COM（component object modul 组件对象模型）

- 是一个由微软开发的软件架构，让不同语言开发的软件可以互相调用。

COM组件

- COM是将一些功能实现封装成一个组件，然后通过COM组件的接口来调用组件的功能。

COM组件的接口

COM规范规定，任何组件或者接口都必须从IUnknown接口派生。IUnknown定义了3个重要函数，分别是：

- **QueryInterface**： 查询接口
- **AddRef**： 增加引用计数
- **Release**： 减少引用计数

除了IUnknwon接口外，还有另外一个重要接口**IClassFactory。**IClassFactory接口定义了一个函数：**CreateInstance**，用于创建组件的实例。 COM组件主要有3种类型：

- 进程内组件（CLSCTX_INPROC_SERVER）
- 本地进程组件（CLSCTX_LOCAL_SERVER）
- 远程组件（CLSCTX_REMOTE_SERVER）

COM组件开发

实现一个COM组件需要以下步骤：

1. COM组件接口
2. COM组件实现类

1. 编写COM组件接口

```
// Interface.h
#pragma once
/*
COM 组件接口
接口注意点：
1. 所有函数采用标准调用约定
2. 所有结果通过传出参数传出
3. 所有错误信息通过返回值传出
4. 所有接口都必须能进行接口查询
5. 所有接口都必须提供引用计数的功能
*/
```

```
#include <Windows.h>

// interface id，COM组件接口唯一标识
static const GUID IID_IUnkwon =
{ 0x15b00fbc, 0x5e7, 0x41d9, { 0xa5, 0xb2, 0xee, 0x36, 0xed, 0xbf, 0x8e, 0x95 } };

// 工厂组件接口
static const GUID IID_IFactory =
{ 0x1bdc4e8c, 0x6bed, 0x4153, { 0xa5, 0x3, 0x56, 0xc2, 0x9e, 0x6d, 0xd1, 0x4 } };

// 组件接口
static const GUID IID_ISayHello =
{ 0x9c995d0a, 0x1f3a, 0x43e2, { 0xbb, 0x8a, 0x23, 0x1a, 0x79, 0x73, 0xcf, 0x38 }
};

// 当前组件库唯一标识
static const GUID CLSID_MYCOM =
{ 0x74e5413d, 0xe51b, 0x4cd8, { 0xa9, 0x32, 0x96, 0xdc, 0xd, 0x3, 0x14, 0x27 } };

/*
IUnknown_my 接口
COM 组件标准接口，外部使用者可以通过该接口查询要使用的类
IUnknown_my 接口为基类，COM组件所有实现必须继承自它
*/
class IUnknown_my
{
public:
    virtual HRESULT __stdcall QueryInterface(const GUID iid, void** pInterface) =
0;
    virtual void __stdcall AddRef() = 0;
    virtual void __stdcall ReleaseRef() = 0;
};

/*
IFactory 工厂类
对于外部使用者来说，COM组件的类是不可知的，那么实例化这个类就可以通过与之对应的类工厂进行
实例化，
IFactory 为基类，COM组件的所有实现的类工厂必须继承自它
*/

class IFactory : public IUnknown_my
{
public:
    virtual HRESULT __stdcall CreateInstance(const GUID iid, void** pInterface) =
0;
};

/*
ISayHello COM组件接口
COM组件提供的接口
继承自 IUnknown_my 接口
外部使用者真正使用的功能接口
*/
class ISayHello : public IUnknown_my
```

```
{
public:
    virtual HRESULT __stdcall SayHello() = 0;
};

/*
IComFactory COM组件工厂接口
COM组件提供的工厂接口
继承自 IFactory
当有多个 COM 组件时，可以通过该类工厂实例化不同的 COM 对象
*/
class IComFactory : public IFactory
{
public:
    virtual HRESULT __stdcall CreateCom(const GUID iid, void** pInterface) = 0;
    virtual HRESULT __stdcall CreateCom(const GUID iid, int arg1, int arg2, void**
pInterface) = 0;
};

/*
导出函数，外部使用者通过该函数获取对应的 COM 组件的工厂类
*/
extern "C" __declspec(dllexport) HRESULT GetClassObject(const GUID clsid, void**
pObject);
using PFN_GetClassObject = HRESULT(*)(const GUID clsid, void** pObject);
```

2. 编写COM组件实现类

```
// 实现COM类
#include <iostream>

#include "interface.h"

class CMycom : public ISayHello
{
public:
    // 通过 ISayHello 继承
    HRESULT __stdcall QueryInterface(const GUID iid, void** pInterface) override
    {
        if (iid == IID_IUnknown)
        {
            *pInterface = this;
        }
        else if (iid == IID_ISayHello)
        {
            *pInterface = (ISayHello*)this;
        }
        else
        {
            return E_NOTIMPL;
        }
```

```
        AddRef();
        return S_OK;
    }
    void __stdcall AddRef() override
    {
        ++m_nRefCnt;
    }
    void __stdcall ReleaseRef() override
    {
        if (--m_nRefCnt == 0)
        {
            delete this;
        }
    }
    HRESULT __stdcall SayHello() override
    {
        std::cout << "hello" << std::endl;
        return S_OK;
    }
private:
    int m_nRefCnt = 1;
};

class CMycomFactory : public IFactory
{
public:
    // 通过 IFactory 继承
    HRESULT __stdcall QueryInterface(const GUID iid, void** pInterface) override
    {
        if (iid == IID_IUnknown || iid == IID_IFactory)
        {
            *pInterface = this;
        }
        else
        {
            return E_NOTIMPL;
        }
        AddRef();
        return S_OK;
    }

    void __stdcall AddRef() override
    {
        ++m_nRefCnt;
    }
    void __stdcall ReleaseRef() override
    {
        if (--m_nRefCnt == 0)
        {
            delete this;
        }
    }
```

```
    HRESULT __stdcall CreateInstance(const GUID iid, void** pInterface) override
    {
        CMycom* pCom = new CMycom;
        if (pCom == nullptr)
        {
            return E_FAIL;
        }
        HRESULT hErr = pCom->QueryInterface(iid, pInterface);
        pCom->ReleaseRef();
        return hErr;
    }

private:
    int m_nRefCnt = 1;
};

HRESULT GetClassObject(const GUID clsid, void** pObject)
{
    // 判断是否时本组件库支持的组件
    if (clsid != CLSID_MYCOM)
    {
        return E_FAIL;
    }

    // 返回对应组件的工厂类
    *pObject = new CMycomFactory;
    return S_OK;
}
```

COM组件使用

```
#include "../Dll1/Interface.h"

int main()
{
    // 加载 COM 组件所在文件
    HMODULE hCom = LoadLibrary("Dll1.dll");
    PFN_GetClassObject pfnGetClassObject =
(PFN_GetClassObject)GetProcAddress(hCom, "GetClassObject");

    // 找到对应工厂类
    IFactory* pFactory = nullptr;
    HRESULT hErr = pfnGetClassObject(CLSID_MYCOM, (void**)&pFactory);
    if (FAILED(hErr))
    {
        return 0;
    }

    // 通过该工厂类创建 COM 组件
    IUnknown_my* pUnk = nullptr;
    pFactory->CreateInstance(IID_IUnknown, (void**) &pUnk);
```

```
    // 通过 COM 组件找到接口
    ISayHello* pSayHello = nullptr;
    pUnk->QueryInterface(IID_ISayHello, (void**)&pSayHello);
    pSayHello->SayHello();
    pSayHello->ReleaseRef();

    pUnk->ReleaseRef();

    // 或者通过该工厂类创建 COM 组件时找到该接口
    ISayHello* pSayHello1 = nullptr;
    pFactory->CreateInstance(IID_ISayHello, (void**)&pSayHello1);
    pSayHello1->SayHello();
    pSayHello1->ReleaseRef();

    pFactory->ReleaseRef();
    return 0;
}
```