COM (Component Object Model)

Component Object Model is a standard that allow application to be built from binary software components. COM is platform independent and we can make COM Component in any language that supports structures and pointers. COM includes: -

- A binary standard for function calling between components.
- A provision for strongly-typed groupings of functions into interfaces.
- A way for components to dynamically discover the interfaces implemented by other components.
- A mechanism to identify components and their interfaces uniquely.
- A "component loader" to set up component interactions and additionally to help manage component interactions.

COM define the behaviour of its object. Like other objects that objects contain data. That data of COM objects may be called through one or more related functions. And that group of functions are called interfaces. COM requires only pointers to access of those functions which are in particular interface. Remember that Interface of COM is different than interfaces of C++.

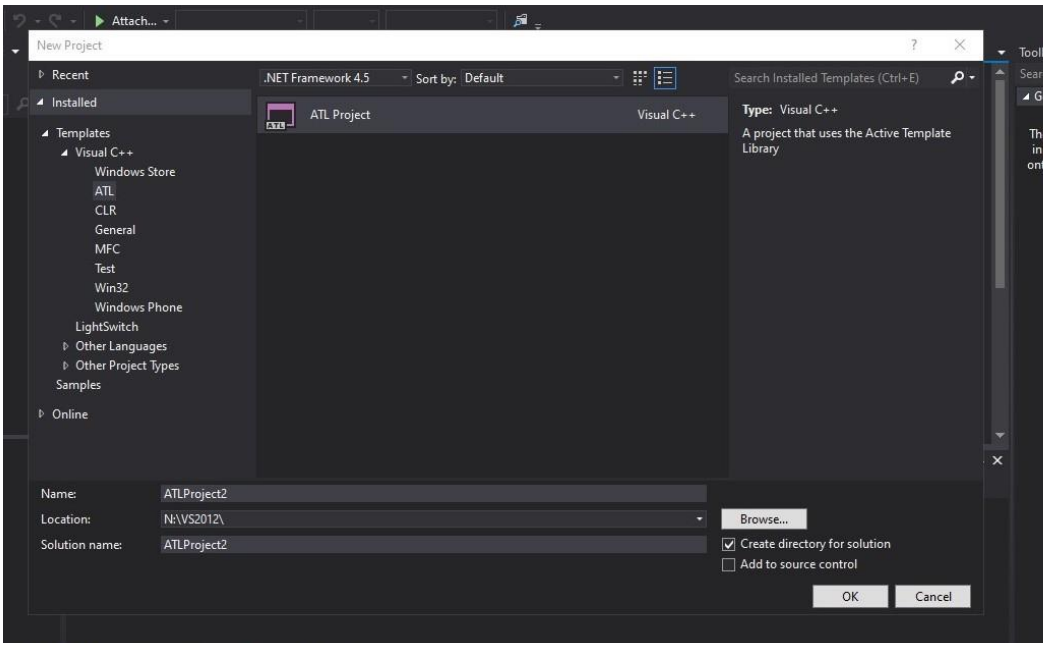
COM allows objects to interact across process in given boundaries. Object of COM uses code that implements each method of interface and provide us one binary compliant pointers to that functions in its library. It then makes those functions available for any client who ask for pointer to interface.

There are mainly two methods that we can load COM component: -

- ActiveX Control
- CoInitialize method

Creating ATL / COM DLL

- For this 1st we have to create ATL project.

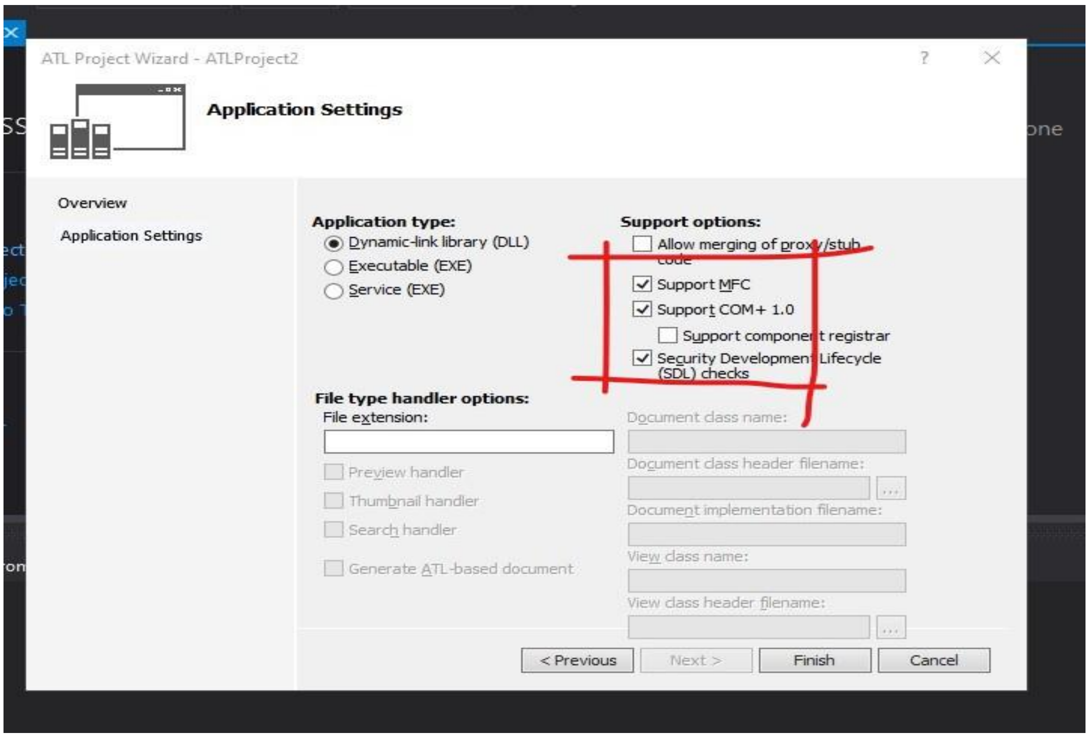


New Project

- Installed > Templates > Visual C++ > ATL
- .NET Framework 4.5 · Sort by: Default
- ATL Project — Visual C++
- Type: Visual C++
- A project that uses the Active Template Library

Name: ATLProject2
Location: N:\VS2012\
Solution name: ATLProject2
☑ Create directory for solution
☐ Add to source control

- Then in support option we just need to check in MFC and COM support.



ATL Project Wizard - ATLProject2

Application Settings

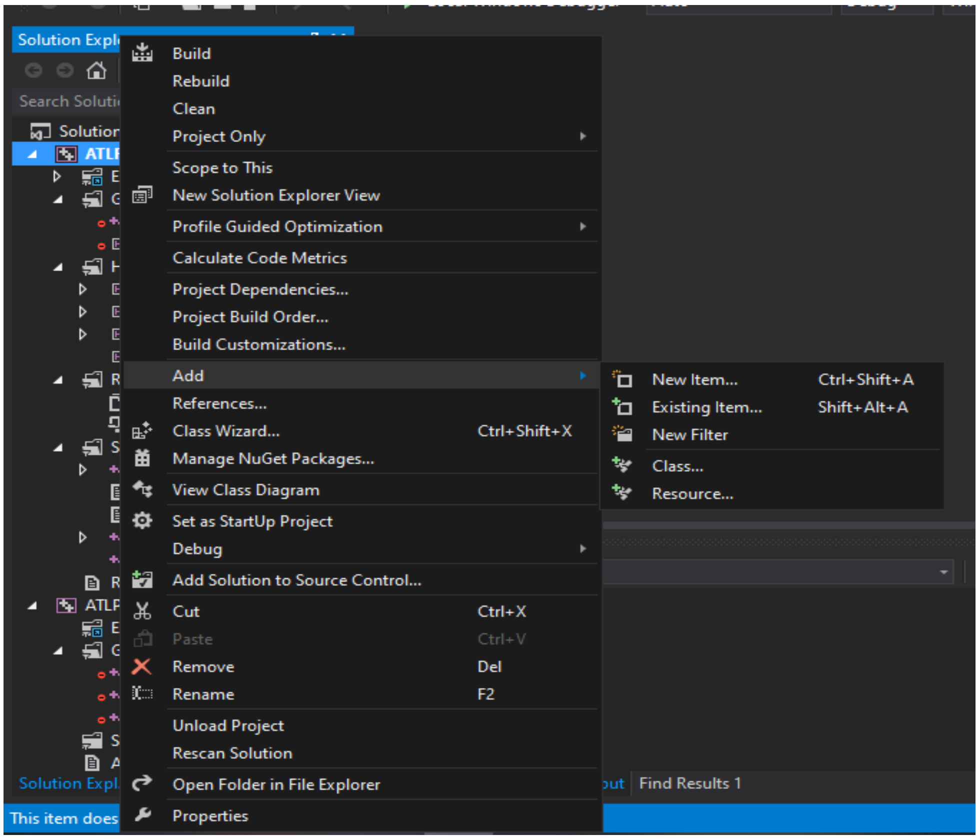
Application type:
- ● Dynamic-link library (DLL)
- ○ Executable (EXE)
- ○ Service (EXE)

Support options:
- ☐ Allow merging of proxy/stub code
- ☑ Support MFC
- ☑ Support COM+ 1.0
- ☐ Support component registrar
- ☑ Security Development Lifecycle (SDL) checks

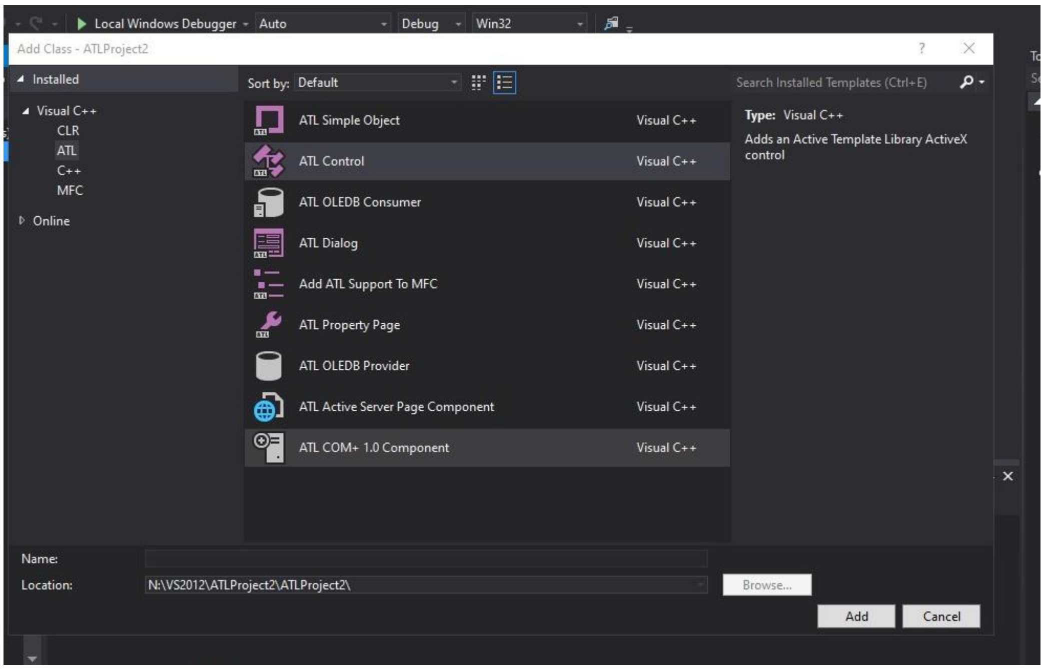
File type handler options:
File extension:
- ☐ Preview handler
- ☐ Thumbnail handler
- ☐ Search handler
- ☐ Generate ATL-based document

- After clicking on finishing button you see your sample ATL project. Now you need to insert ATL Control.

- For that just right click on Your project name and select ass below and in Add select Class



- Then select ATL Control option.

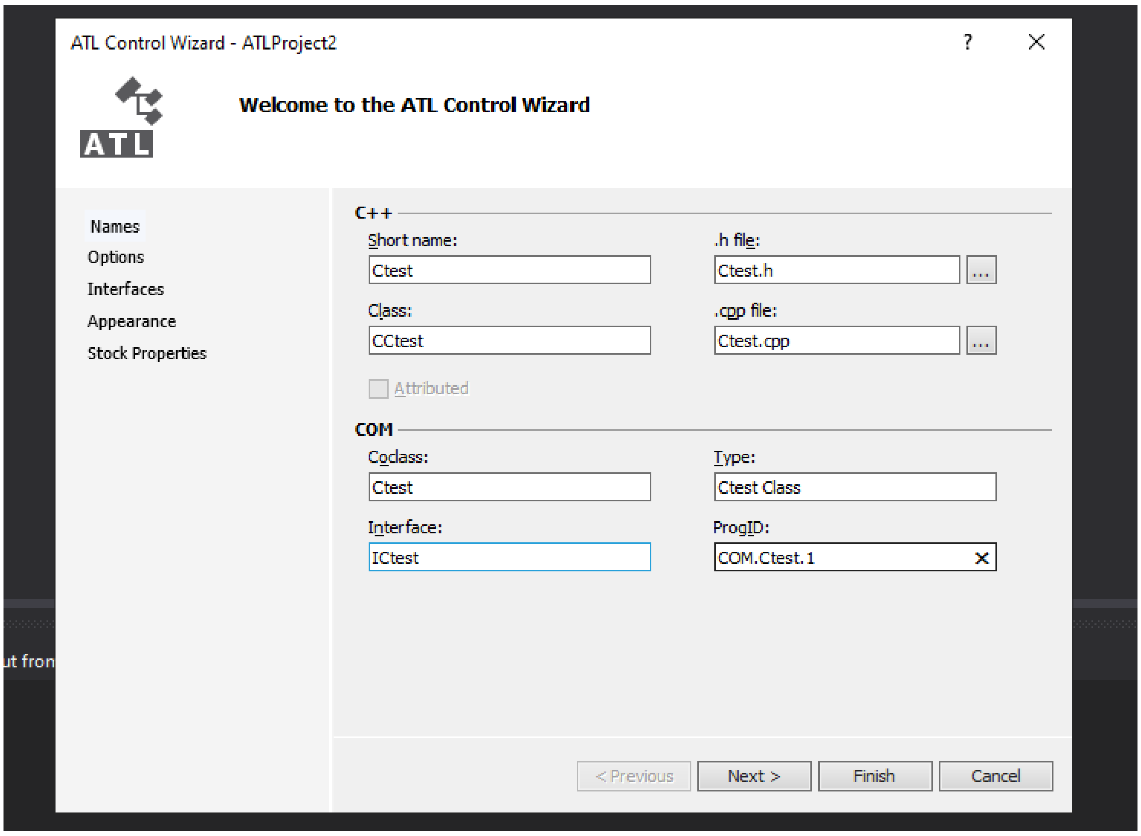

Local Windows Debugger
Auto
Debug
Win32
Add Class - ATLProject2
Installed
Visual C++
CLR
ATL
C++
MFC
Online
Search Installed Templates (Ctrl+E)
ATL Simple Object Visual C++
ATL Control Visual C++
ATL OLEDB Consumer Visual C++
ATL Dialog Visual C++
Add ATL Support To MFC Visual C++
ATL Property Page Visual C++
ATL OLEDB Provider Visual C++
ATL Active Server Page Component Visual C++
ATL COM+ 1.0 Component Visual C++
Browse...
Add
Cancel

Sort by: Default

Type: Visual C++
Adds an Active Template Library ActiveX control

Name:
Location: N:\VS2012\ATLProject2\ATLProject2\

- Add names for Classes , .h file CoClasses and ProgID like below example.


ATL Control Wizard - ATLProject2

Welcome to the ATL Control Wizard

C++

Short name: Ctest

.h file: Ctest.h

Class: CCtest

.cpp file: Ctest.cpp

Attributed

COM

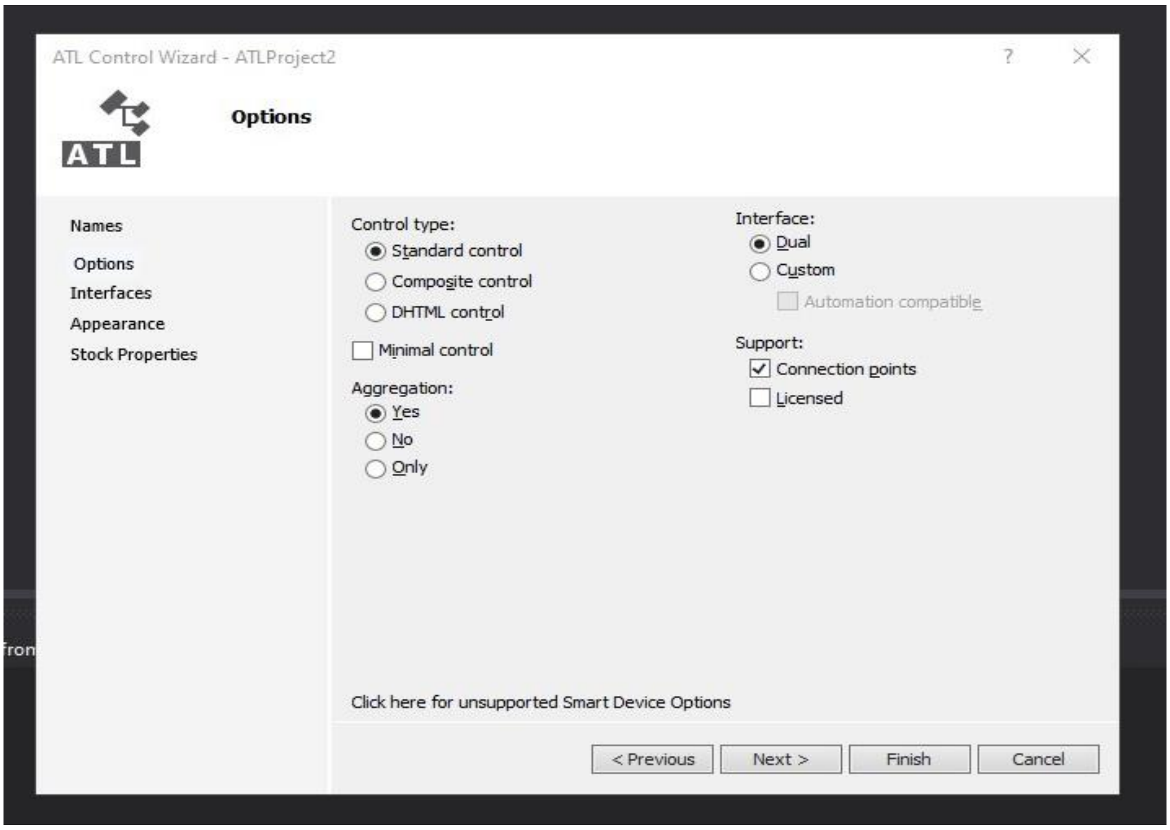
Coclass: Ctest

Type: Ctest Class

Interface: ICtest

ProgID: COM.Ctest.1

- Select following and click next.


ATL Control Wizard - ATLProject2

Options

Control type:
- [✓ Standard control]
- Composite control
- DHTML control

Minimal control

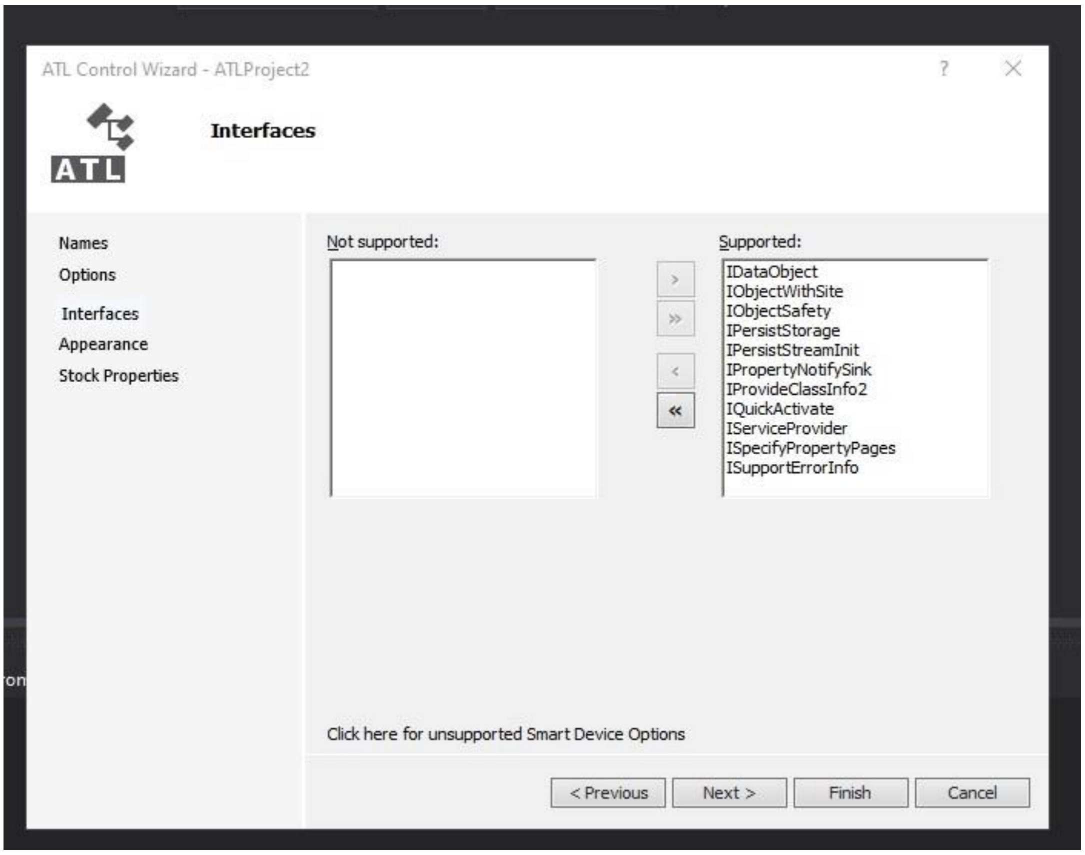
Aggregation:
- [✓ Yes]
- No
- Only

Interface:
- [✓ Dual]
- Custom
- Automation compatible

Support:
- [✓ Connection points]
- Licensed

Click here for unsupported Smart Device Options

- In interface include whichever you needed.



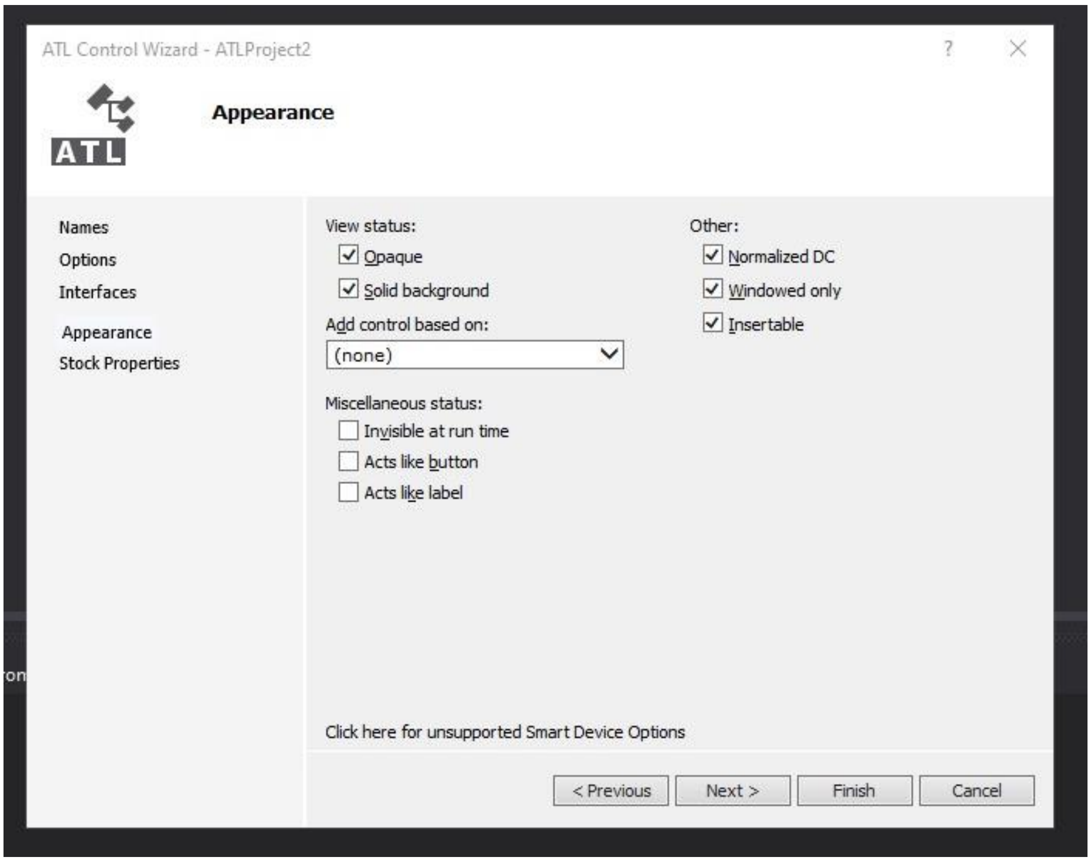
ATL Control Wizard - ATLProject2

Interfaces

Not supported:

Supported:
IDataObject
IObjectWithSite
IObjectSafety
IPersistStorage
IPersistStreamInit
IPropertyNotifySink
IProvideClassInfo2
IQuickActivate
IServiceProvider
ISpecifyPropertyPages
ISupportErrorInfo

Click here for unsupported Smart Device Options



ATL Control Wizard - ATLProject2

Appearance

View status:
- [✓] Opaque
- [✓] Solid background

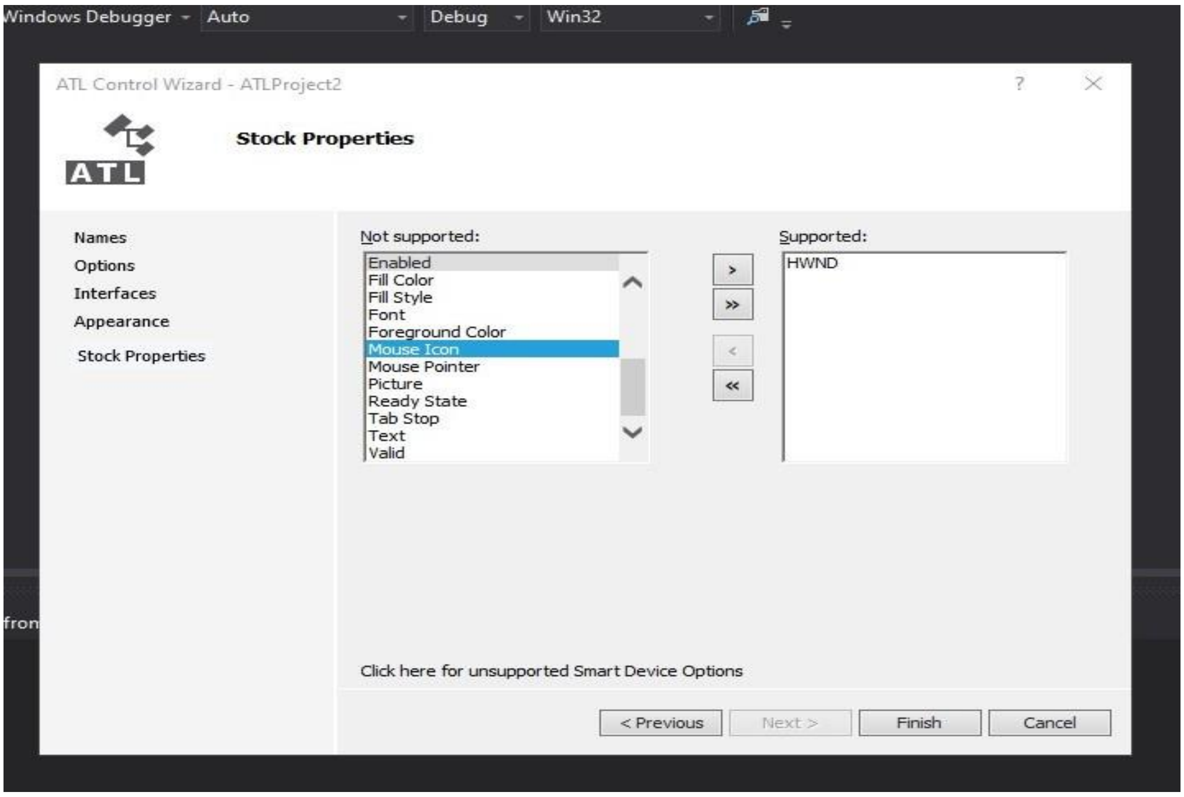
Add control based on: (none)

Miscellaneous status:
- [ ] Invisible at run time
- [ ] Acts like button
- [ ] Acts like label

Other:
- [✓] Normalized DC
- [✓] Windowed only
- [✓] Insertable

Click here for unsupported Smart Device Options

- Choose and finish This control wizard.



ATL Control Wizard - ATLProject2

Stock Properties

Not supported:

Enabled
Fill Color
Fill Style
Font
Foreground Color
Mouse Icon
Mouse Pointer
Picture
Ready State
Tab Stop
Text
Valid

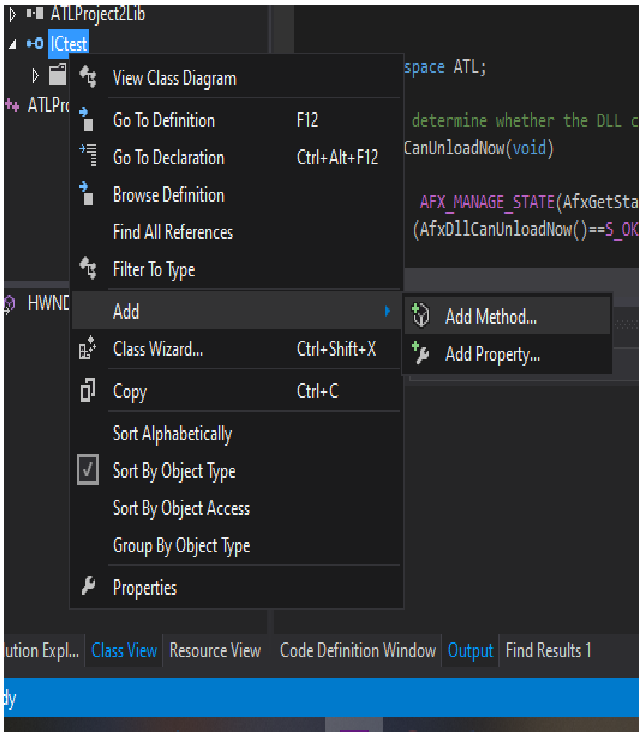
Supported:

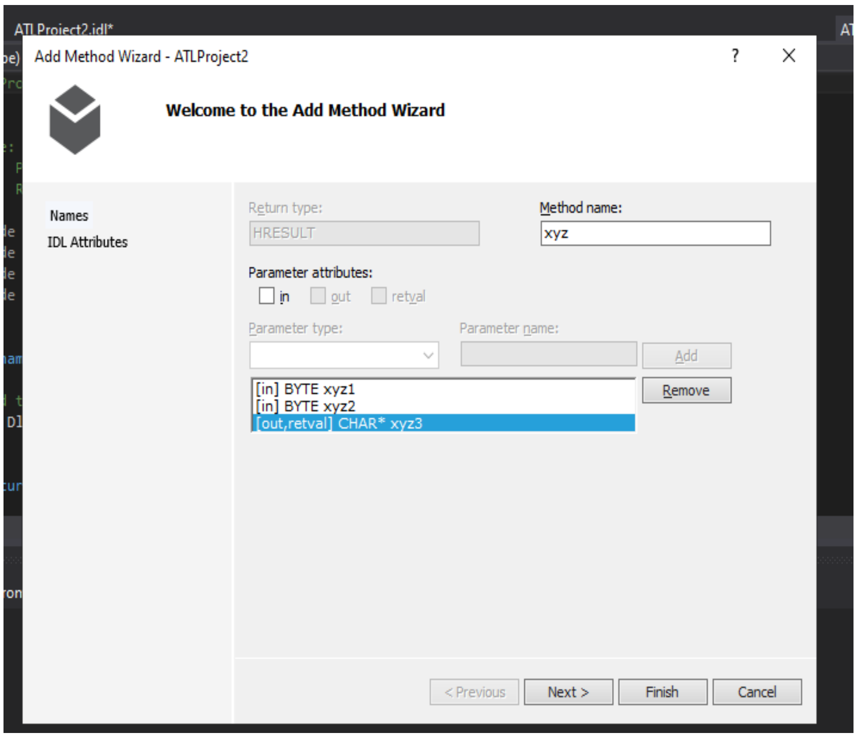
HWND

Click here for unsupported Smart Device Options

- When you finish it will create one Cpp file and one header file and create one entry in .idl file of this class. This is COM Component of which interface was created and when we have to add methods It add those methods in that cpp file for that interface.
- For entering method, we just need to go to class view and find interface name and with right click -> Add -> Add method.
- It will show wizard like in below screenshot in which you have to decide input, output parameters and name of method.
- Then after clicking on next for particular method it shows you id. when you will finish it in cpp file of control it adds one method with all those parameters you entered. You can see declaration of this method in header file. In .idl file one entry of method with 1st ID, parameters, parameter types that input, output or retrieval will be generated.
- Remember when we select pointer then and then it will enable option for output or retrieval.
- Then after building project you need to register that DLL in cmd(as Administrator) with Command **regsvr32 "Path"** or in properties option -> linker-> pre user redirection-> Choose Yes.

Sahil Sidapara(s219)



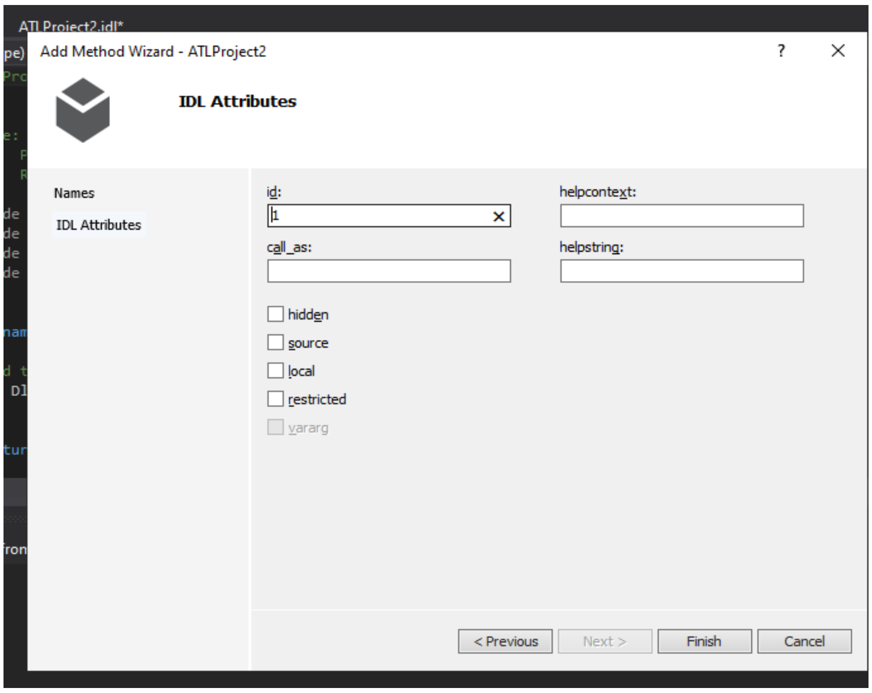

Add Method Wizard - ATLProject2

Welcome to the Add Method Wizard

Names
IDL Attributes

Return type: HRESULT
Method name: xyz
Parameter attributes: in out retval
Parameter type:
Parameter name:

```
[in] BYTE xyz1
[in] BYTE xyz2
[out,retval] CHAR* xyz3
```

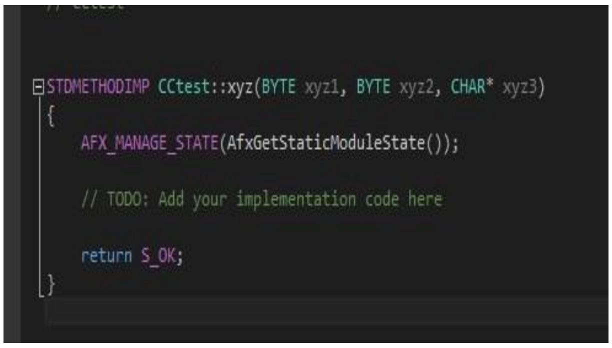

Add Method Wizard - ATLProject2

IDL Attributes

Names
IDL Attributes

id: 1
helpcontext:
call_as:
helpstring:

hidden
source
local
restricted
vararg

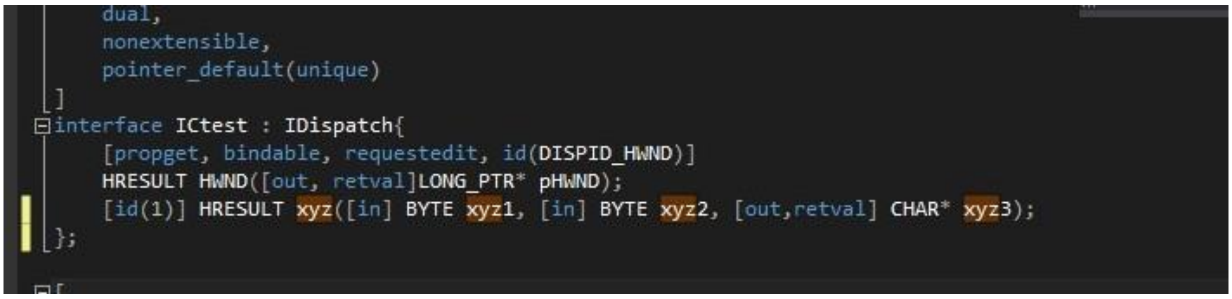


```
STDMETHODIMP CCtest::xyz(BYTE xyz1, BYTE xyz2, CHAR* xyz3)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    // TODO: Add your implementation code here

    return S_OK;
}
```

Ctest.cpp



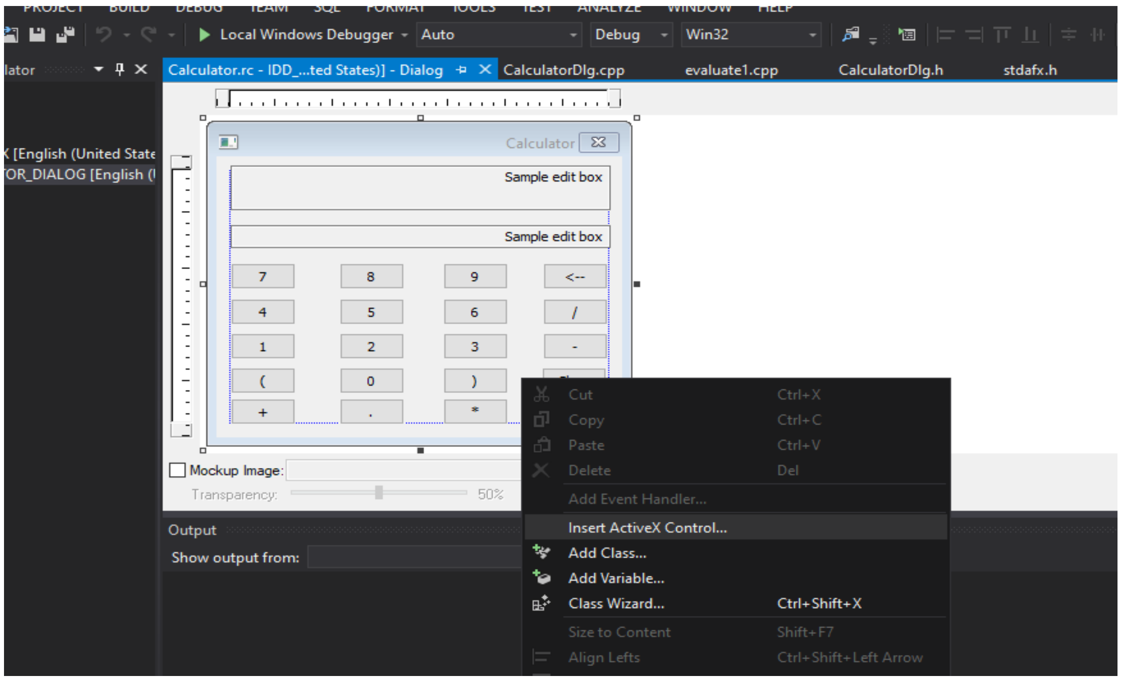
```
        dual,
        nonextensible,
        pointer_default(unique)
    ]
    interface ICtest : IDispatch{
        [propget, bindable, requestedit, id(DISPID_HWND)]
        HRESULT HWND([out, retval]LONG_PTR* pHWND);
        [id(1)] HRESULT xyz([in] BYTE xyz1, [in] BYTE xyz2, [out,retval] CHAR* xyz3);
    };
```

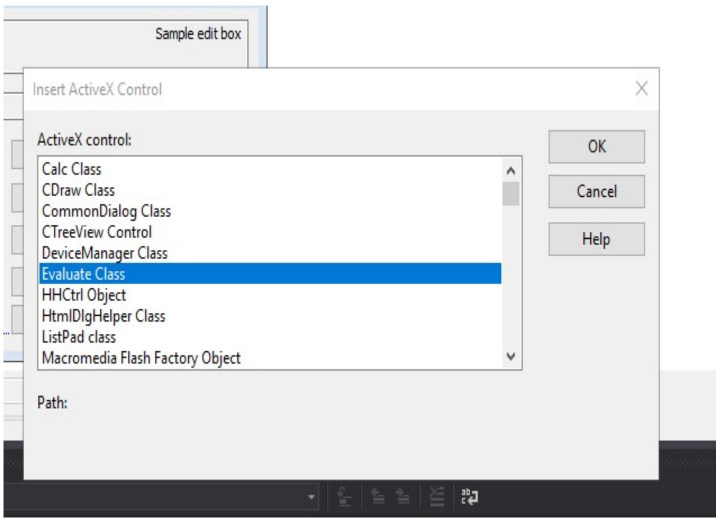
ATLProject2.idl

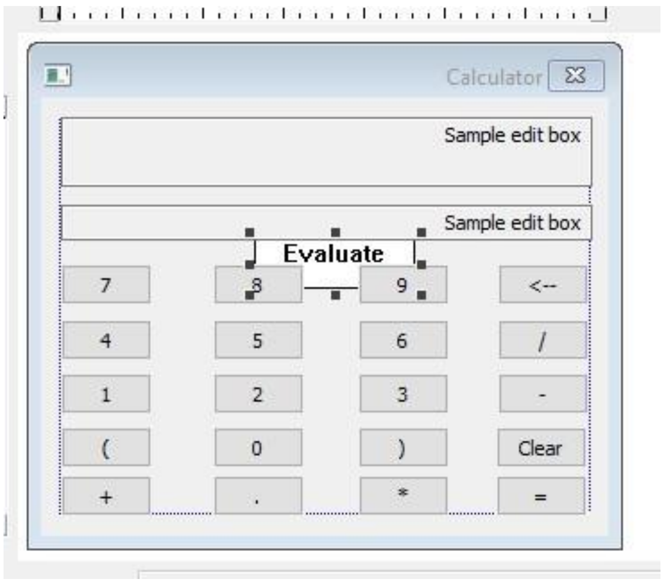
Load COM Component

Using ActiveX Control

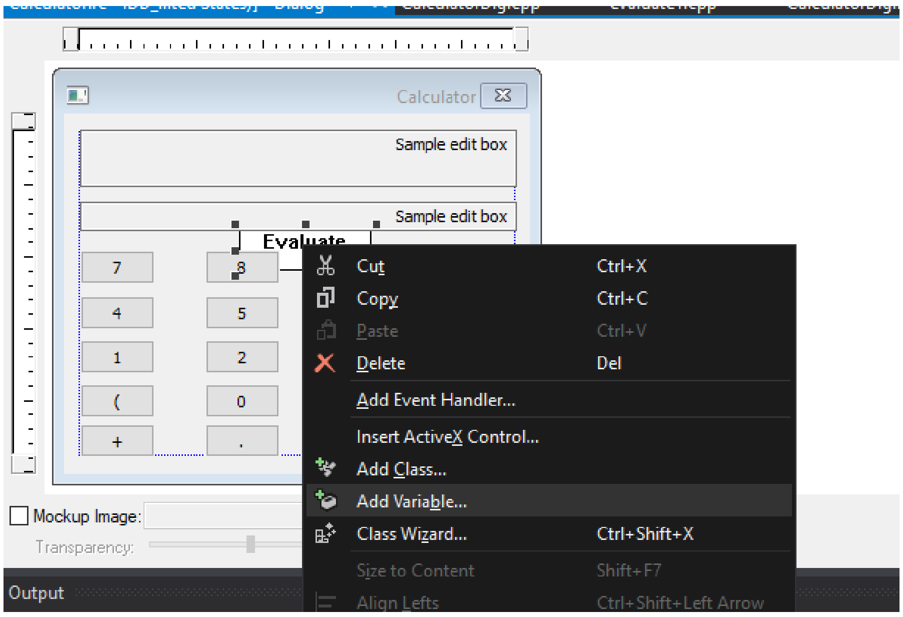
- Suppose I am making one calculator (MFC Dialog based Application) which has all functionalities in DLL file and validation, GUI in that Application then to make communication I have to load ActiveX in Dialog box. I need to insert ActiveX control in dialog. For that in GUI I just have to click Insert ActiveX Control option then select class. It will add Wrapper class automatically in MFC project with its .cpp and .h file.

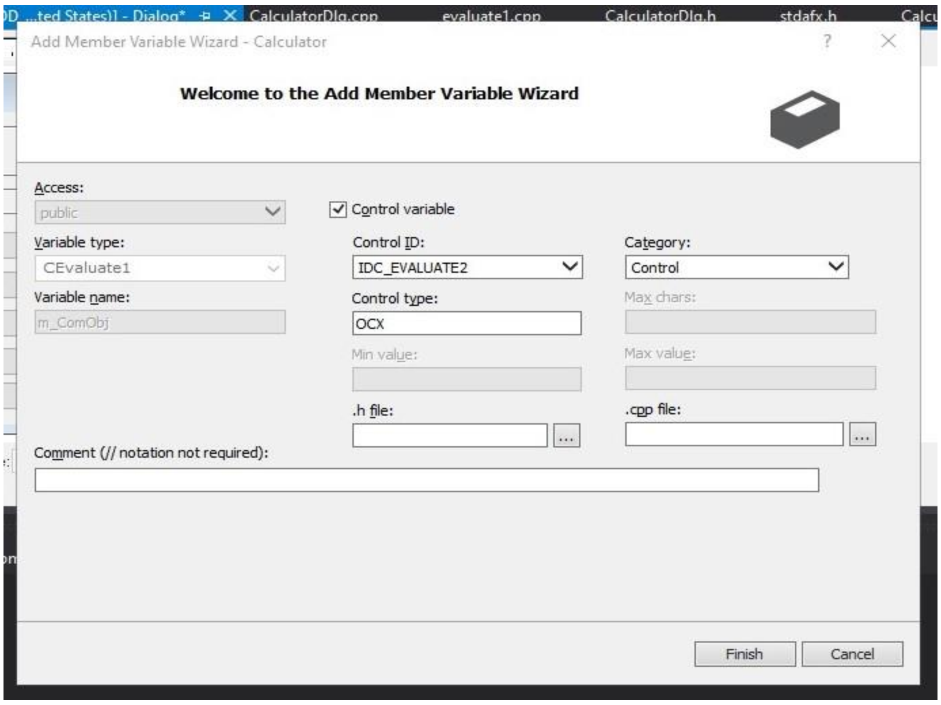






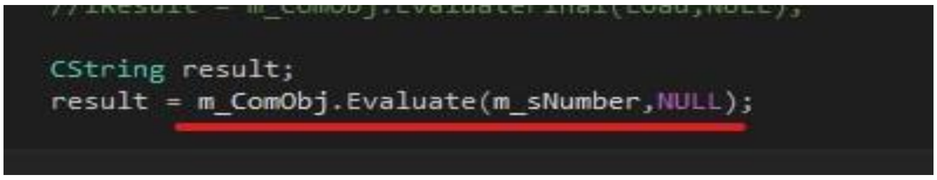
- Simple right click on that ActiveX and select Add variable for adding variable. It will create control variable through which we can call methods of DLL file.




Add Member Variable Wizard - Calculator

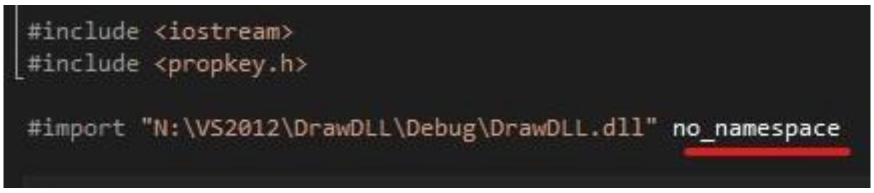
Welcome to the Add Member Variable Wizard

Access: public
Variable type: CEvaluate1
Variable name: m_ComObj
☑ Control variable
Control ID: IDC_EVALUATE2
Category: Control
Control type: OCX
Max chars:
Min value:
Max value:
.h file:
.cpp file:
Comment (// notation not required):


```
CString result;
result = m_ComObj.Evaluate(m_sNumber,NULL);
```

Using CoInitialize Method

- Suppose I am making one MDI Application which has some features to draw particular shapes, save and open those shapes again. I had created one COM DLL which include all functionalities of drawing.
- Now I want to load COM component in MDI and all method of drawing to draw in MDI application.
- So 1st I need to import that COM DLL. Like shown in below Screenshot.

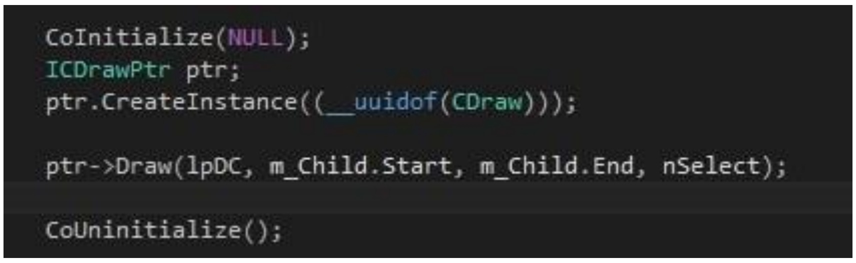


```
#include <iostream>
#include <propkey.h>

#import "N:\VS2012\DrawDLL\Debug\DrawDLL.dll" no_namespace
```

DrawView.Cpp

- Here no_namespace means it's namespace is not generated by compiler.
- You can also do this in another way. Just follow this path -> Properties -> VC++ dependencies -> Library Dependencies ->Add path and then simply #import "DLL name" no_namespace.
- Then just call CoInitialize method when you need to load COM Object like below. Here we will call it in DrawView.cpp



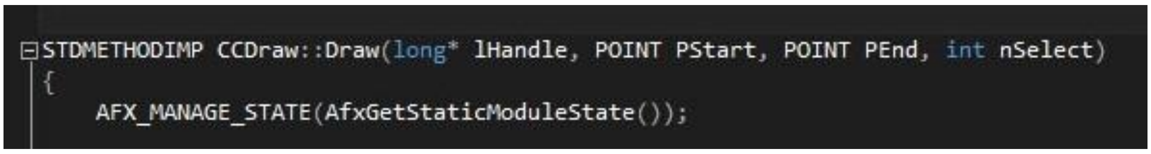
```
CoInitialize(NULL);
ICDrawPtr ptr;
ptr.CreateInstance((__uuidof(CDraw)));

ptr->Draw(lpDC, m_Child.Start, m_Child.End, nSelect);

CoUninitialize();
```

DrawView.cpp

- Here as shown in above screenshot we need to create pointer of interface. Here ICDrawPtr is pointer.
- Then ptr points to the instance of COM component(CDraw) or COM object which is created by CreateInstance method. Through this pointer we can access Draw method of COM DLL. Which is shown below: -



```
STDMETHODIMP CCDraw::Draw(long* lHandle, POINT PStart, POINT PEnd, int nSelect)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
```

CDraw.cpp