Introduction

A critical aspect of COM is how clients and servers interact. A COM client is whatever code or object gets a pointer to a COM server and uses its services by calling the methods of its interfaces. A COM server is any object that provides services to clients; these services are in the form of COM interface implementations that can be called by any client that is able to get a pointer to one of the interfaces on the server object.

There are two main types of servers, in-process and out-of-process. In-process servers are implemented in a dynamic linked library (DLL), and out-of-process servers are implemented in an executable file (EXE). Out-of-process servers can reside either on the local computer or on a remote computer. In addition, COM provides a mechanism that allows an in-process server (a DLL) to run in a surrogate EXE process to gain the advantage of being able to run the process on a remote computer.

The COM programming model and constructs have now been extended so that COM clients and servers can work together across the network, not just within a given computer. This enables existing applications to interact with new applications and with each other across networks with proper administration, and new applications can be written to take advantage of networking features.

COM client applications do not need to be aware of how server objects are packaged, whether they are packaged as in-process objects (in DLLs) or as local or remote objects (in EXEs). Distributed COM further allows objects to be packaged as service applications, synchronizing COM with the rich administrative and system-integration capabilities of Windows.

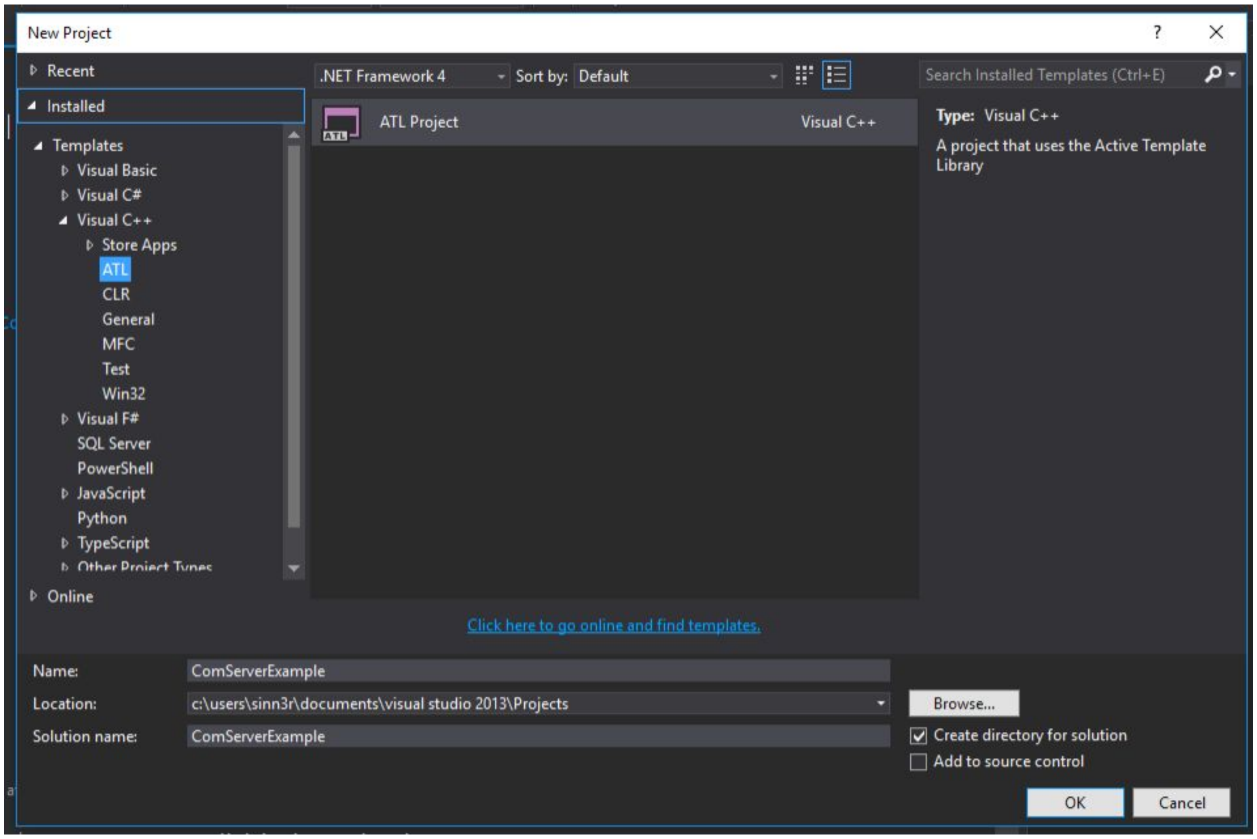
This documentation explains how to create a COM server and client using Visual Studio 2013 on Windows 10.

Documented by Wei Chen.

Building an In-Process COM Server

In this tutorial, we will create a COM object, with a method that prints “Hello World.”

First, create a new **ATL Project**:



Type: Visual C++
A project that uses the Active Template Library

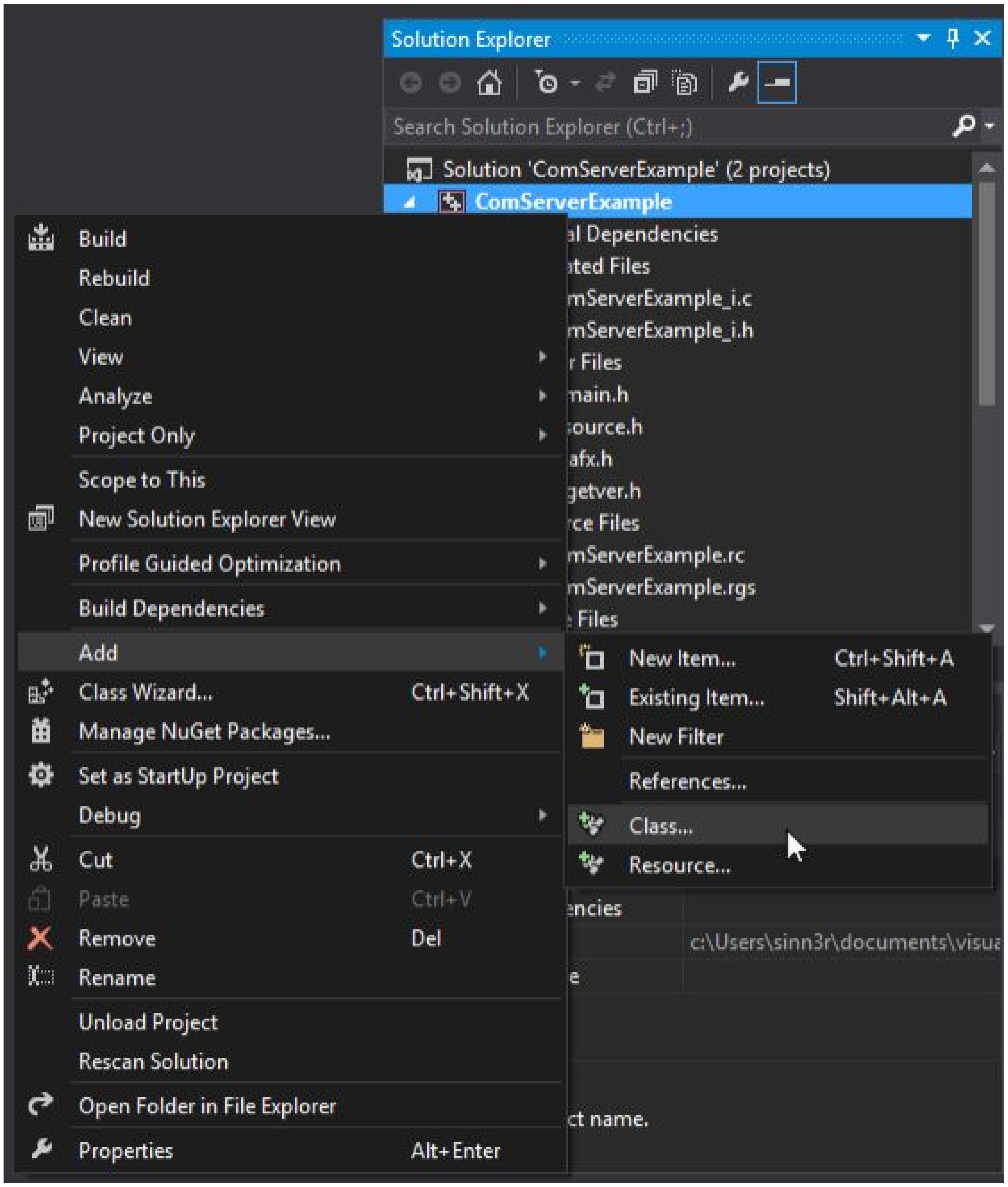
Name: ComServerExample
Location: c:\users\sinn3r\documents\visual studio 2013\Projects
Solution name: ComServerExample
[✓] Create directory for solution
[ ] Add to source control

In Application Settings, check these things:

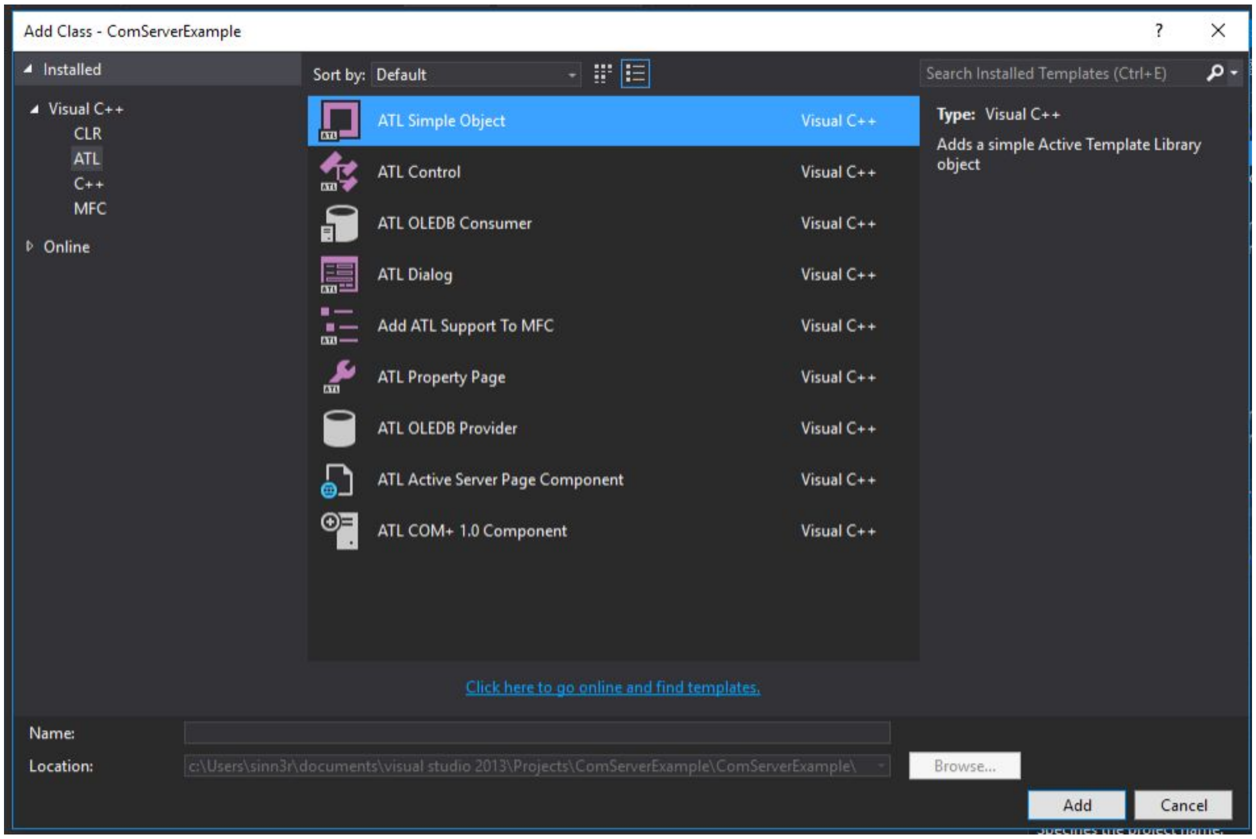
- Dynamic-link library (DLL)
- Support MFC
- Support COM+ 1.0

The reason you want to check the last two is because if you don’t, you will not be able to add a new class later.

In the **Solution Explorer** panel, right click on the **ComServerExample** solution -> **Add** -> **Class**:



The Class option will take you to a prompt where you can choose what type class you want to add. In here, click on **ATL** on the left, and then choose **ATL Simple Object**:


Add Class - ComServerExample
Installed
Visual C++
CLR
ATL
C++
MFC
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
Click here to go online and find templates.
Browse...
Add
Cancel

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++

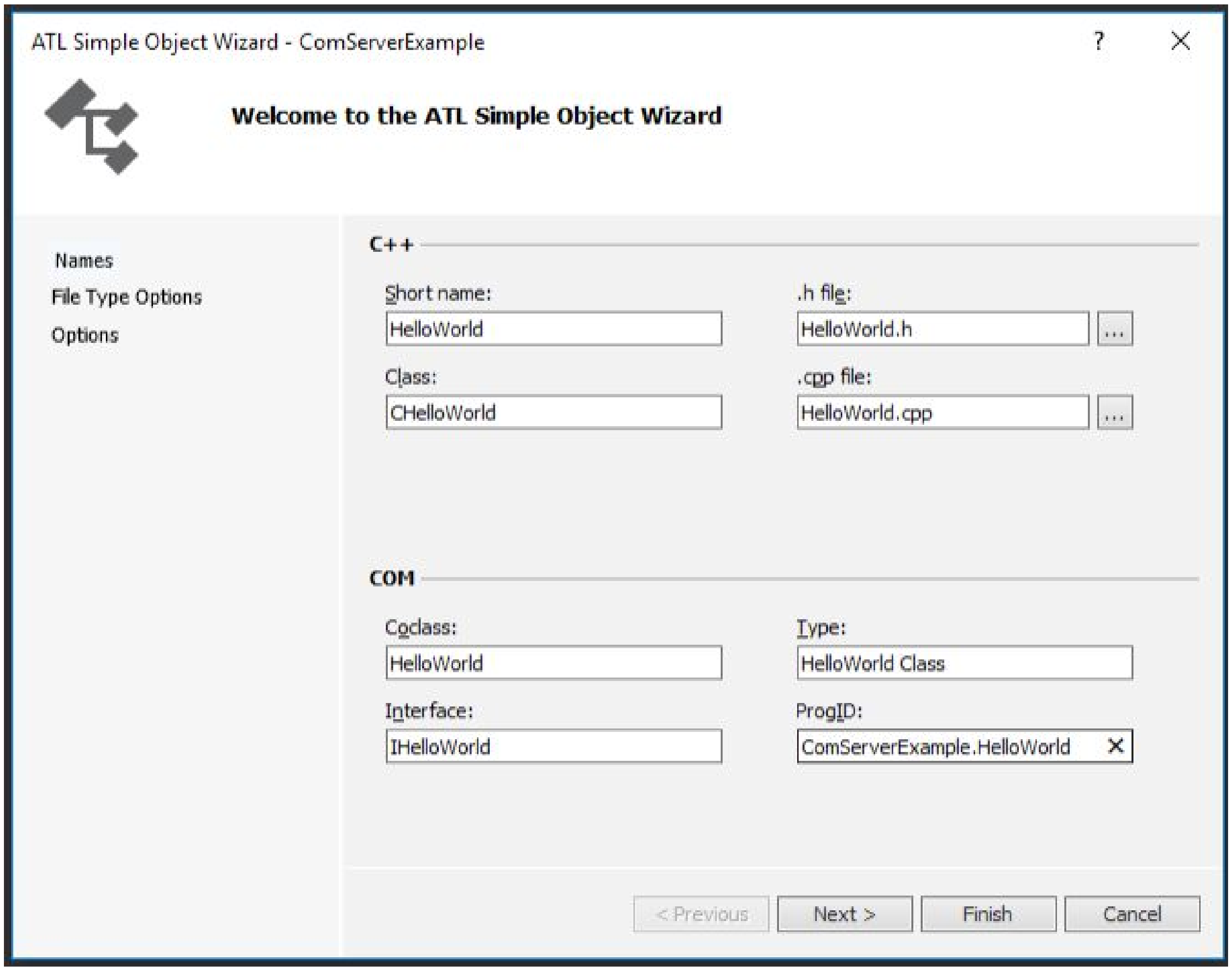
Type: Visual C++

Adds a simple Active Template Library object

Name:

Location: c:\Users\sinn3r\documents\visual studio 2013\Projects\ComServerExample\ComServerExample\

Next, you will be taken to the **ATL Simple Object Wizard**. In here, all you need to do is enter the **short name**, and then the program will fill out the rest. If you'd like, you can also provide a ProgID, but that's really optional:


ATL Simple Object Wizard - ComServerExample
Welcome to the ATL Simple Object Wizard
Names
File Type Options
Options
C++
COM
< Previous
Next >
Finish
Cancel

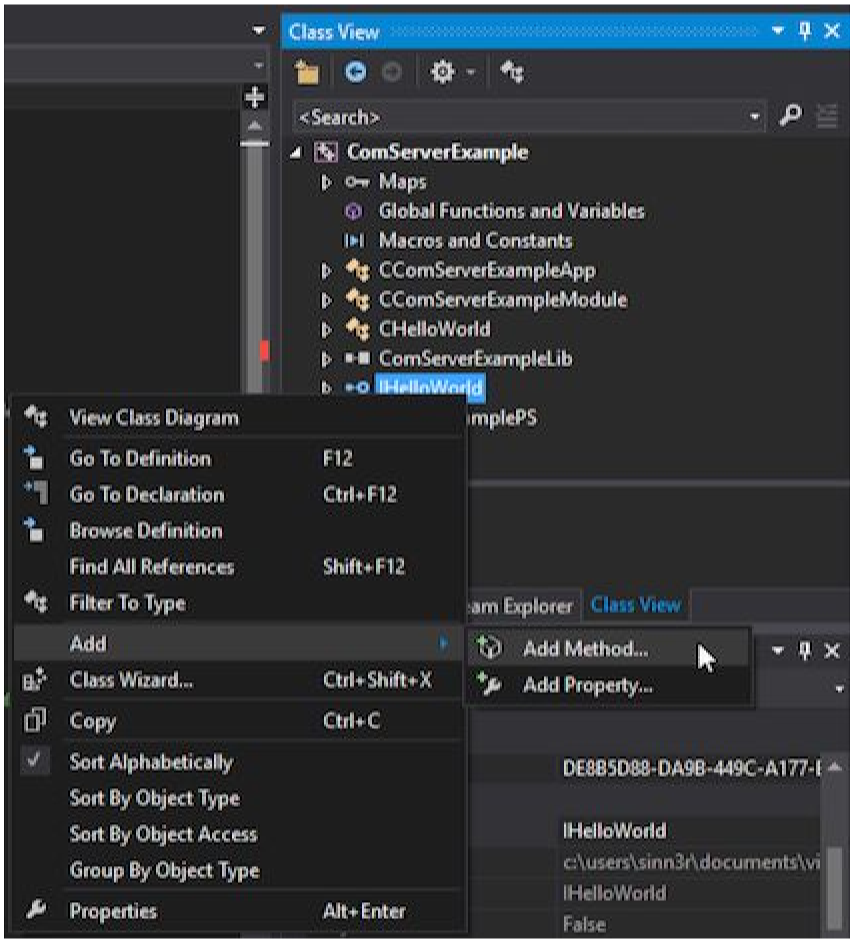
Short name: HelloWorld
.h file: HelloWorld.h
Class: CHelloWorld
.cpp file: HelloWorld.cpp
Coclass: HelloWorld
Type: HelloWorld Class
Interface: IHelloWorld
ProgID: ComServerExample.HelloWorld

After filling these out, you can click **Finish**.

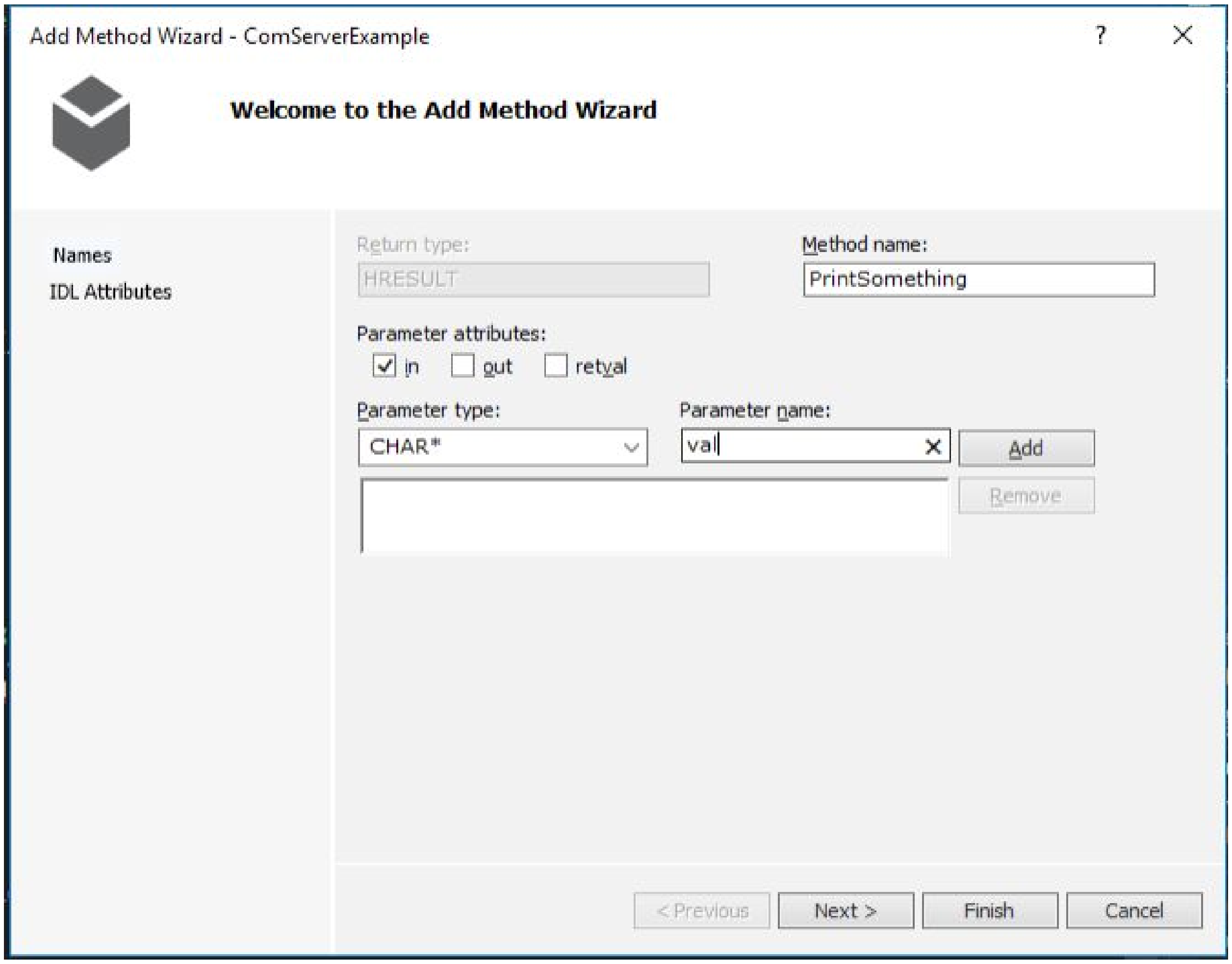
After the class is created, on the right, go to Class View. You will see there are two classes:

- ComServerExample
- ComServerExamplePS

Expand the first one (**ComServerExample**), and then right-click on the **IHelloWorld** interface, which will take you to the **Add Method Wizard**:



In the **Add Method Wizard**, add a method name, parameter attribute, type, and the name, which will basically create the skeleton of the method code:



Add Method Wizard - ComServerExample

Welcome to the Add Method Wizard

Return type: HRESULT

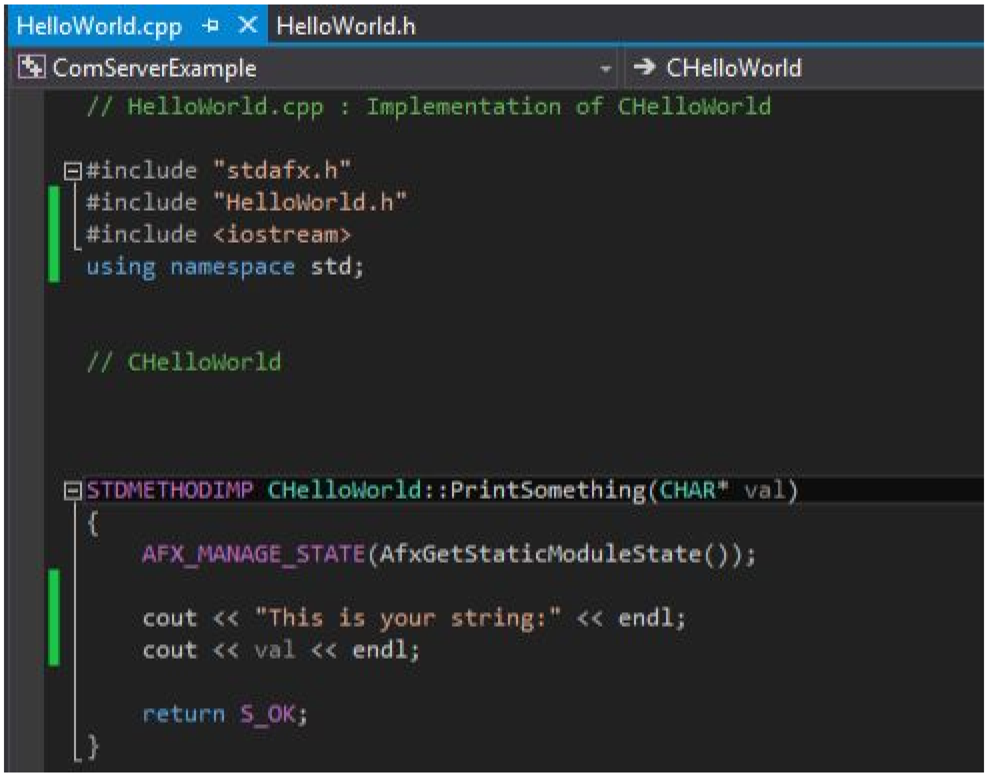
Method name: PrintSomething

Parameter attributes: [✓ in] out retval

Parameter type: CHAR*

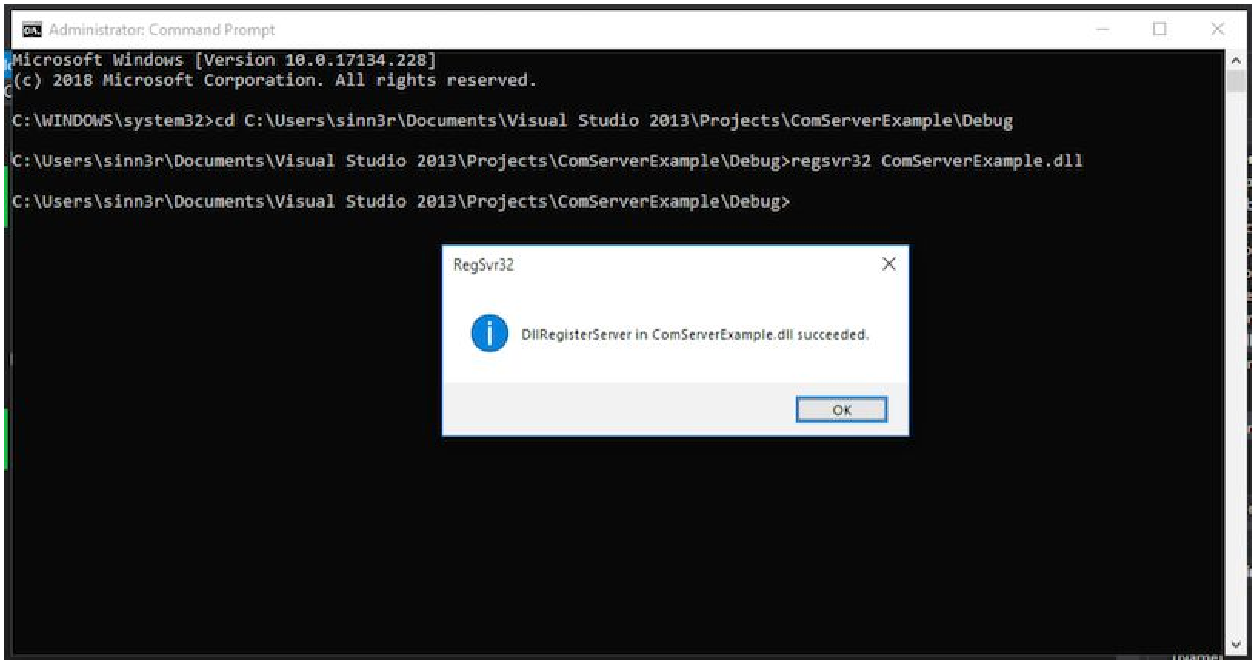
Parameter name: val

Go ahead and click Finish, and then code for that method will be generated. For this example, our method will basically just print whatever the val argument is:



```cpp
// HelloWorld.cpp : Implementation of CHelloWorld

#include "stdafx.h"
#include "HelloWorld.h"
#include <iostream>
using namespace std;


// CHelloWorld



STDMETHODIMP CHelloWorld::PrintSomething(CHAR* val)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    cout << "This is your string:" << endl;
    cout << val << endl;

    return S_OK;
}
```

Next, go to Build, and then build the solution. The DLL can be found in Visual Studio's Projects directory. Make sure to register it (as admin) like so:



```
Microsoft Windows [Version 10.0.17134.228]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\sinn3r\Documents\Visual Studio 2013\Projects\ComServerExample\Debug

C:\Users\sinn3r\Documents\Visual Studio 2013\Projects\ComServerExample\Debug>regsvr32 ComServerExample.dll

C:\Users\sinn3r\Documents\Visual Studio 2013\Projects\ComServerExample\Debug>
```

RegSvr32

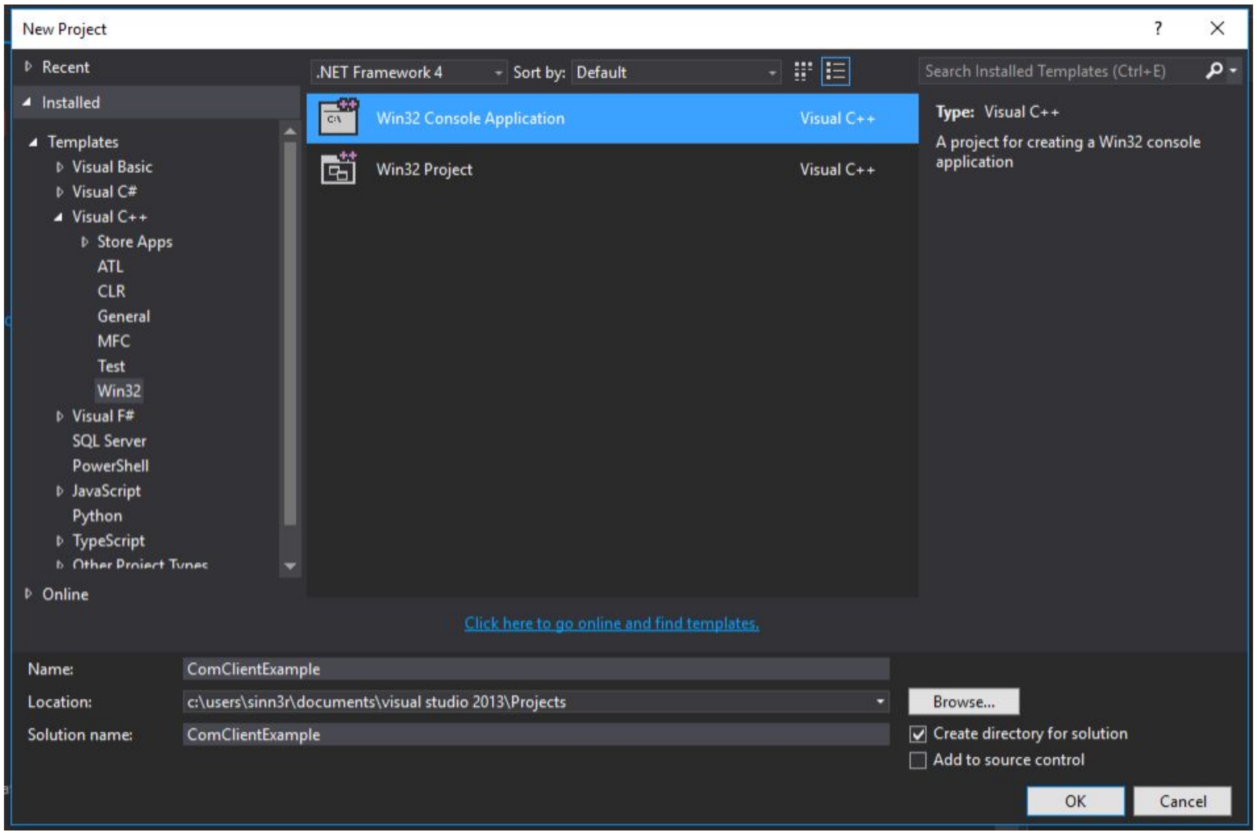
DllRegisterServer in ComServerExample.dll succeeded.

OK

And now the COM object is ready to use.

Building a COM Client

To build a client, we can create a **Win32 Console Application**:



Type: Visual C++

A project for creating a Win32 console application

Name: ComClientExample

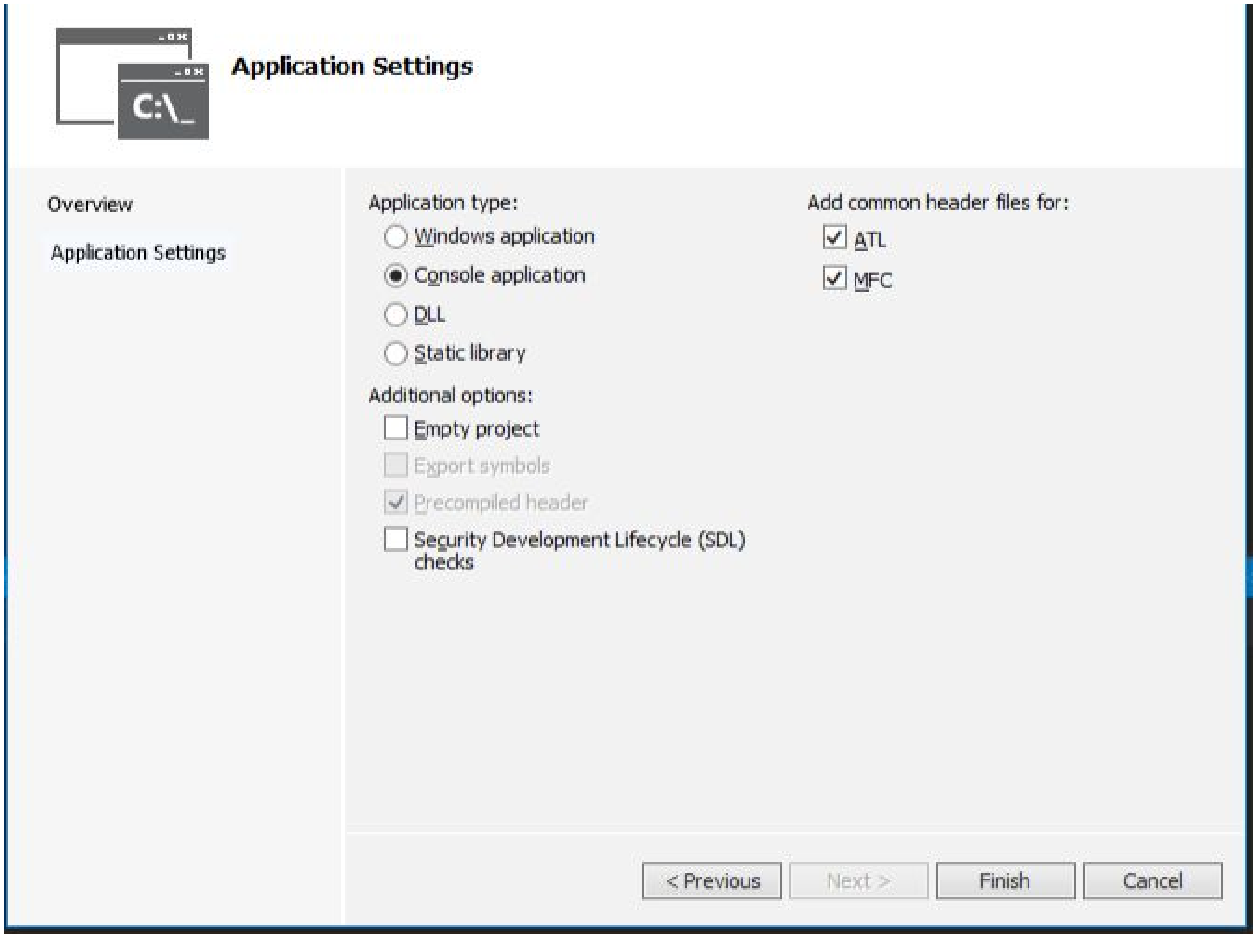
Location: c:\users\sinn3r\documents\visual studio 2013\Projects

Solution name: ComClientExample

[✓] Create directory for solution

[ ] Add to source control

In the Win32 Application Wizard, check **ATL** and **MFC** for common header files:



Application Settings

Overview

Application Settings

Application type:

- ◯ Windows application
- [✓ Console application]
- ◯ DLL
- ◯ Static library

Additional options:

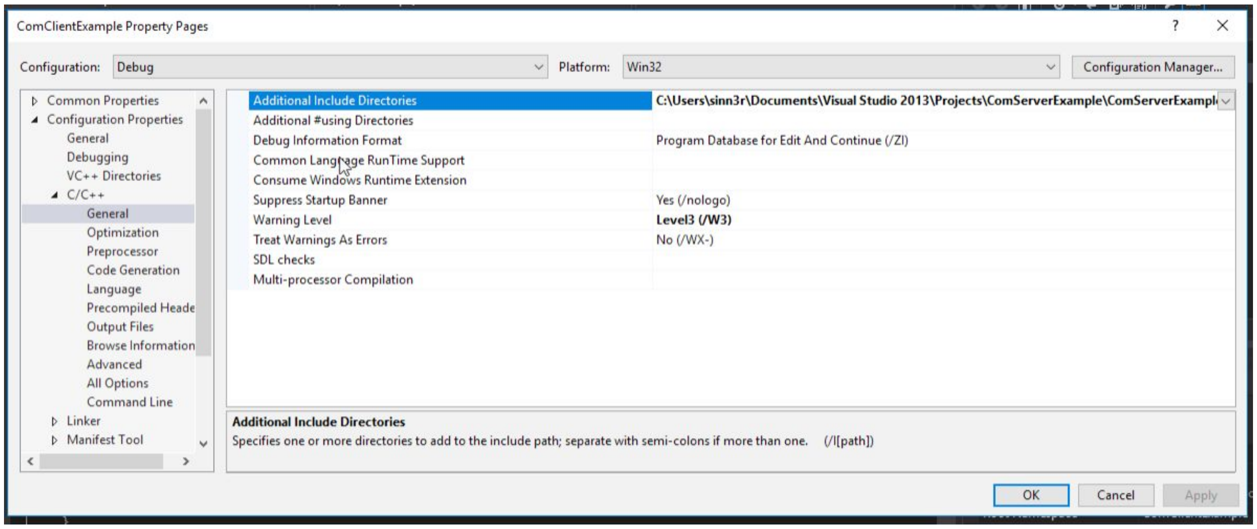
- ☐ Empty project
- ☐ Export symbols
- ☑ Precompiled header
- ☐ Security Development Lifecycle (SDL) checks

Add common header files for:

- ☑ ATL
- ☑ MFC

After the project is created, on the right panel, right-click on the solution properties, and then add this in the **Additional Include Directories** setting (the directory path should be where you see the *_i.c, *_i.h, and the cpp file for the interface, etc):

```
C:\Users\sinn3r\Documents\Visual Studio
2013\Projects\ComServerExample\ComServerExample;%(AdditionalIncludeDir
ectories)
```



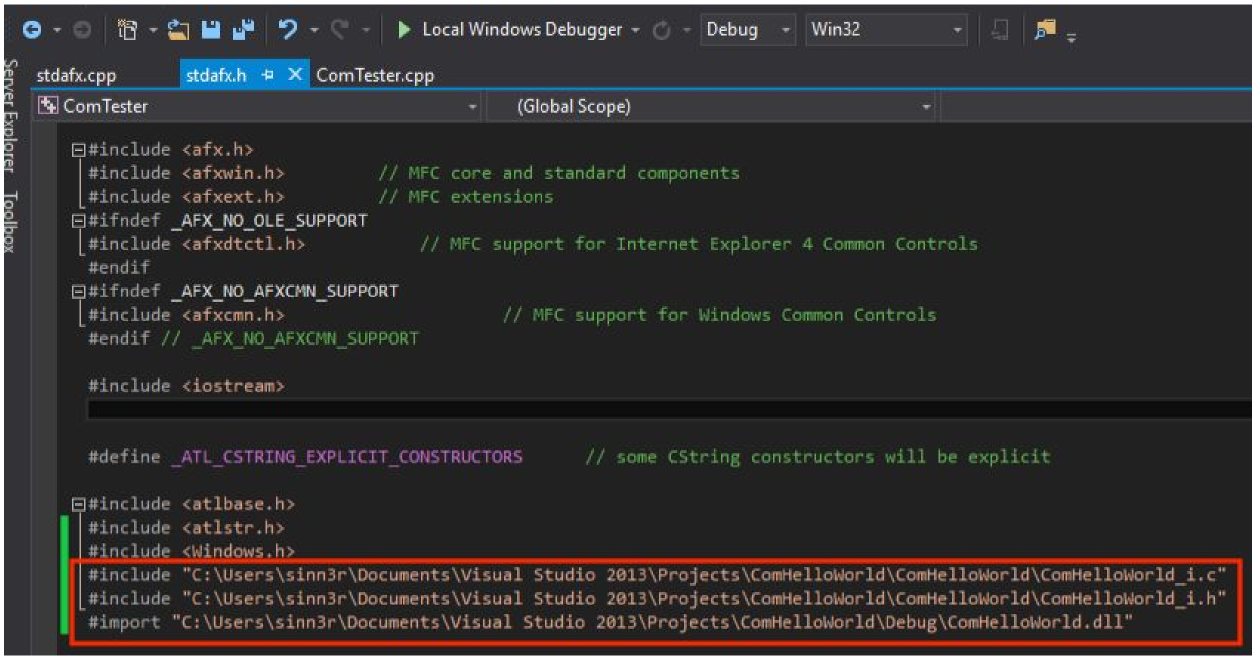
ComClientExample Property Pages

Configuration: Debug — Platform: Win32

Additional Include Directories	C:\Users\sinn3r\Documents\Visual Studio 2013\Projects\ComServerExample\ComServerExampl
Additional #using Directories	
Debug Information Format	Program Database for Edit And Continue (/ZI)
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	Level3 (/W3)
Treat Warnings As Errors	No (/WX-)
SDL checks	
Multi-processor Compilation	

Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])

Next, go to the **stdafx.h** file, and add the following for the ComServerSample object:



```
#include <afx.h>
#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#ifndef _AFX_NO_OLE_SUPPORT
#include <afxdtctl.h>           // MFC support for Internet Explorer 4 Common Controls
#endif
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT

#include <iostream>


#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS      // some CString constructors will be explicit

#include <atlbase.h>
#include <atlstr.h>
#include <Windows.h>
#include "C:\Users\sinn3r\Documents\Visual Studio 2013\Projects\ComHelloWorld\ComHelloWorld\ComHelloWorld_i.c"
#include "C:\Users\sinn3r\Documents\Visual Studio 2013\Projects\ComHelloWorld\ComHelloWorld\ComHelloWorld_i.h"
#import "C:\Users\sinn3r\Documents\Visual Studio 2013\Projects\ComHelloWorld\Debug\ComHelloWorld.dll"
```

And finally, add the following function in ComTester.cpp to call the method in the COM object:

```
void DoTest() {
      CoInitialize(NULL);
      IClassFactory* factory;
      IUnknown* pIunk;
      IHelloWorld* obj; // The HelloWorld interface from
ComServerExample
      HRESULT r;

      r = CoGetClassObject(CLSID_HelloWorld, CLSCTX_INPROC, NULL,
IID_IClassFactory, (void**)&factory);
      if (!SUCCEEDED(r)) {
            cout << "Failed to do CoGetClassObject" << endl;
            return;
      }

      r = factory->CreateInstance(NULL, IID_IUnknown, (void**)&pIunk);
      if (!SUCCEEDED(r)) {
            cout << "Failed to do CreateInstance" << endl;
            return;
      }

      r = pIunk->QueryInterface(IID_IHelloWorld, (void**)&obj);
      if (!SUCCEEDED(r)) {
            cout << "Failed to do QueryInterface" << endl;
            return;
      }

      cout << "Calling method" << endl;
      obj->Test("Hello World");
      obj->Release();
      factory->Release();
}
```

References

- Building a local COM server and client: A step by step example
- Component Object Model in C++
- Calling COM DLLs from Console Applications
- How to use C++ COM DLL in C++ Win32 Console Application
- A Beginner Tutorial for Writing Simple COM/ATL DLL for VS2012
- Introduction to COM - What It Is and How to Use It
- How do you create a COM DLL in Visual Studio 2018