ATL: ACTIVE TEMPLATE LIBRARY

KRÓTKI WSTĘP DO PROGRAMOWANIA

JAROSŁAW FRANCIK

Biblioteka ATL, rozprowadzana wraz z pakietem Visual Studio/C++, jest przeznaczona do tworzenia modułów COM. W porównaniu z MFC oferuje nie tylko większe możliwości w tym zakresie, ale przede wszystkim gwarantuje znacznie mniejsze rozmiary skompilowanych modułów – co jest niezwykle istotne, szczególnie w przypadku tworzenia kontrolek ActiveX przeznaczonych do instalowania przez Internet.

Podobnie, jak w przypadku MFC, dogłębne poznanie biblioteki ATL wymaga długiego przygotowania. Jednak proste, ale funkcjonalne moduły można tworzyć znając jedynie najważniejsze elementy tej biblioteki.

Tworzone w oparciu o ATL klasy komponentów zapewniają obsługę standardowych interfejsów COM. Najprostszy z nich, zwany *prostym obiektem* (*simple object*) obsługuje interfejsy *IUnknown* i *IClassFactory*, tak więc programista nie musi (na ogół) dbać o ich implementację.

TWORZENIE MODUŁÓW SERWERA COM

Utworzenie modułu serwera COM wymaga wybrania opcji *File|New* i wskazania *ATL COM Wizard* jako typu projektu. W kolejnym oknie ustala się m.in. typ serwera (wewnątrzprocesowy – DLL lub zewnątrzproceoswy – EXE). Po zatwierdzeniu otrzymujemy szkieletowy moduł DLL lub EXE, wraz z typowymi elementami implementacji (np. w przypadku DLL funkcjami *DllGetClassObject, DllCanUnloadNow, DllRegisterServer* i *DllUnregisterServer*). Moduł ten nie zawiera początkowo żadnych klas. Poniżej przedstawiono sposób implementacji dwóch prostych klas COM.

TWORZENIE PROSTYCH KOMPONENTÓW – LICZNIK ZWYKŁY

Klasy komponentów tworzy się za pomocą narzędzia dostępnego w menu poprzez *Insert|New ATL Object*. Następnie, korzystając z narzędzia *Class View* można wprowadzać metody i właściwości.

Poniżej przedstawiono szczegółowy przepis na stworzenie prostej klasy komponentów implementujących licznik zliczający w górę:

1. Utwórz nowy projekt (*File|New*) wybierając pozycję *ATL COM AppWizard*. Przyjmij domyślne ustawienia kreatora (powstanie moduł DLL).
2. Wprowadź nowy obiekt (właściwie: klasę): wybierz opcję *Insert|New ATL Object*.
3. Gdy na ekranie pojawi się okno *ATL Object Wizard*, wybierz pozycję *Objects* i kliknij ikonę *Simple Object*, a następnie naciśnij przycisk *Next*.
4. Wpisz w pole tekstowe *Short Name* nazwę tworzonego komponentu. Kliknij OK.
   W tym momencie mamy już utworzoną klasę komponentu. Oznacza to miedzy innymi, że zaimplementowany jest już interfejs *IUnknown*, *IFactoryClass* oraz *IDispatch* (utworzona klasa ma interfejs podwójny (dual) – o czym zadecydowało jedno z ustawień w kreatorze).
5. W obszarze *ClassView* wybierz utworzony dopiero co interfejs i wybierz w menu kontekstowym *Add Method*.
6. W oknie dialogowym utwórz metodę *Inc* – bezparametrową. Powtórz operację dla metody *Reset*.
7. Ponownie korzystając z *ClassView* utwórz właściwość *Value*. Określ dla niej typ *long* i wyłącz zaznaczenie pola *Put Function* (w ten sposób utworzymy właściwość tylko do odczytu).
8. W obszarze *ClassView* wybierz klasę implementującą interfejs i wybierz opcję *Add Member Variable*. Dodaj zmienną składową *long m_val*.
9. Dodaj inicjalizację zmiennej w konstruktorze: *m_val = 0;*
10. W pliku implementacyjnym uzupełnij implementację metod (należy dodać tylko wyróżnione linijki, pozostała część kodu jest wygenerowana automatycznie.):

```
STDMETHODIMP CCounter::Inc()
{
        m_val++;
        return S_OK;
}
```

```
STDMETHODIMP CCounter::Reset()
{
        m_val = 0;
        return S_OK;
}
```

```
STDMETHODIMP
 CCounter::get_Value(long *pVal)
{
        *pVal = m_val;
        return S_OK;
}
```

W tym momencie klasa jest już gotowa do użycia. Można ją przetestować na przykład za pomocą niewielkiego programu w *Visual Basicu*.

TWORZENIE KOMPONENTÓW GENERUJĄCYCH ZDARZENIA – LICZNIK WSTECZ

Zdarzenia są generowane za pośrednictwem interfejsu punktu połączenia *IConnectionPoint*. Poniżej przedstawiono klasę liczników zliczających w dół, wysyłających sygnał po osiągnięciu zera:

1. Utwórz projekt podobnie, jak w przypadku poprzedniej klasy. Możesz też utworzyć nową klasę w ramach tego samego modułu i projektu.
2. Utwórz klasę liczników wstecz bazując na szablonie *Simple Object*. **Uwaga**: w kreatorze obiektów COM przejdź do zakładki *Attributes* i wskaż opcję *Support Connection Points*. Możesz też zaznaczyć opcję *Support ISupportErrorInfo*.
3. Utwórz metodę *Dec* oraz właściwość *Value* (tym razem do odczytu i zapisu).
4. Tak, jak poprzednio, utwórz zmienną *m_val*, w której przechowywana będzie wartość atrybutu *Value*. Zainicjalizuj ją w konstruktorze.
5. Stwórz implementację funkcji składowych:

```
STDMETHODIMP CCounter::Dec()
{
        m_val--;
        return S_OK;
}
```

```
STDMETHODIMP
CCounter::get_Value(long *pVal)
{
        *pVal = m_val;
        return S_OK;
}
```

```
STDMETHODIMP CCoun-
ter::put_Value(long newVal)
{
        m_val = newVal;
        return S_OK;
}
```

   Utworzona klasa przypomina podstawową klasę liczników. Obecnie dodamy obsługę *IConnectionPoint*.
6. Odnajdź w *ClassView* pozycję odpowiadającą interfejsowi połączeń, np. *_ICounterEvents*.
7. Za pomocą menu kontekstowego i opcji (uwaga!) *AddMethod* dodaj zdarzenie *Zero*. *Return type* ustaw na *void*.
8. Skompiluj projekt. Następny punkt będzie realizowany w oparciu o utworzoną podczas tej kompilacji bibliotekę typów[*].
9. Uruchom implementację punktu połączeń: w tym celu w *ClassView* zaznacz nazwę klasy liczników i wybierz opcję *Implement Connection Point*. Zaznacz wyświetloną w okienku nazwę interfejsu. Ten krok należy powtórzyć za każdym razem, gdy doda się nowe zdarzenie.
   W tym momencie zostanie stworzona implementacja dla klasy połączenia (zob. plik *CP.H). W klasie tej dostępna jest metoda *Fire_Zero* wyzwalająca zdarzenie. Uwaga – zdarzenia należy interpretować jako wywołania interfejsu, który zaimplementowany jest po stronie klienta.
10. Zmień stosownie funkcję *Dec*:

```
STDMETHODIMP CCounter::Dec()
{
        m_val--;
        if (!m_val) Fire_Zero();
        return S_OK;
}
```

11. Jeżeli w p. 2 włączono opcję *Support ISupportErrorInfo*, możesz uruchomić mechanizm diagnostyki błędów, wprowadzając nastepującą linijkę kodu na początku metody *Dec*:

```
STDMETHODIMP CCounter::Dec()
{       if (m_val <= 0) return Error("Próba zejścia poniżej zera!");
```

Wypróbuj klasę za pomocą prostego programu klienckiego.

PROPOZYCJE ZADAŃ DO WYKONANIA W RAMACH ZAJĘĆ LABORATORYJNYCH

1. Utworzyć prostą klasę komponentów.
2. Utworzyć klasę komponentów wysyłających zdarzenia.
3. W oparciu o szablon *Full Control* dostępny w pozycji *Controls* po wybraniu opcji *Insert|New ATL Object* stworzyć projekt kontrolki *ActiveX*. Kontrolka powinna implementować metody, właściwości i generować zdarzenia. Zaimplementować też co najmniej jedną spośród *Stock properties.*

   Uwaga! Określenie wyglądu kontrolki wymaga przepisania metody *OnDraw*. Po zmianie wartości atrybutu, który ma wpływ na wygląd kontrolki, należy odświeżyć widok wysyłając *FireViewChange()*.

   Aby dodać do kontroli funkcje obsługujące komunikaty Windows należy zastosować *CallView* i wybrać opcję *Add Windows Message Handler...*

[*] Niektóre wersje Visual Studio błędnie rozpoznają identyfikator *IID__I???Events* jako niezdefiniowany. Należy go wówczas ręcznie zamienić na *DIID__I???Events*.