# Programming Study
# COM Interop

Sungwoo Nam

2019.1.30

# Using .NET assembly at C++

- .NET assembly exposed as COM
- C++ import .NET assembly using ATL COM interface
- Data should be marshalled to/from .NET

# Foo Bar example

```
namespace MyCompany.Lib
{
    [ComVisible(true)]
    [Guid("71D1B3CD-3CE8-46B8-BF36-329543717F03")]
    [InterfaceType(ComInterfaceType.InterfaceIsDual)]
    public interface IFoo
    {
        int Bar(int arg);
    }

    [ComVisible(true)]
    [Guid("6F1C4CFC-4BBF-46A1-A9DD-98A4D83691D1")]
    [ClassInterface(ClassInterfaceType.None)]
    [ProgId("MyCompany.Lib.Foo")]
    public class Foo : IFoo
    {
        public int Bar(int arg)
        {
            return 42 + arg;
        }
    }
}
```

```
#include <atlbase.h>
#include <atlcom.h>
#include <atlstr.h>

#import "...\MyCompany.Lib.tlb" named_guids


{
  using namespace MyCompany_Lib;
  IFooPtr foo(__uuidof(Foo));
  long ret = foo->Bar(42);
  assert(ret == 84);
}
```

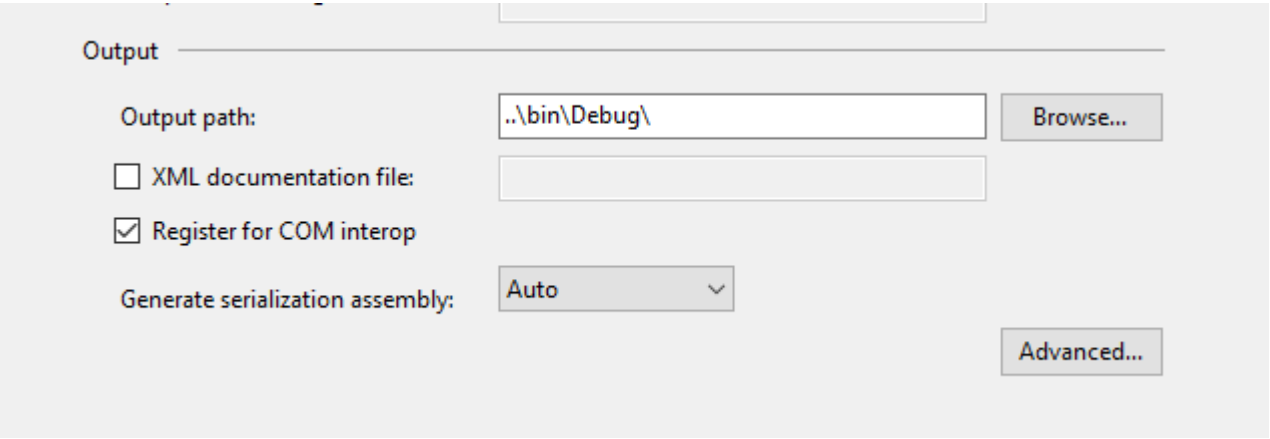
# tlb , tli

```
namespace MyCompany.Lib {

struct __declspec(uuid("71d1b3cd-3ce8-46b8-bf36-329543717f03")) IFoo;

_COM_SMARTPTR_TYPEDEF(IFoo, __uuidof(IFoo));

struct __declspec(uuid("71d1b3cd-3ce8-46b8-bf36-329543717f03"))
IFoo : IDispatch
{
    long Bar (long arg );

    virtual HRESULT __stdcall raw_Bar ( long arg, long * pRetVal ) = 0;
}

struct __declspec(uuid("6f1c4cfc-4bbf-46a1-a9dd-98a4d83691d1")) Foo;

#include "...\MyCompany.Lib.tli"
```

```
inline long IFoo::Bar ( long arg ) {
    long _result = 0;
    HRESULT _hr = raw_Bar(arg, &_result);
    if (FAILED(_hr)) _com_issue_errorex(_hr, this, __uuidof(this));
    return _result;
}
```

# Build option

- Start VisualStudio with admin
- Register for COM interop




- Or use tlbimp.exe
- Sign .NET assembly to deploy

# ComInterfaceType

```
...
[InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
public interface IFoo
{
...

...
[InterfaceType(ComInterfaceType.InterfaceIsIDispatch)]
public interface IFoo
{
...

...
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
public interface IFoo
...
```

```
class IUnknown {
  virtual HRESULT QueryInterface( REFIID riid, __RPC_FAR *ppvOb
ject) = 0;
  virtual ULONG AddRef( void) = 0;
  virtual ULONG Release( void) = 0;
}

class IDispatch : public IUnknown {
  virtual HRESULT GetTypeInfoCount(UINT *pctinfo) = 0;
  virtual HRESULT GetTypeInfo( ...) = 0;
  virtual HRESULT GetIDsOfNames( ... ) = 0;
  virtual HRESULT Invoke( ... ) = 0;
}


// tlb
IFoo : IUnknown
{

IFoo : IDispatch
{

IFoo : IDispatch
{
```

# ClassInteface Type

```
[ClassInterface(ClassInterfaceType.None)]
public class Foo : IFoo
{
...


[ClassInterface(ClassInterfaceType.AutoDual)]
public class ProgressBase
{
```

```
// tlb

struct __declspec(uuid("6f1c4cfc-4bbf-46a1-a9dd-98a4d83691d1"))
Foo;



struct __declspec(uuid("97754ee6-048b-3d1d-a477-deb25d4bc983"))
_ProgressBase : IDispatch
{
    __declspec(property(get=GetToString))
    _bstr_t ToString;

    _bstr_t GetToString ( );
    VARIANT_BOOL Equals ( const _variant_t & obj );
    long GetHashCode ( );
    _TypePtr GetType ( );

...
```

# Property

```
public interface IFoo
{
    int Koo { get; set; }
}

public class Foo : IFoo
{
    public int Koo
    {
        get;
        set;
    }
}
```

```
IFooPtr foo(__uuidof(Foo));

foo->PutKoo(42);
ret = foo->GetKoo();
assert(ret == 42);

foo->Koo = 11;
assert(foo->Koo == 11);
ret = foo->GetKoo();
assert(ret == 11);
```

# Exception

```
public class Foo : IFoo
{
    public int Bar(int arg)
    {
        if( arg != 42 )
        {
            throw new ArgumentException(
               "Only 42 allowed");
        }

        return 42 + arg;
    }

}
```

```
IFooPtr foo(__uuidof(Foo));

try
{
  ret = foo->Bar(43);
}
catch (_com_error& ex)
{
  cout << ex.Source() << " ," << ex.Description() << endl;
}


> MyCompany.Lib , Only 42 allowed
```

# Prog ID

```
namespace MyCompany.Lib
{
    [ComVisible(true)]
    [Guid("71D1B3CD-3CE8-46B8-BF36-329543717F03")]
    [InterfaceType(ComInterfaceType.InterfaceIsDual)]
    public interface IFoo
    {
        int Bar(int arg);
    }

    [ComVisible(true)]
    [Guid("6F1C4CFC-4BBF-46A1-A9DD-98A4D83691D1")]
    [ClassInterface(ClassInterfaceType.None)]
    [ProgId("MyCompany.Lib.Foo")]
    public class Foo : IFoo
    {
        public int Bar(int arg)
        {
            return 42 + arg;
        }
    }
}
```

```
IFooPtr foo = IFooPtr(L"MyCompany.Lig.Foo");
ret = foo->Bar(42);
assert(ret == 84);
```

# Marshalling

```
[ComVisible(true)]
[Guid("6A56D671-70DA-481A-AA9C-080B97D016D6")]
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
public interface IImage
{
    byte[] Data { get; }
    int Width { get; }
    int Height { get; }
    byte GetPixel(int x, int y);
}

internal class ImageBase : IImage
{
    public byte[] Data { get; set; }
    public int Width { get; set; }
    public int Height { get; set; }
    public byte GetPixel(int x, int y)
    {
        return Data[y * Width + x];
    }
}

[ComVisible(true)]
[Guid("9B5F50F0-66AB-40E1-8150-3E3688D297A7")]
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
public interface IImageFactory
{
    IImage CreateImage(
      IntPtr data, int width, int height);
}
```

# Marshalling - Continued

```
[ComVisible(true)]
[Guid("9B5F50F0-66AB-40E1-8150-3E3688D297A7")]
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
public interface IImageFactory
{
    IImage CreateImage(
      IntPtr data, int width, int height);
}

[ComVisible(true)]
[Guid("8F5BB630-4736-4099-8F93-B23AFDC2AF2D")]
[ClassInterface(ClassInterfaceType.None)]
public class ImageFactory : IImageFactory
{
     public IImage CreateImage(
        IntPtr data, int width, int height )
     {
        int size = width * height;
        byte[] managedData = new byte[size];
        Marshal.Copy(data, managedData, 0, size);
        return new ImageBase {
           Data = managedData,
           Width = width,
           Height = height
        };
    }
}
```

```
vector<uint8_t> m(320 * 240, 0);
m[24 + 42 * 320] = 0x42;

IImageFactoryPtr fab(__uuidof(ImageFactory));
IImagePtr image = fab->CreateImage_2(
   (LONG)m.data(), 320, 240);

uint8_t pixelValue = image->GetPixel(24, 42);
assert(pixelValue == 0x42);
```

# Enum

```
[ComVisible(true)]
[Guid("DD6EF614-CA16-43C1-9FD8-CBA386FD43C1")]
public enum AlgorithmTarget
{
    General = 0,
    Scratch = 1,
    Brightness = 2,
    Focus = 3,
}

[ComVisible(true)]
[Guid("FDE54767-12B4-4084-8868-F0033CE9ABD5")]
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
public interface IAlgorithm
{
    AlgorithmTarget Target { get; }
}

[ComVisible(true)]
[Guid("24A6E633-5911-4948-866A-CAF357D22BE2")]
[ClassInterface(ClassInterfaceType.None)]
public class ThresholdAlgorithm : IAlgorithm
{
    public AlgorithmTarget Target {
        get {
            return AlgorithmTarget.Brightness;
        }
    }
}
```

```
// tlb
enum __declspec(uuid("dd6ef614-ca16-43c1-9fd8-cba386fd43c1")
)
AlgorithmTarget
{
    AlgorithmTarget_General = 0,
    AlgorithmTarget_Scratch = 1,
    AlgorithmTarget_Brightness = 2,
    AlgorithmTarget_Focus = 3
};


// cpp
IAlgorithmPtr algo(__uuidof(ThresholdAlgorithm));
assert(algo->Target == AlgorithmTarget_Brightness);
```

# Struct

```
[ComVisible(true)]
[Guid("2672A17B-0D1A-4C8C-9922-D7E78A746D8E")]
public struct Defect
{
    public int X;
    public int Y;
    public int Area;
}

...
public interface IAlgorithm
{
    int GetDefectCount();
    void FillDefect(int index, ref Defect d);
    Defect GetDefect(int index);
}

...
public class ThresholdAlgorithm : IAlgorithm
{
    public int GetDefectCount() { return 2; }
    public Defect GetDefect(int index)
    {
        return new Defect { X = 3, Y = 2, Area = 1 };
    }
    public void FillDefect(int index, ref Defect d)
    {
        d.X = 1;
        d.Y = 2;
        d.Area = 3;
    }
}
```

```
// tlb
#pragma pack(push, 4)
struct __declspec
(uuid("2672a17b-0d1a-4c8c-9922-d7e78a746d8e"))
Defect
{
    long x;
    long y;
    long Area;
};



IAlgorithmPtr algo(__uuidof(ThresholdAlgorithm));
assert(algo->GetDefectCount() == 2);

Defect d = algo->GetDefect(0);
assert(d.x == 3);
assert(d.y == 2);
assert(d.Area == 1);

algo->FillDefect(0, &d);
assert(d.x == 1);
assert(d.y == 2);
assert(d.Area == 3);
```

# Enumerable

```
...
public interface IAlgorithm
{
    void SetImage(int index, IImage image);

    [ComVisible(false)]
    IEnumerable<IImage> Images { get; }

    System.Collections.IEnumerator GetImageEnumerator();
}

...
public class ThresholdAlgorithm : IAlgorithm
{
    IDictionary<int, IImage> IndexedImages =
       new Dictionary<int, IImage>();

    public void SetImage(int index, IImage image)
    {
        if( IndexedImages.ContainsKey(index)) {
            IndexedImages[index] = image;
        } else {
            IndexedImages.Add(index, image);
        }
    }

    public IEnumerable<IImage> Images {
        get { return IndexedImages.Values; }
    }

    public IEnumerator GetImageEnumerator() {
        return IndexedImages.Values.GetEnumerator();
    }
```

```
IAlgorithmPtr algo(__uuidof(ThresholdAlgorithm));

IImageFactoryPtr fab(__uuidof(ImageFactory));
algo->SetImage(0, fab->CreateImage_3(320, 240, 8, 42));
algo->SetImage(1, fab->CreateImage_3(64, 128, 8, 255));

IEnumVARIANTPtr I = algo->GetImageEnumerator();
for (;;)
{
    _variant_t v;
    ULONG count = 0;
    if (FAILED(I->Next(1, &v, &count)) || count != 1)
        break;

    IImagePtr image(v.punkVal);
    uint8_t pixelValue = image->GetPixel(0, 0);
    if (image->Width == 320) {
        assert(pixelValue == 42);
    } else {
        assert(pixelValue == 255);
    }
}
```

# Callback

```
[ComVisible(true)]
[InterfaceType(ComInterfaceType.InterfaceIsIDispatch)]
[Guid("A7673505-60FA-4DDA-91D8-A2CB609C19A1")]
public interface IProgress
{
    [DispId(1)] void Info(string msg);

    [DispId(2)] void Warn(string msg);
}

[ComVisible(true)]
[ClassInterface(ClassInterfaceType.AutoDual)]
[ComSourceInterfaces(typeof(IProgress))]
[Guid("3F8748C9-659D-4012-9467-CD49BCD39516")]
public class ProgressBase
{
    [ComVisible(false)]
    public delegate void LogFn(string msg);

    public event LogFn Info;
    public event LogFn Warn;

    protected void FireInfo(string msg)
    {
        if (Info == null) return;
        Info(msg);
    }

    protected void FireWarn(string msg)
    {
        if (Warn == null) return;
        Warn(msg);
    }
}
```

```
#import
  "C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscorlib.tlb"
  named_guids no_namespace rename("_Module", "_NETModule")

#import "...\MyCompany.Lib.tlb" named_guids

_ATL_FUNC_INFO LogFnInfo =
  { CC_STDCALL, VT_EMPTY, 1, { VT_BSTR } };

class LogSink : public IDispEventImpl<
  1, LogSink, &MyCompany_Lib::DIID_IProgress>
{
public:
  BEGIN_SINK_MAP(LogSink)
    SINK_ENTRY_INFO(1, MyCompany_Lib::DIID_IProgress,
      1, OnInfo, &LogFnInfo)
    SINK_ENTRY_INFO(1, MyCompany_Lib::DIID_IProgress,
      2, OnWarn, &LogFnInfo)
  END_SINK_MAP()

  LogSink(MyCompany_Lib::_ProgressBase* pbase)
  {
    m_ProgressBase = pbase;
    m_ProgressBase->AddRef();
    DispEventAdvise(m_ProgressBase);
  }

  ~LogSink()
  {
    m_ProgressBase->Release();
    DispEventUnadvise(m_ProgressBase);
   }
```

# Callback - Continued

```
...
public interface IAlgorithm
{
    void Run(string arg);
}

...
public class ThresholdAlgorithm :
  ProgressBase, IAlgorithm
{
  public void Run(string arg)
  {
    if( arg == "bar")
    {
      FireWarn(
        "Warning:This algo does work well with bar");
    }

    FireInfo( string.Format(
      "Starting Algorithm with {0}", arg ));

...

    FireInfo( string.Format(
      "Stopped Algorithm with {0}", arg ));
  }
}
```

```
  void __stdcall OnInfo(BSTR msg)
  {
    _bstr_t bmsg(msg,false);
    cout << "Info : " << bmsg << endl;
   }

  void __stdcall OnWarn(BSTR msg)
  {
    _bstr_t bmsg(msg, false);
    cout << "Warn : " << bmsg << endl;
  }

private:
  MyCompany_Lib::_ProgressBase* m_ProgressBase;
};

using namespace MyCompany_Lib;
IAlgorithmPtr algo(__uuidof(ThresholdAlgorithm));
LogSink logger((_ProgressBase*)(IAlgorithm *)algo);

algo->Run(_bstr_t("foo"));
algo->Run(_bstr_t("bar"));


Info : Starting Algorithm with foo
Info : Stopped Algorithm with foo
Warn : Warning : This algo does work well with bar
Info : Starting Algorithm with bar
Info : Stopped Algorithm with bar
```