
DEFINITION DER GRUNDLEGENDEN ELEMENTE:

Interface:

Das Interface ist eine Gruppe von Funktionen, die Methoden genannt werden. Interface Namen starten immer mit I, z.B. IFeature. In C++ wird ein Interface als eine abstrakte Basisklasse implementiert. Interfaces können vererbt werden.

Coclass (component object class):

Eine coclass ist in einer DLL oder einer EXE enthalten und beinhaltet den Code von einem oder mehreren Interfaces. Ein COM Objekt ist eine Instanz von einer coclass im Datenspeicher.

COM Server:

Ein COM Server ist eine binary (DLL oder EXE) die eine oder mehrere coclasses enthält.

Registration:

Erstellung von Register Einträgen, die Windows übermitteln wo ein COM Server lokalisiert werden kann.

GUID (global unique identifier):

Der GUID ist eine 128 – bit große Nummer. Über die GUID's können, COM Sprachen unabhängig, Interfaces und coclasses identifiziert werden. In der bisherigen COM Geschichte konnten Namens Kollisionen vermieden werden. GUID's werden manchmal auch als UUID's bezeichnet.

ID:

Eine Class ID, CLSID, ist eine GUID die eine coclasse beschreibt. Eine Interface ID, IID, ist eine GUID die ein Interface beschreibt.

Warum GUID's:

1. Jede Programmiersprache kann diesen nur aus Buchstaben bestehenden Identifier interpretieren
2. Jede GUID ist, wenn sie richtig erzeugt wurde, einzigartig.

HRESULT:

HRESULT ist ein integrierter Typ, der von COM für die Rückgabe von error und success Codes zuständig ist.

COM Library:

Die COM Library ist Teil des Betriebssystems in dem gearbeitet wird.

ARBEITEN MIT COM OBJEKTEN:

COM muss Programmiersprachen unabhängig sein. Die COM Library stellt ihre eigenen Objekt – Management Routinen zur Verfügung.

Ein Vergleich zwischen COM und C++:

Erzeugen eines neuen Objektes:

- In C++: mit operator new oder create auf einem Stack
- In COM: aufrufen einer API in der COM Library

Löschen eines Objektes:

- In C++: mit operator delete
- In COM: COM Objekte löschen sich selber, wenn der Reference Count 0 ist

Wenn man ein COM Objekt erzeugt, sagt man der COM Library welches Interface genutzt wird. Wenn das Objekt richtig erzeugt wurde, dann gibt die COM Library einen Pointer auf das Interface zurück. Über diesen Pointer können dann die Methoden des Interfaces angesprochen werden.

Erzeugen eines COM Objektes:

Um ein COM Objekt zu erzeugen und ein Interface von diesem Objekt zu bekommen, ruft man die COM Library API CoCreateInstance() auf:

```
HRESULT CoCreateInstance (
    REFCLSID rclsid,
    LPUNKNOWN pUnkOuter,
    DWORD dwClsContext,
    REFIID riid,
    LPVOID* ppv );
```

rclsid:

Die CLSID der coclass.

pUnkOuter:

Wird nur genutzt wenn COM Objekte vereinigt werden. Hierbei werden existierende coclasses genommen und um neu Methoden erweitert. Meistens kann NULL gesetzt werden.

dwClsContext:

Zeigt an welche Art von Server wir nutzen wollen. Für eine in-process DLL muss der Parameter auf CLSCTX_INPROC_SERVER.

riid:

ID des Interface welches zurückgegeben werden soll.

ppv:

Adresse des Interface Pointers. Die COM Library gibt das Interface über diesen Parameter zurück-

Ruft man nun CoCreateInstance() auf, wird zuerst nach der CLSID im Register gesucht, dann wird die Location des Servers gelesen und der Server in den Speicher geladen. Abschließend wird eine Instanz der coclasses erzeugt.

```
HRESULT hr;
IShellLink* pISL;

hr = CoCreateInstance ( CLSID_ShellLink,           // CLSID of coclass
                       NULL,                     // not used - aggregation
                       CLSCTX_INPROC_SERVER,     // type of server
                       IID_IShellLink,          // IID of interface
                       (void**) &pISL );        // Pointer to our interface pointer

if ( SUCCEEDED ( hr ) )
{
    // Call methods using pISL here.
}
else
{
    // Couldn't create the COM object. hr holds the error code.
}
```

Löschen eines COM Objektes:

Das IUnknown Interface das jedes COM Objekt implementiert enthält die Methode Release(). Mit dieser Methode kann dem COM Objekt gesagt werden, dass es nicht länger gebraucht wird.

Wenn eine Applikation aus vielen COM Objekten besteht, ist es wichtig das Release() aufgerufen wird, wenn man ein Interface nicht mehr benötigt. Wenn man dies nicht tut, dann werden die COM Objekte und die DLLs im Speicher bleiben und völlig nutzlos Speicherplatz in Anspruch nehmen. Wenn die Applikation eine lange Zeit laufen soll, sollte die CoFreeUnusedLibraries() API während des Prozesses aufgerufen werden.

```
// Create COM object as above. Then...

if ( SUCCEEDED ( hr ) )
{
    // Call methods using pISL here.

    // Tell the COM object that we're done with it.
    pISL->Release();
}
```

Das Basis Interface – IUnknown:

Jedes COM Interface stammt von IUnknown. Der Name sagt aus, dass, wenn man einen IUnknown Pointer zu einem COM Objekt hat, nicht weiß was das grundlegende Objekt ist, da jedes COM Objekt mit IUnknown implementiert wird.

IUnknown hat 3 Methoden:

1. AddRef() :

Sagt dem COM Objekt die Anzahl der Referenzen zu erhöhen. Diese Funktion würde z.B. genutzt, falls man ein Interface Pointer kopiert und sowohl der Originale Pointer, als auch die Kopie weiter verwendet werden soll.

2. Release() :

Sagt dem COM Objekt die Anzahl der Referenzen zu verringern.

3. QueryInterface() :

Fordert einen Interface Pointer von einem COM Objekt an. Diese Methode wird genutzt wenn eine coclass mehr als ein Interface implementiert.

Wenn man ein COM Objekt mit CoCreateInstance() erzeugt, erhält man einen Interface Pointer zurück. Wenn nun aber ein COM Objekt mehr als ein Interface implementiert, muss QueryInterface() genutzt werden, um den zusätzlichen Pointer zu erhalten, der genutzt wird.

```
HRESULT IUnknown::QueryInterface (
    REFIID iid,    //Interface ID
    void** ppv ); //Adresse des Interface Pointers
```

Quelle:

<http://www.codeproject.com/com/comintro.asp>