**Использование классов в DLL (Дополнительный лекционный материал)**

Последовательно рассмотрим создание компонентов COM в с++. Нужно создать серверную часть в виде dll, в которую поместить класс, и клиентскую часть (обычное консольное приложение). Начнем с серверной части. Итак создаем приложение типа MFC DLL. Присвоим ему имя ClassDll. Нам нужно реализовать текст компонента в файле (имя присвоим) ClassDLL.cpp. Вот этот файл:

// ClassDLL.cpp : Defines the initialization routines for the DLL.

//

#include "stdafx.h"

#include "ClassDLL.h"

#include <iostream>

#include <windows.h>

#include "IKlass.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

using namespace std;

// CClassDLLApp

BEGIN\_MESSAGE\_MAP(CClassDLLApp, CWinApp)

END\_MESSAGE\_MAP()

// CClassDLLApp construction

CClassDLLApp::CClassDLLApp()

{

// TODO: add construction code here,

// Place all significant initialization in InitInstance

}

// The one and only CClassDLLApp object

CClassDLLApp theApp;

// CClassDLLApp initialization

BOOL CClassDLLApp::InitInstance()

{

CWinApp::InitInstance();

return TRUE;

}

class MyKlass : public IGEN {

public:

MyKlass()

: m\_data(0)

{

cerr << "MyKlass constructor\n";

}

~MyKlass()

{

cerr << "MyKlass destructor\n";

}

void destroy()

{

delete this;

}

int increase\_val(int param)

{

m\_data += param;

return m\_data;

}

private:

int m\_data;

};

extern "C" MyKlass\* create\_klass()

{

return new MyKlass;

}

Все, что мы здесь сделали – это добавили свой класс MyKlass. Этот класс наследуется от интерфейса IGEN, который мы должны определить. Интерфейс нужно определить в заголовочном файле (у нас это IKlass.h). Итак, запомним:

1. Создаем заготовку MFC DLL – при создании проекта
2. Определяем класс cpp
3. Определяем заголовочный файл с классом

В нашем классе MyKlass мы добавили метод

int increase\_val(int param)

{

m\_data += param;

return m\_data;

}

Этот метод просто добавляет число param к переменной класса с именем m\_data.

Теперь важно: мы вставили метод create\_klass для создания экземпляра класса. Без этого метода вся наша техника не имеет эффекта.

extern "C" MyKlass\* create\_klass()

{

return new MyKlass;

}

Обратим внимание на то, что метод является экспортируемым. Т.е. он будет доступен в клиенте через создаваемую dll.

Теперь об интерфейсе IKlass.h. Приводим его содержимое

class IGEN {

public:

virtual void destroy() = 0;

virtual int increase\_val(int param) = 0;

};

В интерфейсе объявлен класс IGEN и метод destroy – деструктор класса, и increase\_val. Несмотря на лишенность последнего всякого практического смысла, мы используем его в клиенте, когда создаем экземпляр класса, упакованного в dll. Ну что ж, с серверной dll все. Создаем ее через меню Build и исправляем ошибки, если есть. Вся dll и сопутствующие файлы помещается в папке Debug (рис.9):

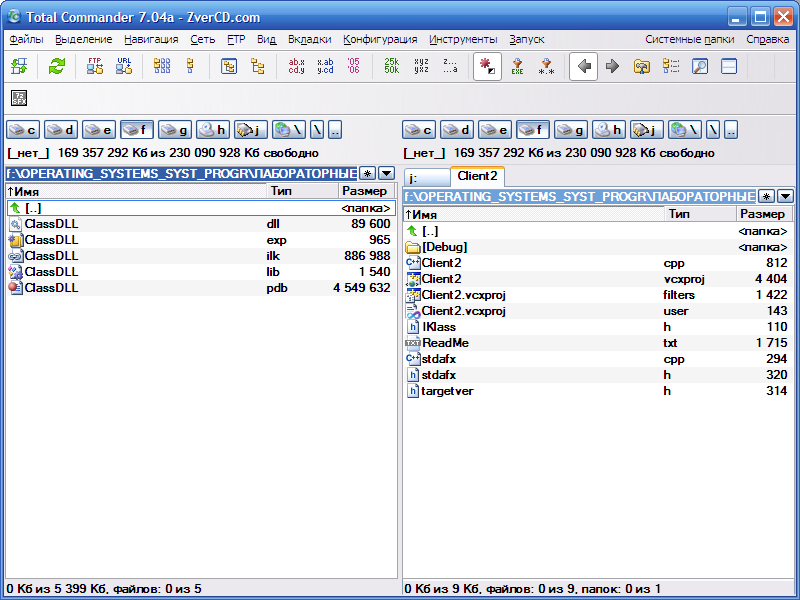


Рис.9 Папка с созданными файлами

Переходим к с созданию клиента. Клиент – это консольное приложение, не dll. Назовем клиентское приложение Client2. Вот его текст на языке C++

// Client2.cpp : Defines the entry point for the console application.

//

#include "StdAfx.h"

#include "windows.h"

#include "string.h"

#include "stdio.h"

#include "stdafx.h"

#include "IKlass.h"

int main()

{

typedef IGEN\* (\*CREATEFUNCPTR)();

TCHAR name2[30]= TEXT("ClassDll.dll");

HINSTANCE hComponent=::LoadLibrary(name2);

if(hComponent==NULL)

{

printf("Error in loading library\n");

getchar();

return NULL;

}

CREATEFUNCPTR ppv=

(CREATEFUNCPTR)::GetProcAddress(hComponent,"create\_klass");

if(ppv==NULL)

{

printf("Cannot find createInstance\n");

getchar();

return NULL;

}

IGEN\* instance = ppv();

int i=instance->increase\_val(10);

printf("The result is %d",i);

getchar();

return 0;

}

Прежде, чем разбираться в этом тексте, нам следует создать заголовочный файл IKlass.h такой же в точности, что и на серверной стороне и вставить его в проект клиента (рис. 10):

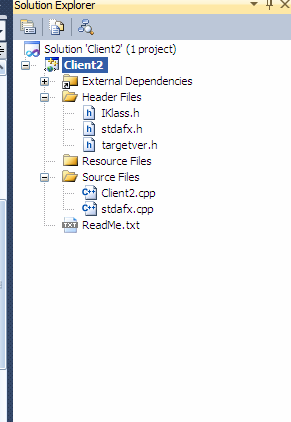


Рис.10 Структура проекта клиента

Последнее, что надо сделать – это перенести файлы dll (скопировать) в папку клиента Debug. Если этой папки нет, то ее нужно создать, отправив клиентское приложение на компиляцию (или создав его, в конце концов вручную!). Подключаем в приложение клиента следующие модули

#include "stdafx.h"

#include "ClassDLL.h"

#include <iostream>

#include <windows.h>

#include "IKlass.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

using namespace std;

Клиентская папка Debug имеет следующее содержимое (рис. 11):

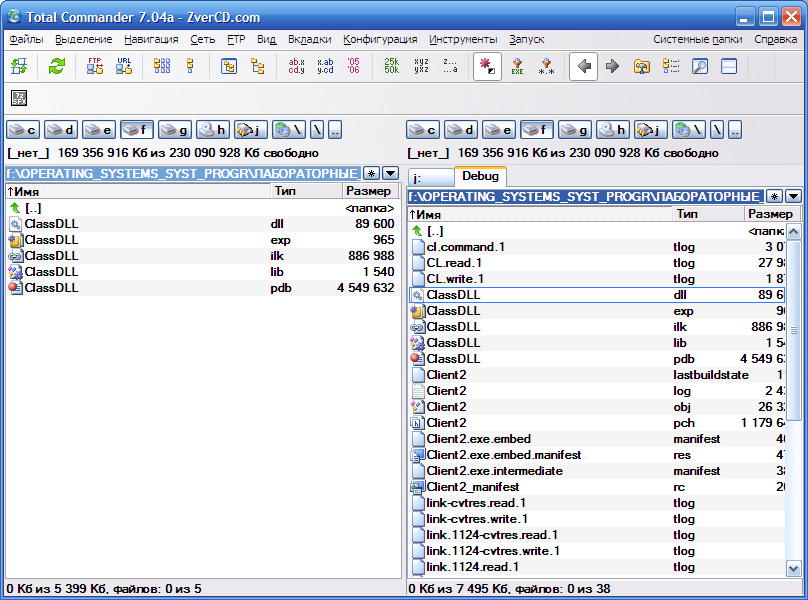


Рис.11 Содержимое клиентской папки debug

Теперь по существу самой клиентской программы. Во-первых, нам нужно загрузить dll в память:

TCHAR name2[30]= TEXT("ClassDll.dll");

HINSTANCE hComponent=::LoadLibrary(name2);

Во-вторых, нужно получить адрес метода для создания экземпляра серверного класса:

CREATEFUNCPTR ppv=

(CREATEFUNCPTR)::GetProcAddress(hComponent,"create\_klass");

Заметим, что create\_klass − это экспортируемый метод из dll, который возвращает экземпляр класса MyKlass (а не IGEN, между прочим!). Здесь легко сбиться с пути истинного.

Итак – переменная ppv – это указатель на метод create\_klass. Теперь мы ее применяем

IGEN\* instance = ppv();

Сейчас нами, наконец, создана ссылка на экземпляр класса MyKlass (хотя по тексту, вроде бы это ссылка на интерфейс IGEN). Но не дадим себя ввести в обман. Итак, instance – это ссылка ссылка на экземпляр класса MyKlass. Ну и все – вызываем метод класса increase\_val:

int i=instance->increase\_val(10);

Запускаем клиентское приложение на выполнение (рис.12):

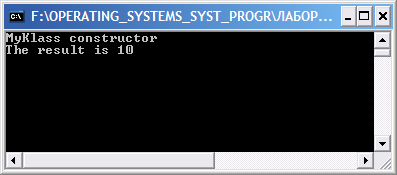


Рис.12 Окно клиентского приложения

Не забудьте подключить интерфейс IKlass.h в клиентское приложение.