COM Instrukcja do ćwiczenia laboratoryjnego z przedmiotu **Programowanie składnikowe w modelu COM** sem 8

ĆWICZENIE 2 OBJECT LINKING AND EMBEDDING (OLE) CZĘŚĆ 1: SERWERY

Proponowane ćwiczenie, podobnie jak poprzednio, wykorzystuje jako platformę programową środowisko *Microsoft Visual C++* oraz bibliotekę *MFC*.

TECHNOLOGIA OLE

Technologia OLE to historycznie najstarsza technologia związana z COM. Jej pierwsza wersja pojawiła się już w szesnastobitowym systemie Windows 3.1 i stanowiła oddzielny, samodzielny model, którego przeznaczenie sprowadzało się do możliwości umieszczania dokumentów wykonanych i kontrolowanych przez jedną aplikację w obrębie dokumentów innej aplikacji. Dopiero z upływem czasu (i wersji – OLE 2) podstawowe mechanizmy komunikacji międzyaplikacyjnej zostały uogólnione stając się zaczątkiem technologii COM. Technologia ta nosiła zresztą przez długi czas nazwę OLE (por. automatyzacja OLE, kontrolki OLE itp.). Obecnie OLE jest uważane za jedną z aplikacji COM, a jej funkcjonalność z grubsza jest podobna do tej pierwotnej, znanej jeszcze w erze Windows 3.1 – choć oczywiście obecnie jest dużo bardziej wydajna i stabilna niż wtedy. OLE rozpowszechniło się tak bardzo, że obecnie niewiele jest aplikacji nie zapewniających obsługi tego mechanizmu. Jest to zresztą jednym z czołowych warunków uzyskania certyfikatu zgodności oprogramowania z Windows.

Aplikacja **serwera OLE** to aplikacja, której dokumenty mogą być włączane w skład dokumentów innej aplikacji, **klienta OLE**. Dokument taki nazywamy **osadzonym** (*embedded*). Dokument innej aplikacji, do której go włączamy, to dokument **kontenera** (*container*). Nie zajmujemy się tu bliżej dokumentami dołączonymi (*linked*).

Podstawowe koncepcje współpracy aplikacji serwera OLE (*Object Linking and Embedding*) z dokumentem kontenera nie są skomplikowane, jednak stworzenie takiej aplikacji, ze względu na mnogość niezbędnych do zaimplementowania interfejsów, nie jest niestety proste. Naprzeciw potrzebom programistów wychodzą dostępne biblioteki, w których większa część implementacji jest już zrealizowana, a dla programisty pozostaje opracowanie jedynie tych części, które bezpośrednio wpływają na wygląd lub zachowanie się obiektu OLE osadzonego w dokumencie. W ramach ćwiczenia stworzony zostanie moduł aplikacji będącej serwerem OLE w oparciu o bibliotekę MFC.

ARCHITEKTURA OLE SERVER W MFC

Biblioteka MFC zawiera kilka stosunkowo prostych w obsłudze klas wykorzystywanych w serwerach. Są to:

- ***COleServerDoc*** to specjalna klasa dokumentów, wzbogacona względem macierzystej klasy *CDocument* o możliwość pracy w środowisku dokumentu - kontenera. W związku z tym jego odpowiedzialność jest podwójna:
  - obsługuje dokument w normalnym trybie pracy aplikacji,
  - obsługuje edycję aktywowanego dokumentu osadzonego.
- ***COleServerItem*** implementuje interfejs *OLE Server*, udostępniany aplikacji klienckiej. Interfejs ten odpowiada między innymi za:
  - serializację dokumentu osadzonego (funkcja Serialize),
  - określenie rozmiarów dokumentu osadzonego (funkcja OnGetExtent),
  - wyświetlanie graficznej reprezentacji dokumentu osadzonego w obszarze kontenera – z wyjątkiem czasu, gdy jest on aktywowany (funkcja OnDraw),
  - uruchomienie dokumentu (pochodnej COleServerDoc) na czas jego aktywacji.

- ***COleIPFrameWnd*** implementuje interfejs ramki aktywacji w miejscu (*in place*), udostępniany aplikacji klienckiej jako obszar roboczy dla włączonego dokumentu. Obiekty tej klasy odpowiadają też za utworzenie pasków narzędziowych wyświetlanych przez aplikację kliencką w trakcie edycji dokumentu osadzonego (funkcja *OnCreateControlBars*).

W chwili aktywacji dokumentu (np. przez podwójne kliknięcie) tworzony jest obiekt klasy *COleIPFrameWnd* (a raczej klasy potomnej, która w aplikacji stworzonej przez AppWizarda będzie się zapewne nazywała *CInPlaceFrame*). Jako podklasa ramek dokumentów (*CFrameWnd*) stanowi ona środowisko pracy klas dokumentów i widoków – zatem podczas aktywacji dokumentu osadzonego całą jego edycję kontrolują klasy *CDocument i CView*, które obsługują edycje dokumentów w trakcie normalnej, pełnookienkowej pracy aplikacji. Dzięki temu każda poprawna aplikacja jest w zasadzie gotowa do obsłużenia trybu edycji dla dokumentów osadzonych.

W czasie, gdy dokument osadzony jest jedynie wyświetlany przez aplikację kliencką, a nie jest aktywowany, jego reprezentację – dla aplikacji klienta – stanowi interfejs implementowany przez obiekt z klasy *COleServerItem*. Obiekt ten jest tworzony automatycznie przez aplikację serwera na żądanie klienta. Proces utworzenia tego obiektu jest kontrolowany przez klasę dokumentu (*COleServerDoc::GetEmbeddedObject*). Klient może żądać wykonania określonych operacji (serializacja dokumentu osadzonego, określenie jego rozmiarów, wyświetlenie go) – poprzez wywołanie odpowiednich funkcji odpowiednich interfejsów. MFC wywołania te przekłada na wywołania odpowiednich funkcji wirtualnych klasy *COleServerItem*.

Zestawienie najważniejszych klas i ich metod

`COleServerDoc`	dokument aplikacji serwera OLE
`OnGetEmbeddedItem`	tworzy i udostępnia obiekt *COleServerItem* reprezentujący ten dokument
`NotifyChanged`	zawiadamia klienta OLE o zmianie zawartości i wyglądu dokumentu
`COleServerItem`	implementuje interfejs osadzonego dokumentu dla potrzeb klienta
`Serialize`	zapewnia serializację (odczyt i zapis) danych osadzonego dokumentu
`OnGetExtent`	określa obszar zajmowany przez osadzony dokument w kontenerze
`OnDraw`	definiuje sposób wyswietlania osadzonego dokumentu dokumentu w kontenerze. Serwer przechowuje obraz w metapliku i odświeża go jedynie wskutek wywołania *COleServerDoc::NotifyChanged*
`CopyToClipboard`	kopiuje informację o dokumencie do schowka Windows
`COleIPFrameWnd`	implementuje interfejs ramki aktywacji osadzonego dokumentu
`OnCreate`	między innymi tworzy uchwyty do zmiany rozmiaru ramki
`OnCreateControlBars`	tworzy paski narzędziowe serwera OLE w obrębie aplikacji kontenera

TWORZENIE SERWERÓW OLE W VC++ 6.0

W środowisku VC++ 6.0 utworzenie aplikacji będącej serwerem OLE jest stosunkowo proste, między innymi dlatego, że większość niezbędnych w tym celu elementów dostarcza aplikacja *AppWizarda* w postaci gotowej do użycia. Oto dokładne wytyczne:

1. Utworzyć aplikację za pomocą *AppWizarda*. W kroku 3 należy zaznaczyć opcję *Full-server*. Utworzona aplikacja będzie pełną aplikacją serwera OLE. Można też zaznaczyć opcję *Mini-server*, której efektem będzie serwer OLE, którego nie można uruchomić jako osobnej aplikacji.

   W efekcie powstanie typowy projekt MFC wzbogacony o klasę pochodną *COleServerItem* (*CXxxSrvrItem*) oraz o klasę *CInPlaceFrame*. Klasa dokumentu wywiedziona będzie z *COleServerDoc.*

2. Należy zaimplementować normalną funkcjonalność aplikacji. Z punktu widzenia sensowności osadzania dokumentów za niezbędne należy tu uznać zaimplementowanie serializacji dokumentów (*CXxxDoc::Serialize*).

   W tym momencie dysponujemy już aplikacją, której osadzone dokumenty można niemal poprawnie edytować w obrębie dokumentu kontenera.

3. Zaimplementować seriazlizację obiektów *CXxxSrvrItem* (funkcja składowa *Serialize*). Na ogół wystarcza odwołanie do funkcji *Serizlize* obiektu dokumentu – i taką właśnie, domyślną procedurę tworzy *AppWizard* – zatem krok ten można pominąć.

4. Zaimplementować ustalanie rozmiaru obiektów *CXxxSrvrItem* (funkcja składowa *OnGetExtent*). Można również i ten krok pominąć, gdyż prostą wersję implementacji tworzy *AppWizard*).
5. Zaimplementować wyświetlanie obiektów *CXxxSrvrItem* (funkcja składowa *OnDraw*). **Wskazówka:** ponieważ treść *CXxxSrvrItem::OnDraw* i *CXxxView::OnDraw* jest zwykle bardzo podobna, często stosuje się pomocniczą funkcję rysującą, umieszczoną w klasie *CXxxDoc*, która jest wykorzystywana zarówno przez *CXxxSrvrItem* jak i *CXxxView*. Uwaga: domyślna implementacja nie robi nic ciekawego i dlatego kroku tego nie należy pomijać.
6. We wszystkich miejscach, w których zawartość dokumentu ulega zmianie w takim stopniu, że powinno to rzutować na wygląd wyświetlanego obiektu *CXxxSrvrItem* należy wywołać funkcję *COleServerDoc::NotifyChanged* – celem odświeżenia widoku *CXxxSrvrItem.*

Bardzo istotna jest prawidłowa realizacja punktu 6. Technologia OLE Client została bowiem zoptymalizowana w taki sposób, by ograniczyć konieczność wywoływania aplikacji serwera jedynie do tych przypadków, gdy osadzony dokument w istotny sposób zmienia zawartość. Klient uruchamia interfejs serwera w momencie osadzenia nowego dokumentu w kontenerze. Powoduje to wywołanie funkcji *COleServerItem ::OnDraw*. Jej wynik jest następnie przechowywany w metapliku graficznym wraz z dokumentem klienta, dzięki czemu wizerunek osadzonego dokumentu można odtworzyć bez uruchamiania aplikacji serwera. Jednak powoduje to, że o ile serwer nie wywoła funkcji *COleServerDoc::NotifyChanged*, to zawartość wspomnianego metapliku nie zostanie odświeżona, i po zakończeniu edycji osadzonego dokumentu jego wygląd wróci do stanu sprzed edycji.

KOPIOWANIE OBIEKTÓW OLE

Klasa *COleServerItem* udostępnia niezwykle wygodną w użyciu funkcję składową *CopyToClipboard*, która pozwala na skopiowanie do schowka informacji dotyczącej dokumentu edytowanego przez aplikację w trybie samodzielnym. Wklejenie zawartości schowka do aplikacji klienta OLE jest wówczas równoważne utworzeniu osadzonego dokumentu w obrębie dokumentu kontenera. Przykładową funkcję realizującą operacją kopiowania dokumentu do schowka przedstawiono poniżej:

```
void CXxxDoc::OnEditCopy()
{
	CXxxSrvrItem* pItem = GetEmbeddedItem();
	pItem->CopyToClipboard(TRUE);
}
```

ZADANIA DO WYKONANIA

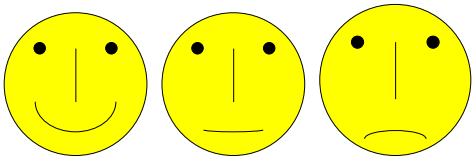
1. Zapoznać się z aplikacją serwera OLE *Smile*
2. Stworzyć wyspecyfikowaną przez prowadzącego aplikację serwera OLE, wypróbować ją w kontenerze, jakim może być na przykład dokument Microsoft Word. Aplikacja powinna udostępniać użytkownikowi aplikacji klienckiej własny zestaw potrzebnych pasków narzędziowych i menu.

Sprawozdanie z ćwiczenia powinno zawierać specyfikację stworzonej aplikacji serwera.

Załącznikiem do ćwiczenia jest aplikacja *Smile*. W stosunku do pierwotnego projektu wygenerowanego przez *AppWizarda* wprowadzone zmiany obejmują:

- podstawową funkcjonalność aplikacji (*OnDraw* i *OnLButtonDown* w klasie *CSmileView* oraz *Serialize* i zmienne *m_x* i *m_y* w klasie *CSmileDoc*),
- implementacja funkcji *CSmileSrvrItem::OnDraw*,
- uzupełnienie funkcji *CSmileView*::*OnLButtonDown* o wywołanie *CSmileDoc::NotifyChanged*.

A oto trzy osadzone dokumenty tej klasy:



opracowanie: Jarosław Francik