**Создание COM компонента на Visual C++**

Воробьёв Станислав

В этой статье я хотел бы описать создание простенького COM компонента на Visual C++, он, в принципе, ничего полезного делать не будет, он просто послужит моделью для создания COM'ов, которые будут более функцианальными.

Итак приступим, откройте Visual C++, в меню New выберите ATL COM AppWizard,затем напишите в поле Project Name: MyCom и нажмите Ok. В этом Wizard'е всего 1 шаг, вы должны выбрать тип, выберите DLL, вы также можете, например, добавить поддержку MFC, но мы не будем этого делать, т.к наш компонент не будет каким-то очень замудрёным, он будет простым и будет служить для ознакомительных целей, поэтому не надо нагружать его дополнительными килобайтами библиотеки MFC.

Проект создан, теперь надо добавить в него COM-объект, нажмите на Insert в главном меню и выберите New ATL Object...

Выберите категорию Objects и объект Simple Object. Теперь вы видите диалоговое окно, щёлкните по вкладке Names, В поле Short Name введите имя компонента, который вы предполагаете создать, пусь это будет MyObject, заметьте что все остальные поля заполняются автоматически, рекомендуется так и оставить.

Если хотите, можете изменить поле type, это просто описание COM'а, давайте введём My first Class. Теперь щёлкните по вкладке Attributes, выберите Single threading model, Custom interface и No Aggregation. Всё, вы создали компонент, ну а теперь надо сделать его рабочим. Во вкладке ClassView вы видите созданный вами класс CMyObject и интерфейс IMyObject, он нужен для создания библиотеки типов.

Выберите интерфейс IMyObject во вкладке ClassView, щёлкните по нему правой кнопкой мыши и из контекстного меню выберите Add Method...

Теперь вы видите перед собой диалоговое окно Add Method to Interface, в поле Method Name введите ShowMessageBox, а в поле parameters введите:[in] const BSTR StringToWrite, [out,retval] long \*Result.Поясняю в кратце эту строку: [in] указывает на то, что StringToWrite вводится в функцию, а [out] на то, что этот параметр возвращается, [retval] означает, что этот параметр будет возвращаться всей функцией, это нужно для того, чтобы компонент мог работать, например, в среде Visual Basic, т.к VB не поддерживает тип данных HRESULT, который возвращает эта функция. Вместо того чтобы возвращать данное типа HRESULT, в VB этот метод вернёт данное типа long. Метод создан. В нашем примере он будет показывать пользователю окно сообщение с текстом, находящимся в перемменой StringToWrite.

Давайте теперь создадим свойство строкового типа, которое будет отвечать за заголовок окна сообщения, можно было бы конечно в свойство ShowMessageBox добавить ешё один [in] параметр, который бы отвечал за это, но щас наша цель разработать демонстрационный компонент, поэтому создадим свойство. Щёлкните вновь по интерфейсу IMyObject правой кнопкой мыши и выберите Add Property... В поле Property Type выберите BSTR, а в поле Property Name введите Caption и щёлкните на Ok. Во вкладке ClassView выберите СMyObject->IMyObject вы видите 2 функции: get\_Caption, put\_Caption. Функция put\_Caption вызывается, когда вы присваиваете свойству Caption новое значение, а get\_Caption, когда считываете. В интерфейсе этих функций не будет, можете щас откомпилировать проект и зайти в Visual Basic.

< P> Выберите в менюProject->References, найдите в списке MyCom 1.0 Type Library отметьте его галочкой и щёлкните на Ok. В модуле напишите:

Dim MyObj As New MyObject

Private Sub Form\_Load()

MyObj.

End Sub

Перед вами открывается список методов и свойств, как видите здесь только 1 метод ShowMessageBox и только 1 свойство Caption. Ни то ни другое пока не работает, потому что мы ещё не реализовали метод ShowMessageBox и функции put\_Caption и get\_Caption. Давайте заставим их работать! Сначала реализуем метод ShowMessageBox. Во вкладке ClassViewвыберитеCMyObject->IMyObject->ShowMessageBox и напишите здесь следующий код:

\_bstr\_t temp(StringToWrite);

\_bstr\_t caption(m\_Caption);

\*Result=MessageBox(NULL,temp,caption,MB\_YESNO|MB\_ICONINFORMATION);

temp.~\_bstr\_t();

caption.~\_bstr\_t();

return S\_OK;

Здесь вы видите переменную m\_Caption, которую нам предстоит создать в будущем для связи свойства Caption и компонента, а также наверное незнакомый класс \_bstr\_t. \_bstr\_t предоставляет полезные операторы и методы для работы с типом BSTR, но чтобы его использовать вы должны подключить header comdef.h, откройте MyObject.h и после строки #include "resource.h" // main symbols добавьте #include "comdef.h". В этом коде MessageBox вернёт либо IDYES либо IDNO, в зависимости от того, на какую кнопку нажмёт пользователь, и это значение будет возвращать функция, т.к мы написали \*Result=MessageBox..., а Result возвращаемый параметр. Возможно, тип long для Result был выбран неверно, т.к MessageBox возвращает числа от 1 до 9, но как я уже говорил, этот компонент - просто тест. Теперь осталось реализовать функции put\_Caption и get\_Caption. Чтобы это сделать, надо сначала добавить в класс CMyObject защищённую переменную m\_Caption типа BSTR, её также не будет в интерфейсе, но она будет использоваться, чтобы хранить то значение, которое пользователь присвоит свойству Caption. Откройте файл MyObject.h и после строк:

public:

STDMETHOD(get\_Caption)(/\*[out, retval]\*/ BSTR \*pVal);

STDMETHOD(put\_Caption)(/\*[in]\*/ BSTR newVal);

STDMETHOD(ShowMessageBox)(/\*[in]\*/ const BSTR StringToWrite, /\*[out,retval]\*/ long \*Result);

напишите:

protected:

BSTR m\_Caption;

Здесь же в конструкторе класса напишите:

\_bstr\_t temp("Just a test!");

m\_Caption=temp.copy();

temp.~\_bstr\_t();

Теперь переменная m\_Caption инициализированна, по умолчанию в ней будет находиться строка "Just a test!". Ну а сейчас можно наконец-то заполнить функции get\_Caption и put\_Caption. Откройте файл MyObject.cpp, найдите там функцию get\_Caption и напишите в ней:

\*pVal=m\_Caption;

return S\_OK;

Теперь найдите функцию put\_Caption и напишите в ней следующее:

m\_Caption=newVal;

return S\_OK;

Ну вот и всё в принципе, теперь можно компилировать компонент.