



Ambientes de Desenvolvimento Avançados

<http://www.dei.isep.ipp.pt/~jtavares/ADAV/ADAV.htm>

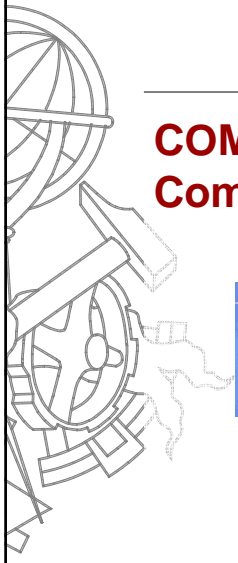
Aula 11

Engenharia Informática


2004/2005

José António Tavares
jrt@isep.ipp.pt

1




COM/OLE: Componentes Base



2004/2005

ADAV
Ambientes de Desenvolvimento Avançados


2



COM/OLE: Componentes Base

- GUID = Globally Unique Identifier
- Interfaces
- IUnknown
- Qual o 'aspecto' de um Interface?
- Implementação de Objectos
- Dispatch
- Polimorfismo no COM
- Mecanismos de Reutilização no COM
 - Delegação
 - Agregação
- Interface IClassFactory

2004/2005 ADAV 3
Ambientes de Desenvolvimento Avançados



GUID = Globally Unique Identifier

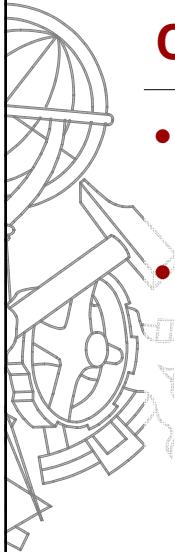
Ex:
3fad3020-16b7-11ce-80eb-00aa003d7352

Estrutura de um GUID:

```
typedef struct GUID
{
    DWORD Data1;
    WORD Data2;
    WORD Data3;
    BYTE Data4[8];
} GUID;

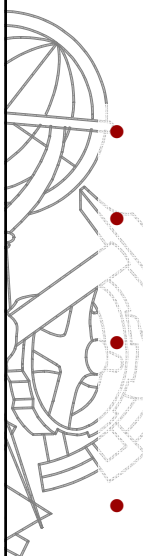
typedef GUID CLSID; /* Classe ID */
typedef GUID IID; /* Interface ID */
```

2004/2005 ADAV 4
Ambientes de Desenvolvimento Avançados



CLSID

- Each COM class is identified by a **CLSID**, a unique 128-bit **GUID** (Global Unique Identifier), which the server must register
- COM uses this **CLSID**, at the request of a client, to access the DLL or EXE containing the code that implements the class, which then creates an instance of the object (or finds an existing instance)



Interfaces

- Um objecto tem uma identidade e pode ser utilizado através dos seus interfaces.
- Os interfaces de um objecto não são mais do que conjuntos de funções que estão semanticamente relacionadas.
- Cada interface de um objecto significa que este suporta um conjunto de funcionalidades independentes do tempo e do espaço (através da especificação do seu **IID**).
- Apesar de um objecto pertencer a uma classe (**CLSID**) são os seus interfaces que o tornam útil...

Globally Unique Interface ID

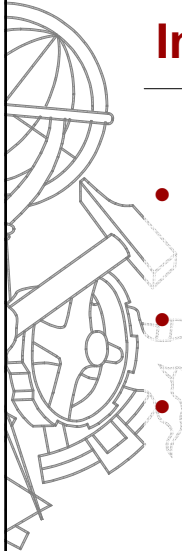
- Each interface is referred to at run time with a globally unique interface identifier (IID)
- An IID is a 128-bit number
- This IID allows a client to ask an object precisely whether it supports the interface
- Eliminates the possibility of duplication that could occur with any other naming scheme
- Permits multiple versions of the same interface with the same name

2004/2005 ADAV 7
Ambientes de Desenvolvimento Avançados

Interfaces

- A representação binária de um interface...

2004/2005 ADAV 8
Ambientes de Desenvolvimento Avançados

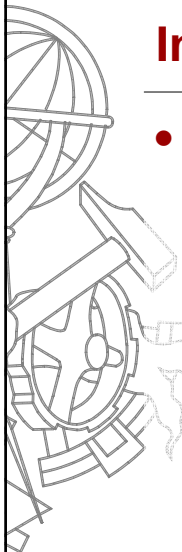


Interfaces

I Unknown

- O Interface **I Unknown** permite inquirir o objecto sobre outros interfaces que este suporte assim como controlar a utilização do objecto.
- Todos os Interfaces COM derivam directa ou indirectamente do Interface **I Unknown**.
- Assim, em qualquer Interface os primeiros métodos são sempre os métodos de **I Unknown**.

2004/2005 ADAV 9
Ambientes de Desenvolvimento Avançados



Interfaces

- Os métodos do interface **I Unknown** são:
 - **QueryInterface**
 - Permite obter outros interfaces suportados pelo objecto
 - **AddRef**
 - Incrementa o contador de referências
 - **Release**
 - Decrementa o contador de referências

2004/2005 ADAV 10
Ambientes de Desenvolvimento Avançados

Interface

Propriedades do método QueryInterface

Supondo os interfaces IPrimeiro, ISegundo e ITerceiro

Reflexiva	pPrimeiro->QueryInterface(IID_IPrimeiro) deve ter sempre sucesso.
Simétrica	Se pSegundo foi obtido através de pPrimeiro->QueryInterface(IID_ISegundo), então pSegundo->QueryInterface(IID_IPrimeiro) tem que ter sucesso.
Transitiva	Se pSegundo foi obtido através de pPrimeiro->QueryInterface(IID_ISegundo) e pTerceiro foi obtido através de pSegundo->QueryInterface(IID_ITerceiro), então pTerceiro->QueryInterface(IID_IPrimeiro) deve ter sempre sucesso.

Qual o 'aspecto' de um Interface?

- Por exemplo o IUnknown em C++,

```
#define interface struct  
  
interface IUnknown  
{  
public:  
    virtual HRESULT __stdcall QueryInterface(  
        /*[in]*/ REFIID riid,  
        /*[out]*/ void __RPC_FAR * __RPC_FAR *ppvObject) = 0;  
    virtual ULONG __stdcall AddRef(void) = 0;  
    virtual ULONG __stdcall Release(void) = 0;  
};
```

Qual o 'aspecto' de um Interface?

- 'Retirando' as especificidades de especificação do interface em C++ obtemos...

```
interface IUnknown
{
    HRESULT QueryInterface(IIID& iid, void **ppv);
    ULONG AddRef(void);
    ULONG Release(void);
};
```

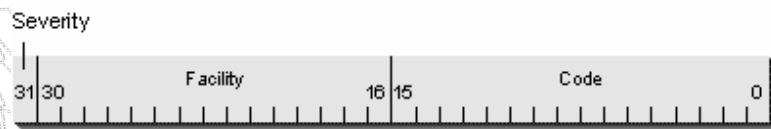
2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

13

Qual o 'aspecto' de um Interface?

- O Retorno de quase todos os métodos de interfaces COM é um HRESULT (*handle to result*)...



Severity: (1 bit) Severity field

- 0 *Success*. The function was successful.
- 1 *Error*. The function failed due to an error condition.

Facility: (15 bits) Indicates which group of status codes this belongs to. Microsoft reserves the exclusive right to define facility codes. FACILITY_ITF is used for all errors arising from custom interfaces.

Code: (16 bits) Describes what actually took place, error or otherwise.

2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

14

Unknown and QueryInterface

```
HRESULT QueryInterface(GUID iid, void** pptr) // referencing earlier slide
{
    if(iid == IID_ICat) // Input GUID compared to Cat interface's GUID
        *pptr = static_cast<ICat>(this);
    else if(iid == IID_IDog)
        *pptr = static_cast<IDog>(this);
    else if(iid == IID_IUnknown) //Common base interface. Cat or dog?-ambiguity!
        *pptr = static_cast<IDog>(this);
        // Implementer must choose! We choose dog
        // Other supported interfaces are checked against ...
    else { // Unsupported interface was asked for
        *pptr = 0;
        return E_NOINTERFACE;
    }
    // If *pptr is non-null AddRef() must be called
    reinterpret_cast<IUnknown*>(*pptr)->AddRef();
    return S_OK;
}
```

2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

15

Interfaces and IDL

```
[
    uuid(E4DF78F3-FE5B-11D3-9015-00C04FC12D77),
    helpstring("My Datatypes Lib")
]
library CalcTypes { // keyword causing tlb generation
    importlib("stdole32.tlb"); // required.
    [ // Intf defined "inside lib"
        uuid(E4DF78F4-FE5B-11D3-9015-00C04FC12D77),
        object
    ]
    interface IMyInsi deTypeI bDefl nedl f : IUnknown {
        HRESULT Method1();
    }
    [ uuid(E4DF78F5-FE5B-11D3-9015-00C04FC12D77) ]
    coclass Calc { // reference the desired interfaces here
        [default] interface ICalculator; // cause TLB Inclusion
        interface IMyInsi deTypeI bDefl nedl f;
    }
}
```

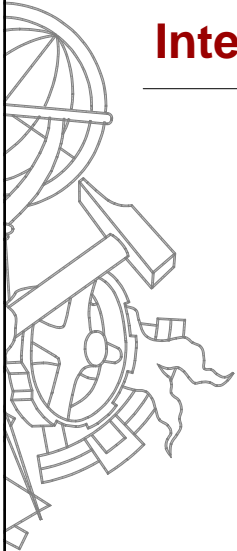
2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

16

Interfaces and IDL

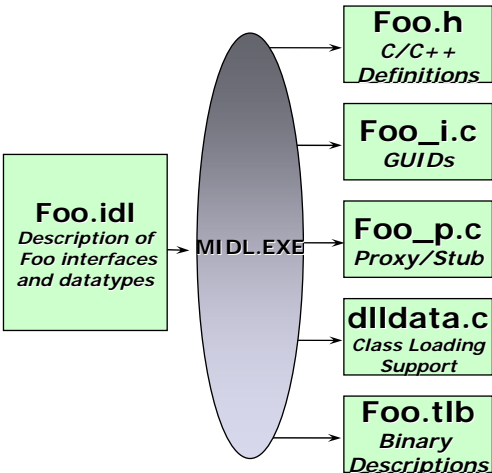
ShopCOM IDL



2004/2005 ADAV 17
 Ambientes de Desenvolvimento Avançados

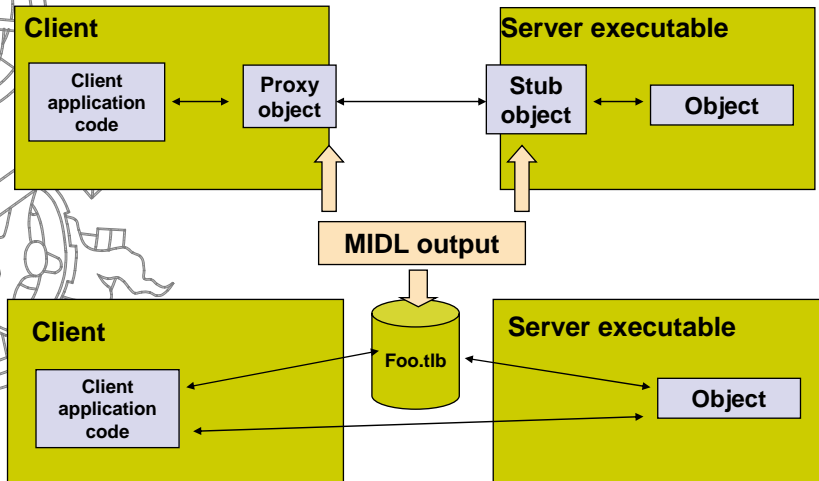
MIDL Compiler

- Microsoft IDL compiler
- Takes an IDL-file as input and generates mostly C/C++ code (not binary code) into:
 - The headers for the interfaces
 - A .tlb type library for the server (a binary)
 - The marshalling source code



2004/2005 ADAV 18
 Ambientes de Desenvolvimento Avançados

MIDL Compiler, cont



2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

19

Implementação de Objectos

Vamos supor um objecto que enumera rectangulos...

A definição do seu interface em C++ poderia ser...

```
typedef IEnumRECT *PENUMRECT;  
  
struct IEnumRECT  
{  
    STDMETHOD(QueryInterface)(REFIID, PPVOID)=0;  
    STDMETHOD_(ULONG, AddRef)(void)=0;  
    STDMETHOD_(ULONG, Release)(void)=0;  
    STDMETHOD(Next)(DWORD, LPRECT, LPDWORD)=0;  
    STDMETHOD(Skip)(DWORD)=0;  
    STDMETHOD(Reset)(void)=0;  
    STDMETHOD(Clone)(PENUMRECT *)=0;  
};
```

2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

20

A implementação de um objecto em C++ que suportasse a interface...

```
class CEnumRect : public IEnumRECT
{
private:
    DWORD m_cRef; //Reference count
    DWORD m_iCur; //Current enum position
    RECT m_rgrc[CRECTS]; //RECTS we enumerate

public:
    CEnumRect(void);
    ~CEnumRect(void);

    STDMETHODIMP QueryInterface(REFIID, PPVOID);
    STDMETHODIMP_(ULONG) AddRef(void);
    STDMETHODIMP_(ULONG) Release(void);

    //IEnumRECT members
    STDMETHODIMP Next(ULONG, LPRECT, ULONG *);
    STDMETHODIMP Skip(ULONG);
    STDMETHODIMP Reset(void);
    STDMETHODIMP Clone(PENUMRECT *);
};
```

2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

21

A implementação do construtor

```
CEnumRect::CEnumRect(void)
{
    UINT i;

    //Initialize array of rectangles.
    for (i=0; i < CRECTS; i++)
        SetRect(&m_rgrc[i], i, i*2, i*3, i*4);

    //Ref counts always start at 0.
    m_cRef=0;

    //Current pointer is first element.
    m_iCur=0;

    return;
}
```

2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

22

A implementação do método QueryInterface

```
STDMETHODIMP CEnumRect::QueryInterface(REFIID riid,
                                         PPVOID ppv)
{
    *ppv=NULL;
    if (IID_IUnknown==riid || IID_IEnumRECT==riid)
        *ppv=this;
    if (NULL==*ppv)
        return ResultFromScode(E_NOINTERFACE);
    ((LPUNKNOWN)*ppv)->AddRef();
    return NOERROR;
}
```

2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

23

A implementação dos métodos AddRef e Release

```
STDMETHODIMP_(ULONG) CEnumRect::AddRef(void)
{
    return ++m_cRef;
}
STDMETHODIMP_(ULONG) CEnumRect::Release(void)
{
    if (0!--m_cRef)
        return m_cRef;
    delete this;
    return 0;
}
```

2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

24

A utilização do objecto...

```
BOOL CreateRectEnumeratorCPP(PENUMRECT *ppEnum)
{
    PCEnumRect pER;
    HRESULT hr;

    if (NULL==ppEnum)
        return FALSE;

    //Create object.
    pER=new CEnumRect();

    if (NULL==pER)
        return FALSE;

    //Get interface, which calls AddRef.
    hr=pER->QueryInterface(IID_IEnumRECT, (void **)ppEnum);
    return SUCCEEDED(hr);
}
```

2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

25

IDispatch

- O interface **IDispatch** permite que os objectos, métodos e propriedades fiquem disponíveis para ferramentas de programação e aplicações que suportem *Automation* [ver **Automation**].
- O COM, ao permitir que os objectos implementem este interface, fornece a clientes *Automation* acesso aos objectos. Exemplos de clientes *Automation* são: Visual Basic, VB Script, etc.

2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

26



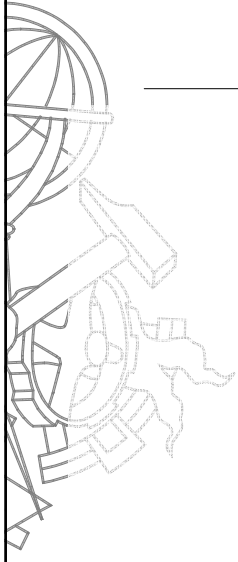
IDispatch

- Métodos do **IDispatch**
 - **GetTypeInfoCount**
Verifica se o objecto disponibiliza informação em *runtime* (0 – não; 1 – sim). Esta informação pode ser útil para *Browsers*, compiladores, etc.
 - **GetTypeInfo**
Obtém informação sobre o objecto. Esta informação pode ser diferente em função da linguagem.
 - **GetIDsOfNames**
Permite obter um identificador para um método ou propriedade para posteriormente ser utilizada pelo *Invoke*.
 - **Invoke**
Permite aceder a métodos e propriedades disponibilizadas pelo objecto.



Criar um Servidor COM utilizando o Wizard-ATL

- Em termos práticos usa-se o *Wizard-ATL* do Visual Studio para a criação de um servidor COM. Isto permite esconder todos os detalhes de baixo nível da implementação de um servidor COM.
- O exemplo do servidor a criar será muito simples e a sua finalidade é a de demonstrar a forma como pode ser utilizado para a construção do esqueleto do servidor, e alertar para as componentes essenciais dos servidores COM.
- O *Wizard ATL* permite que sejam seleccionadas todas as operações básicas para o servidor e fazer a geração da maior parte do código necessário.



Questões

?

2004/2005

ADAV
Ambientes de Desenvolvimento Avançados

29