoCAD or ObjectARX 20XX), registered paths are relative.

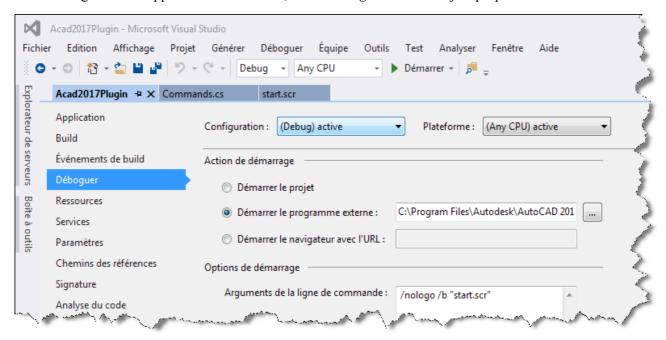
As the template will be exported in a different directory from the solution, need to replace these relative paths by absolute paths.

For example, replace the following relative paths:

```
<Reference Include="AcCoreMgd">
  <HintPath>..\..\..\..\..\..\..\objectARX 2017\inc\AcCoreMgd.dll</HintPath>
  <Private>False</Private>
</Reference>
<Reference Include="AcDbMgd">
  <HintPath>..\..\..\..\..\..\ObjectARX 2017\inc\AcDbMgd.dll</HintPath>
  <Private>False</Private>
</Reference>
<Reference Include="AcMgd">
  <hintPath>..\..\..\..\..\ObjectARX 2017\inc\AcMgd.dll</hintPath>
  <Private>False</Private>
</Reference>
By:
<Reference Include="AcCoreMgd">
  <HintPath>C:\ObjectARX 2017\inc\AcCoreMgd.dll</HintPath>
  <Private>False</Private>
</Reference>
<Reference Include="AcDbMgd">
  <HintPath>C:\ObjectARX 2017\inc\AcDbMgd.dll</HintPath>
  <Private>False</Private>
<Reference Include="AcMgd">
  <HintPath>C:\ObjectARX 2017\inc\AcMgd.dll/HintPath>
  <Private>False</Private>
</Reference>
```

Save the Acad2017Plugin.csproj file.

These changes should appear in Visual Studio, in the *Debug* tab of the Project properties.



Export the Template

Reopen the solution Acad2017Plugin.sln in Visual Studio.

Start debugging (F5) to control the operation. AutoCAD should open and display command line:

```
Command: netload the Assembly file name: "Acad2017Plugin.dll"
```

The below error message indicates either a wrong DLL name in the file start.scr. either that the start.scr file has not been copied into the output directory (see *Copy in the Output Directory* property).

Command: netload the Assembly file name: "Acad2017Plugin.dll" Impossible to load the Assembly. Error details: System.IO.FileNotFoundException: could not load file or assembly...

If everything works correctly, it's time to generate the template.

On the file menu, choose Export Template...

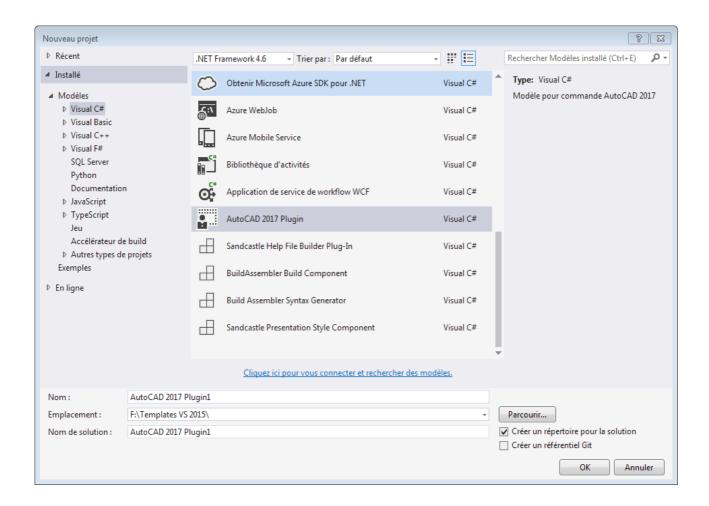
In the box Template Export Wizard, leave checked Project Template and do Continue.

Change the name of the template as desired, *AutoCAD 2017 Plugin*, for example, and possibly add a description: *Template for AutoCAD 2017 Custom Command*. Finish.

With the default options, the template is exported in the form of ZIP file in the files: *Visual Studio 2015\Templates\ProjectTemplates* and *Visual Studio 2015\My Exported Templates* folders of (My) Documents directory.

Visual Studio uses the templates located in *ProjectTemplates*, *My Exported Templates* serves rather as backup.

This template will be now purposed by Visual Studio at the start of a new project.



As is, renaming the project in the dialog box will be automatically reflected in the project generated for the default namespace and the name of the project (DLL), on the other hand, the name of the DLL to be loaded in the script will remain unchanged.

So that the name of the project is reflected in the script, we need to edit the file: *start.scr* and the template configuration file: *My Template.vstemplate* located in:

Visual Studio 2015\Templates\ProjectTemplates\AutoCAD 2017 Plugin.zip.

Extract the contents of archive AutoCAD 2017 Plugin.zip and open start.scr in a text editor.

Replace:

```
netload "Acad2017Plugin.dll"
```

by:

netload "\$projectname\$.dll"

Do not forget a space for validation at the end of the script, save the changes,

```
Open the My Template.vstemplate file in a text editor (it is an XML file) and replace:

<ProjectItem ReplaceParameters="false" TargetFileName="start.scr">start.scr
bv
```

```
<ProjectItem ReplaceParameters="true" TargetFileName="start.scr">start.scr</projectItem>
```

You can also add the version of the Framework required to ensure Visual Studio does offer this template when this version of the Framework is selected, add:

```
<RequiredFrameworkVersion>4.6</RequiredFrameworkVersion>
After the tag: <Description>.
```

Finally, the content of the file should look like this:

```
<VSTemplate Version="3.0.0" xmlns="http://schemas.microsoft.com/developer/vstemplate/2005"
Type="Project">
  <TemplateData>
    <Name>AutoCAD 2017 Plugin
    <Description>Template for AutoCAD 2017 Custom Command/Description>
    <RequiredFrameworkVersion>4.6</RequiredFrameworkVersion>
    <ProjectType>CSharp</ProjectType>
   <ProjectSubType>
   </ProjectSubType>
   <SortOrder>1000</SortOrder>
   <CreateNewFolder>true</CreateNewFolder>
   <DefaultName>AutoCAD 2017 Plugin
   <ProvideDefaultName>true</provideDefaultName>
   <LocationField>Enabled</LocationField>
    <EnableLocationBrowseButton>true</EnableLocationBrowseButton>
   <Icon>__TemplateIcon.ico</Icon>
    </TemplateData>
   <TemplateContent>
   <Project TargetFileName="Acad2017Plugin.csproj" File="Acad2017Plugin.csproj"</pre>
ReplaceParameters="true">
   <ProjectItem ReplaceParameters="true" TargetFileName="Commands.cs">Commands.cs/ProjectItem>
    <Folder Name="Properties" TargetFolderName="Properties">
    <ProjectItem ReplaceParameters="true"</pre>
TargetFileName="AssemblyInfo.cs">AssemblyInfo.cs
    </Folder>
   <ProjectItem ReplaceParameters="true" TargetFileName="start.scr">start.scr</projectItem>
   </Project>
  </TemplateContent>
</VSTemplate>
```

Save the changes and re-create the *AutoCAD 2017 Plugin.zip* archive with the files changed. Replace the old archive in *Visual Studio 2015\Templates\ProjectTemplates*.

Conclusion

The same procedure can be used to create other templates, targeting other versions of AutoCAD and/or framework, or for other types of projects (LISP functions, class libraries, application extensions,...).

It is possible to add instructions in the file start.scr as the launch of a command.

It is also possible to specify a DWG file to open in *the Command Line Arguments* (debug tab of the project properties) with a full or relative path (the root being the output directory).

Example:

```
"Drawing1.dwg" /nologo /b "start.scr"
```