Wstęp do programowania składnikowego

CZĘŚĆ 2:
interfejs *Iunknown*

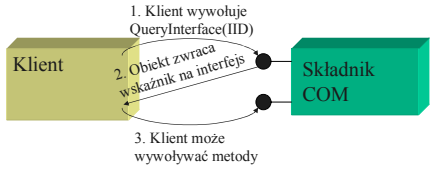
Jarosław Francik

Interfejs *IUnknown*

```
interface IUnknown
{
  virtual HRESULT __stdcall QueryInterface
          (const IID &iid, void **ppc) = 0;
   virtual ULONG __stdcall AddRef() = 0;
   virtual ULONG __stdcall Release() = 0;
};
```

Każdy interfejs COM dziedziczy po IUnknown

IUnknown::QueryInterface




```
HRESULT QueryInterface(const IID &iid, void **p);
```

identyfikator interfejsu – IID (GUID)

IUnknown::QueryInterface (użycie przez klienta)

```
// inicjalizacja składnika
IUnknown *pComp = CreateInstance();

// chcemy skorzystać z interfejsu IA
IA *pIA = NULL;
HRESULT hr = pComp->QueryInterface
                          (IID_IA, (void**)&pIA);
if (SUCCEEDED(hr))
{
  pIA->fa();
}
```

IUnknown::QueryInterface (implementacja)

```
HRESULT _stdcall CMyComp::QueryInterface(REFIID iid,
  void **ppv)
{
  if (iid == IID_IUnknown)
      *ppv = (IA*)this;
  else if (iid == IID_IA)
      *ppv = (IA*)this;
  else if (iid == IID_IB)
      *ppv = (IB*)this;
  else { ppv = NULL; return E_NOINTERFACE; }
  ((IUnknown*)(*ppv))->AddRef();
  return S_OK;
}
```

Funkcja *CreateInstance*

```
IUnknown *CreateInstance()
{
  IUnknown *p = (IA*)new CMyComp;
  p->AddRef();
  return p;
}
```

W bibliotece COM API dostępna jest funkcja *CoCreateInstance* o podobnym działaniu

Drugie podejście

- Co widzi klient?
  - interfejsy
  - identyfikatory interfejsów (IID)
  - nagłówek funkcji *CreateInstance* (w COM korzystamy raczej z funkcji bibliotecznej *CoCreateInstance)*
- Co implementuje składnik?
  - klasa implementująca interfejs *IUnknown* oraz własne interfejsy
  - funkcja CreateInstance

[listing #2]

Nie używamy dziedziczenia wirtualnego! Jest ono niezgodne z formatem binarnym COM

QueryInterface

QueryInterface możemy stosować do dowolnego interfejsu, nie tylko do IUnknown – gdyż wszystkie interfejsy dziedziczą po IUnknown

QueryInterface

```
IUnknown *p = CreateInstance();

IA *pIA = NULL;
hr = p->QueryInterface(IID_IA, (void**)&pIA);
if (!SUCCEEDED(hr)) return;

// Interfejs IB otrzymujemy z IA
IB *pIB = NULL;
hr = pIA->QueryInterface(IID_IB, (void**)&pIB);
if (!SUCCEEDED(hr)) return;

// Interfejs IUnknown otrzymujemy z IB
IUnknown * pUnknown = NULL;
hr = pIB->QueryInterface(IID_IUnknown,
   (void**)&pUnknown);
if (!SUCCEEDED(hr)) return;
```

QueryInterface

- Za każdym razem otrzymuję ten sam interfejs IUnknown
- Zawsze mogę otrzymać interfejs, który już kiedyś uzyskałem
- Zawsze mogę uzyskać interfejs, który już mam
- Jeśli mogę otrzymać interfejs gdziekolwiek, to mogę go otrzymać wszędzie

QueryInterface

- definiuje obiekt
- pozwala udostępniać różne wersje interfejsów dla tego samego obiektu
- pomaga utrzymać kompatybilność wstecz przy wypuszczaniu nowej wersji produktu
- **ALE:** wymaga to dotrzymania założeń przez projektanta nowej wersji!!!

Zliczanie referencji: AddRef/Release

- Klasa składnika zawiera licznik referencji
- AddRef zwiększa licznik
- Release zmniejsza licznik i w razie potrzeby usuwa składnik z pamięci
- Utworzenie obiektu = AddRef
- Porzucenie obiektu = Release

Zliczanie referencji: AddRef/Release

```
ULONG _stdcall CMyComp::AddRef()
{
   return InterlockedIncrement(&m_nRef);
}

ULONG _stdcall CMyComp::Release()
{
   if (InterlockedDecrement(&m_nRef) == 0)
   {
       delete this;
       return 0;
   }
   return m_nRef;
}
```

AddRef/Release: trzy proste zasady

- Wywołaj AddRef zanim zwrócisz wynik
  - jeśli zwracasz interfejs jako wartość funkcji. Dotyczy też QueryInterface i CreateInstance!
- Wywołaj Release kiedy skończysz
  - gdy nie będziesz już dłużej wykorzystywał interfejsu
- Wywołaj AddRef gdy robisz przypisanie

AddRef/Release: Optymalizacja

- Nie musisz stosować AddRef/Release w przypadku użycia wskaźnika o zagnieżdżonym zasięgu (w obrębie zasięgu innego wskaźnika)
  - parametr wejściowy – nic (jest zagnieżdżony)
  - parametr wyjściowy – AddRef
  - parametr wejściowo-wyjściowy – być może AddRef + Release
  - zmienna lokalna – nic (jest zagnieżdżona)
  - zmienna globalna – AddRef + Release
  - w przypadku wątpliwości – AddRef + Release

Drugie podejście...

- Słaby punkt:
  - funkcja CreateInstance zbyt mocno wiąże klienta z serwerem
- Rozwiązanie:
  - skorzystamy z technologii COM...