COM标准导出函数

- **DllGetClassObject** - 返回类工厂

```
STDAPI DllGetClassObject(const CLSID & rclsid, const IID & riid, void ** ppv)
{
    // 检查传入的 CLSID 是否是我们支持的
    if (rclsid!= CLSID_SayHello)
    {
        return CLASS_E_CLASSNOTAVAILABLE;
    }

    // 创建类工厂对象
    ISayHelloClassFactory* pClassFactory = new ISayHelloClassFactory();
    if (pClassFactory == nullptr)
    {
        return E_OUTOFMEMORY;
    }

    // 获取类工厂对象的接口指针
    HRESULT hr = pClassFactory->QueryInterface(riid, ppv);
    if (FAILED(hr))
    {
        pClassFactory->Release();
        return hr;
    }

    return S_OK;
}
```

- **DllRegisterServer** - 安装COM、将信息写入注册表

```
// 要写入到注册表的三个地方
HKEY_CLASSES_ROOT\CLSID\{CLSID_SayHello}\InprocServer32 (默认) 填入DLL路径
HKEY_CLASSES_ROOT\CLSID\{IID_ISayHello}\ProgID (默认) 填入 SayHello
HKEY_CLASSES_ROOT\SayHello\CLSID (默认) 填入 CLSID_SayHello
```

- **DllUnregisterServer** - 卸载COM、从注册表中删除信息
- **DllCanUnloadNow** - 判断是否释放COM在内存中的DLL

```
// 使用全局引用计数、监视是否可以释放DLL
static LONG g_cRef = 0;
```

```
STDAPI DllCanUnloadNow()
{
    return (g_cRef == 0)? S_OK : S_FALSE;
}
```

上述步骤完成后，导出函数，使用**regsvr32**命令行注册COM组件。

COM组件使用

COM库初始化。

```
CoInitialize(NULL);
```

通过clsid拿到对应的工厂类，然后通过工厂类实例化COM组件。

```
STDAPI CoGetClassObject(
  REFCLSID rclsid,  //CLSID associated with the class object
  DWORD dwClsContext,
                    //Context for running executable code
  COSERVERINFO * pServerInfo,
                    //Pointer to machine on which the object is to
                    // be instantiated
  REFIID riid,      //Reference to the identifier of the interface
  LPVOID * ppv      //Address of output variable that receives the
                    // interface pointer requested in riid
);
- 第一个参数是CLSID，通过CLSID可以找到对应的工厂类。
- 第二个参数是上下文，有3种：
-   - CLSCTX_INPROC_SERVER：进程内组件
-   - CLSCTX_LOCAL_SERVER：本地进程组件
-   - CLSCTX_REMOTE_SERVER：远程组件
- 第三个参数是服务器信息，一般为NULL。
- 第四个参数是接口ID，通过接口ID可以找到对应的接口。
- 第五个参数是传出接口指针。
```

通过clsid找到对应接口。

```
STDAPI CoCreateInstance(
  REFCLSID rclsid,  //CLSID associated with the class object
  LPUNKNOWN pUnkOuter, //Pointer to the controlling unknown
  DWORD dwClsContext, //Context for running executable code
  REFIID riid,        //Reference to the identifier of the interface
  LPVOID * ppv        //Address of output variable that receives the interface
pointer requested in riid
);
- 第一个参数是CLSID，通过CLSID可以找到对应的工厂类。
```

- 第二个参数是控制未知，一般为NULL。
- 第三个参数是上下文，有3种：
-   - CLSCTX_INPROC_SERVER：进程内组件
-   - CLSCTX_LOCAL_SERVER：本地进程组件
-   - CLSCTX_REMOTE_SERVER：远程组件
- 第四个参数是接口ID，通过接口ID可以找到对应的接口。
- 第五个参数是传出接口指针。

卸载DLL。

```
CoFreeAllLibrarise();
```

COM库释放。

```
CoUninitialize();
```

COM类型库

- 类型库是COM组件的描述文件，用于描述COM组件的接口、类、方法、属性等信息。
- 定义类型库后，可以被其他语言使用，如 C/C++ 类型库的生成。

```
// mycom.idl
// 定义接口
[
  object,
  uuid("15b00fbc-5e7-41d9-a5b2-ee36edbf8e95"), // 该项必须，其他可以没有
  local,
  pointer_default(unique)
]
interface ISayHello : IUnknow
{
  void Hello(); // 如果有参数，需要使用 `[in]`和 `[out, retval]` 关键字修饰
};
// 生成二进制类型库
[uuid("新的uuid")]
library mycomtlb
{
    [uuid("74e5413d-e51b-4cd8-a932-96dcd031427")]
    coclass Mycom
    {
        interface ISayHello;
    }
}
```

类型库的使用

```
// 将类型库放入DLL的资源中
3 TEXTINCLUDE
BEGIN
    "#include ""mycom.tlb""\r\n"
    "\0"
END
```

现在可以将得到的DLL文件给其他语言使用了。

```
// C++使用类型库
#import "mycom.tlb" no_namespace

int main()
{
    // 创建实例
    Mycom::ISayHelloPtr pHello = new Mycom::Mycom;
    // 调用方法
    pHello->Hello();
    return 0;
}
```

```
// C使用类型库
#include "mycom.h"

int main()
{
    // 创建实例
    ISayHello* pHello = NULL;
    CoCreateInstance(CLSID_Mycom, NULL, CLSCTX_INPROC_SERVER, IID_ISayHello,
(void**)&pHello);
    // 调用方法
    pHello->Hello();
    pHello->Release();
    return 0;
}
```

类型库注册。

```
regsvr32 mycom.tlb
```

类型库卸载。

```
regsvr32 /u mycom.tlb
```

类型库查看。

```
tlbexp mycom.tlb
```

```
// 使用 COM API 注册类型库
STDAPI DllRegisterServer()
{
    // 注册COM组件
    HRESULT hr = RegisterServer(CLSID_SayHello, L"SayHello", L"SayHello",
L"SayHello", L"SayHello", IID_ISayHello);
    if (FAILED(hr))
    {
        return hr;
    }

    // 注册类型库
    ITypeLib* pTypeLib;
    hr = LoadTypeLib(L"mycom.tlb", &pTypeLib);
    if (SUCCEEDED(hr))
    {
        hr = RegisterTypeLib(pTypeLib, L"mycom.tlb", NULL);
        if (SUCCEEDED(hr))
        {
            printf("Type library registered successfully.\n");
        }
        else
        {
            printf("Failed to register type library. Error: %x\n", hr);
        }
        pTypeLib->Release();
    }
    else
    {
        printf("Failed to load type library. Error: %x\n", hr);
    }

    return S_OK;
}
```

```
// 使用 COM API 卸载类型库
STDAPI DllUnregisterServer()
{
    // 卸载COM组件
    HRESULT hr = UnregisterServer(CLSID_SayHello);
```

```
    if (FAILED(hr))
    {
        return hr;
    }

    // 卸载类型库
    ITypeLib* pTypeLib;
    hr = LoadTypeLib(L"mycom.tlb", &pTypeLib);
    if (SUCCEEDED(hr))
    {
        hr = UnregisterTypeLib(pTypeLib, L"mycom.tlb", NULL);
        if (SUCCEEDED(hr))
        {
            printf("Type library unregistered successfully.\n");
        }
        else
        {
            printf("Failed to unregister type library. Error: %x\n", hr);
        }
        pTypeLib->Release();
    }
    else
    {
        printf("Failed to load type library. Error: %x\n", hr);
    }

    return S_OK;
}
```