TWORZENIE SKŁADNIKÓW ZEWNĄTRZPROCESOWYCH

Poniższy zwięzły opis przedstawia najważniejsze kroki niezbędne dla uzyskania klienta i serwera zewnątrz-procesowego COM (serwer w pliku EXE). Tekst ten powinien być używany łącznie z instrukcją dot. tworzenia składników wewnątrzprocesowych (*inproc*). Ilustracją jest też kod źródłowy *approach4*, a także inne materiały dostępne na stronie sun.iinf.polsl.gliwice.pl/~jfrancik/lectures/com.html.

1. STWORZENIE SPECYFIKACJI *IDL*

Ten krok przebiega tak samo, jak w przypadku składników *inproc*.

2. STWORZENIE APLIKACJI KLIENCKIEJ

Jedyną różnicą pomiędzy aplikacją korzystającą ze składnika zewnątrzprocesowego (w stosunku do aplikacji używającej składnika *inproc*) jest wartość trzeciego parametru funkcji *CoCreateinstance*, która musi zawierać ustawiony bit *CLSCTX_LOCAL_SERVER*.

3. STWORZENIE MODUŁU DLL PROXY

Moduł proxy jest niezbędny przy komunikacji klienta z serwerem zewnątrzprocesowym, nie wymaga jednak specjalnego kodowania. Wystarczy w tym celu:

- Stworzyć projekt DLL obejmujący następujące pliki źródłowe, automatycznie generowane podczas kompilacji specyfikacji *IDL*: *dlldata.c*, *abc_i.c*, *abc_p.c* (przy założeniu, że kompilowano plik *abc.idl*).
- Włączyć linkowanie nastepujących bibliotek (poza standardowymi): *rpcndr.lib*, *rpcns4.lib* i *rpcrt4.lib*.
- Dodać definicję symbolu *REGISTER_PROXY_DLL*
- Dołączyć plik *def* (por. przykład *approach4*)
- Skompilować i zlinkować projekt.
- Zarejestrować otrzymany plik *DLL* (za pomocą opcji *Tools|Register Control* w *Visual Studio* lub wywołując program regsvr32).

4. STWORZENIE MODUŁU EXE SERWERA

Większa część kodu źródłowego serwerów zewnątrz- i wewnątrzprocesowych jest taka sama. W szczególności punkty 1 – 4 w części instrukcji dotyczącej tworzenia serwerów *inproc* można zastosować do serwerów zewnątrzprocesowych. Punkty 5 – 7 tej instrukcji nie odnoszą się do serwerów w plikach EXE, w szczególności nie ma tu funkcji eksportowanych. Oto specyficzne dla tego typu serwerów czynności dodatkowe:

1. Stworzenie funkcji *WinMain* jak w przykładzie *Approach4*. Poniżej przedstawiono najważniejsze elementy:
   - Inicjalizacja systemu COM (*CoInitialize*).
   - Analiza linii wywołania: parametrów *RegServer* (rejestracja serwera), *UnregServer* (wyrejestrowanie serwera) i *Embedding* (praca tylko w trybie serwera, a nie aplikacji – zwykle powoduje wyłączenie wyświetlenia okna aplikacji).

     [Uwaga – wśród plików projektu Approach4 znajduja się pliki Registry.h i Registry.cpp z modyfikacjami niezbędnymi przy obsłudze składników zewnątrzprocesowych]
   - Utworzenie i rejestracja fabryk(i) klas (funkcja *CoRegisterClassObject*).
   - Wyświetlenie okna (zwykle z wyjątkiem trybu /*Embedding*) i petla komunikatów
   - Wyrejestrowanie fabryk(i) klas (*CoRevokeClassObject*) i deinicjalizacja *COM* (*CoUninitialize*)
2. Implementacja zarządzania usuwaniem aplikacji z pamięci. Wymaga to:
   - przy dekrementacji globalnych liczników (*g_cComponents* i *g_cServerLocks*) – sprawdzenia, czy aplikacja nie powinna być usunięta z pamięci:
   - `if (g_cComponents == 0 && g_cServerLocks == 0) PostQuitMessage(0);`
   - Blokady usuwania aplikacji, gdy używany jest interfejs użytkownika (okno aplikacji):
   - `InterlockedIncrement(&g_cServerLocks);` – wywołane tuż po otwarciu okna
   - Zniesienia powyższej blokady w chwili zamknięcia okna aplikacji:
   - `case WM_CLOSE: ::InterlockedDecrement(&g_cServerLocks);` – w funkcji okienkowej
   - Dodatkowej kontroli podczas obsługi komunikatu *WM_DESTROY*:
     `case WM_DESTROY: if (g_cComponents == 0 && g_cServerLocks == 0) PostQuitMessage(0);`
3. Zarejestrowanie serwera – poprzez wywołanie go z parametrem */RegServer*.

opracowanie: Jarosław Francik