**Диалоговая панель**

Очень удобным средством для организации взаимодействия пользователя и приложения являются диалоговые панели. Более того, многие приложения могут успешно работать и без главного окна, взаимодействуя с пользователем только через диалоговые панели. Примером такого приложения может служить приложение Scandisk, входящее в состав операционной системы Windows 95.

Библиотека классов MFC содержит класс CDialog XE "CDialog" , специально предназначенный для управления диалоговыми панелями. Как мы рассказывали в предыдущих томах серии “Библиотека системного программиста”, посвященных программированию для операционных систем Windows и Windows 95, диалоговые панели XE "диалоговые панели" бывают двух типов - модальные и немодальные.

После отображения модальных диалоговых панелей блокируется родительское окно приложения и все его дочерние окна. Пользователь не может продолжить работу с приложением, пока не закроет модальную диалоговую панель.

Немодальные диалоговые панели не блокируют работу остальных окон приложения. Поэтому, открыв такую панель вы можете продолжить работать с приложением - использовать меню, открывать другие дочерние окна и диалоговые панели.

Как ни странно, и модальные и немодальные диалоговые панели обслуживаются одним (общим) классом CDialog, наследованным от базового класса CWnd XE "CWnd" (рис. 2.27).

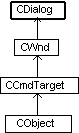


Рис. 2.27. Класс CDialog

В библиотеке MFC версии 1.0 для немодальных диалоговых панелей был предназначен отдельный класс CModalDialog XE "CModalDialog" . Однако, начиная с MFC версии 2.0 он включен в класс CDialog. Для совместимости класс CModalDialog также оставлен, но он определен макрокомандой #define как CDialog (файл Afxwin.h):

#define CModalDialog CDialog

Как создать и отобразить на экране диалоговую панель? В первую очередь необходимо добавить в файл ресурсов приложения шаблон новой диалоговой панели и при помощи редактора ресурсов изменить его по своему усмотрению.

Следующим этапом создается класс для управления диалоговой панелью. Этот класс наследуется непосредственно от базового класса CDialog.

Каждая диалоговая панель обычно содержит несколько органов управления. Работая с диалоговой панелью, пользователь взаимодействует с этими органами управления - нажимает кнопки, вводит текст, выбирает элементы списков. В результате генерируются соответствующие сообщения, которые должны быть обработаны классом диалоговой панели.

Так как класс диалоговой панели обрабатывает сообщения, то он содержит таблицу сообщений и соответствующие методы обработчики сообщений.

Чтобы создать модальную диалоговую панель, сначала необходимо создать объект определенного вами класса диалоговой панели, а затем вызвать метод DoModal, определенный в классе CDialog.

Процедура создания немодальной диалоговой панели несколько другая. Для этого используется метод Create класса CDialog XE "CDialog:Create" . Мы рассмотрим создание немодальных диалоговых панелей позже.

**Приложение с модальной диалоговой панелью**

В этом разделе мы расскажем о том, как создать простейшее приложение с единственной диалоговой панелью. Диалоговая панель будет содержать несколько кнопок, статическое текстовое поле и поле редактирования. В следующей главе мы расскажем, как создать более сложное приложение с главной диалоговой панелью при помощи средств автоматизированного проектирования MFC AppWizard и ClassWizard.

Создайте новый проект под названием MFDialog. В качестве типа приложения выберите из списка Type строку Application (рис. 4.1). Наберите в редакторе исходный текст приложения и сохраните его в файле MFDialog.cpp (листинг 2.13).

Листинг 2.13. Файл MFDialog.cpp

// Включаемый файл для MFC

#include <afxwin.h>

#include "resource.h"

//=====================================================

// Класс CMFDialogApp - главный класс приложения

//=====================================================

class CMFDialogApp : public CWinApp

{

public:

// Мы будем переопределять метод InitInstance,

// предназначенный для инициализации приложения

virtual BOOL InitInstance();

};

// Создаем объект приложение класса CMFDialogApp

CMFDialogApp MFDialogApp;

//=====================================================

// Класс CMyDialog - класс диалоговой панели

//=====================================================

class CMyDialog : public CDialog

{

public:

CMyDialog();

CString m\_Text;

protected:

virtual void DoDataExchange(CDataExchange\* pDX);

// Обработчики сообщений от кнопок диалоговой панели

afx\_msg void OnDefault();

virtual void OnCancel();

virtual void OnOK();

// Макрокоманда необходима, так как класс

// CMyDialog обрабатывает сообщения от органов

// управления диалоговой панели

DECLARE\_MESSAGE\_MAP()

};

// Конструктор клаасса CMyDialog

CMyDialog::CMyDialog() : CDialog(CMyDialog::IDD)

{

// Инициализируем переменную m\_Text

m\_Text = "";

}

//=====================================================

// Метод DoDataExchange класса CMyDialog

//=====================================================

void CMyDialog::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

DDX\_Text(pDX, IDC\_EDIT, m\_Text);

}

//=====================================================

// Таблица сообщений класса CMyDialog

//=====================================================

BEGIN\_MESSAGE\_MAP(CMyDialog, CDialog)

ON\_BN\_CLICKED(IDC\_DEFAULT, OnDefault)

END\_MESSAGE\_MAP()

//=====================================================

// Метод OnDefault класса CMyDialog

//=====================================================

void CMyDialog::OnDefault()

{

// TODO:

m\_Text = "Start Text";

UpdateData(FALSE);

MessageBeep(0);

}

//=====================================================

// Метод OnCancel класса CMyDialog

//=====================================================

void CMyDialog::OnCancel()

{

// Подаем звуковой сигнал

MessageBeep(0);

// Вызываем метод OnCancel базового класса

CDialog::OnCancel();

}

//=====================================================

// Метод OnOK класса CMyDialog

//=====================================================

void CMyDialog::OnOK()

{

// Вызываем метод OnOK базового класса

CDialog::OnOK();

// Подаем звуковой сигнал

MessageBeep(0);

}

//=====================================================

// Метод InitInstance класса CMFDialogApp

//=====================================================

BOOL CMFDialogApp::InitInstance()

{

// Создаем объект класса CMyDialog

CMyDialog dlgTest;

m\_pMainWnd = &dlgTest;

// Отображаем на экране модельную диалоговую панель

dlgTest.DoModal();

// Отображаем на экране значение переменной m\_Text,

// ввходящей в класс CMyDialog

AfxMessageBox(dlgTest.m\_Text);

return FALSE;

}

Создайте файл ресурсов MFDlgRes.rc и добавьте в него новую диалоговую панель. На экране откроется окно редактора диалоговой панели и панель с инструментами Controls (рис. 2.28). По умолчанию новая диалоговая панель называется Dialog и содержит две кнопки OK и Cancel.

Вы можете добавлять в диалоговую панель другие органы управления - кнопки, переключатели, поля редактирования, статические текстовые поля, рисунки. Более того в Visual C++ версии 4.0 вам становятся доступны новые органы управления - многостраничные диалоговые панели, поля для просмотра видеоинформации и т. д.

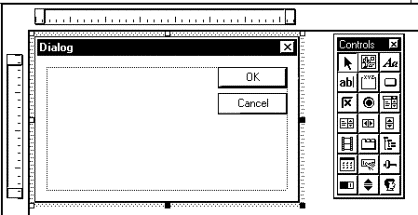


Рис. 2.28. Создание диалоговой панели

В следующей таблице мы кратко описали органы управления диалоговой панели, которые можно добавлять с помощью панели инструментов Controls.

|  |  |  |
| --- | --- | --- |
| Кнопка | Название | Описание |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image085.gif | Select | Если вы нажмете эту кнопку, то сможете выбрать органы управления, которые уже расположены в диалоговой панели |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image086.gif | Picture | Рисунок |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image087.gif | Static Text | Статическое текстовое поле |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image088.gif | Edit Box | Поле редактирования |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image089.gif | Group Box | Прямоугольник, объединяющий группу органов управления |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image090.gif | Button | Кнопка |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image091.gif | Check Box | Переключатель в виде прямоугольника |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image092.gif | Radio Button | Переключатель круглой формы (радиопереключатель) |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image093.gif | Combo Box | Список с окном редактирования |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image094.gif | List Box | Список |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image095.gif | Horizontal Scroll Bar | Горизонтальная полоса просмотра |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image096.gif | Vertical Scroll Bar | Вертикальная полоса просмотра |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image097.gif | Animate | Окно просмотра видео |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image098.gif | Tab Control | Позволяет размещать в диалоговой панели несколько страниц органов управления, каждая из которых имеет “закладку”. Пользователь одновременно видит закладки всех страниц и может выбрать любую из них. |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image099.gif | Tree Control | Позволяет просматривать иерархические (древовидные) структуры данных. Такой орган управления можно использовать для отображения структуры каталогов на диске, заголовков статей справочной системы и т. д. |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image100.gif | List Control | Может использоваться для отображения списка пиктограмм |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image101.gif | Hot Key | Орган управления Hot Key предназначен для ввода комбинации клавиш акселераторов |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image102.gif | Slider | Движок. Обычно используется при отображении видеоинформации. Позволяет перейти к просмотру произвольного кадра |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image103.gif | Progress | Линейный индикатор. Позволяет отображать на экране ход какого-нибудь процесса, например процесса копирования файлов |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image104.gif | Spin | Обычно используется для увеличения или уменьшения значения какого-нибудь параметра |
| http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image105.gif | Custom Control | Орган управления, внешний вид и назначение которого определяется пользователем |

Для нашего первого приложения с диалоговой панелью вам надо добавить только одну кнопку, одно текстовое поле и одно поле редактирования.

Сначала добавьте кнопку. Для этого щелкните по изображению кнопки в панели Controls. Затем переместите указатель мыши в то место диалоговой панели, где вы желаете разместить кнопку и нажмите левую клавишу мыши. В диалоговой панели появится изображение кнопки, названное по умолчанию Button1. Выполните по ней двойной щелчок левой клавишей мыши. На экране появится панель Push Button Propeties, определяющая различные характеристики кнопки. В первую очередь вас будут интересовать поля Caption и ID. В поле Caption введите название кнопки Default, а в поле ID ее идентификатор IDC\_DEFAULT. Остальные характеристики кнопки оставьте без изменения.

Находясь в редакторе ресурсов вы можете сразу попробовать как работает диалоговая панель. Найдите диалоговую панель Dialog (рис. 2.29). Если ее нет на экране, выберите из меню View строку Toolbars и в открывшейся диалоговой панели Toolbars установите переключатель Dialog. Диалоговая панель Dialog содержит ряд кнопок. Первая кнопка, на которой нарисован тумблер, позволяет проверить, как будет работать диалоговая панель.

http://www.frolov-lib.ru/books/bsp/v24/ch2.files/image106.gif

Рис. 2.29. Панель управления Dialog

Остальные кнопки диалоговой панели Dialog позволяют задавать выравнивание органов управления друг относительно друга и относительно границ панели. Последние две кнопки устанавливают разметку на диалоговой панели. Разметка поможет вам ровнее разместить органы управления.

Вы можете изменить заголовок диалоговой панели, ее идентификатор и многие другие параметры, если сделаете двойной щелчок левой кнопкой мыши по заголовку диалоговой панели. На экране появится панель Dialog Properties, содержащая несколько страниц. Выберите страницу General (рис. 2.30).

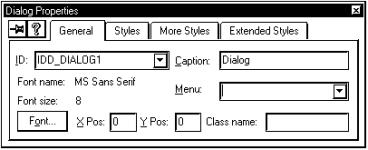


Рис. 2.30. Характеристики диалоговой панели

В поле ID вы можете изменить идентификатор диалоговой панели, в поле Caption - заголовок диалоговой панели. В полях Font name и Font size отображается тип и размер шрифта, который используется для всех текстов в диалоговой панели. Изменить параметры шрифта можно, нажав кнопку Font. Поля X Pos и Y Pos позволяют указать начальное положение диалоговой панели на экране. Если X Pos и Y Pos содержат нулевые значения, начальное положение диалоговой панели определяется операционной системой.

Диалоговая панель может иметь собственное меню. Это меню будет отображаться непосредственно под заголовком диалоговой панели. Если вы желаете подключить к диалоговой панели меню, сначала разработайте меню и запишите его в файл ресурсов. Затем выберите в диалоговой панели Dialog Properties идентификатор данного меню из списка Menu.

Для приложения MFDialog вам надо поменять только идентификатор диалоговой панели и ее название. Измените идентификатор диалоговой панели с IDD\_DIALOG1 на “DIALOGPANEL”, а ее заголовок - с Dialog на My Dialog. Остальные характеристики диалоговой панели оставьте без изменения.

Итак, диалоговая панель “DIALOGPANEL” приложения MFDialog содержит три кнопки, одно статическое текстовое поле и одно поле редактирования. В листинге 2.14 представлен фрагмент файла ресурсов в котором определяется диалоговая панель приложения.

Листинг 2.14. Фрагмент файла MFDlgRes.rc

//////////////////////////////////////////////////////////////

// Диалоговая панель

//

DIALOGPANEL DIALOG DISCARDABLE 0, 0, 186, 46

STYLE DS\_MODALFRAME|DS\_CENTER|WS\_POPUP|WS\_CAPTION|WS\_SYSMENU

CAPTION "My Dialog"

FONT 8, "MS Sans Serif"

BEGIN

DEFPUSHBUTTON "OK",IDOK,129,7,50,14

PUSHBUTTON "Cancel",IDCANCEL,129,24,50,14

PUSHBUTTON "Default",IDC\_DEFAULT,70,7,50,14

EDITTEXT IDC\_EDIT,7,24,113,14,ES\_AUTOHSCROLL

LTEXT "Line Editor",IDC\_STATIC,9,10,34,8

END

Идентификаторы, задействованные в файле ресурсов приложения по умолчанию, определяются во включаемом файле resource.h. Мы привели этот файл в листинге 2.15. Вы можете изменить название включаемого файла, выбрав из меню View строку Resource Includes.

Листинг 2.15. Файл resource.h

//{{NO\_DEPENDENCIES}}

// Включаемый файл, созданный Microsoft Developer Studio

// Используется в файле ресурсов MFDlgRes.rc

//

#define IDR\_MENU 101

#define IDC\_DEFAULT 1000

#define IDC\_EDIT 1001

#define ID\_TEST\_DIALOG 40001

#define ID\_TEST\_EXIT 40002

// Следующие значения идентификаторов используются по

// умолчанию для новых объектов

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NEXT\_RESOURCE\_VALUE 103

#define \_APS\_NEXT\_COMMAND\_VALUE 40003

#define \_APS\_NEXT\_CONTROL\_VALUE 1003

#define \_APS\_NEXT\_SYMED\_VALUE 101

#endif

#endif

Обратите внимание, что включаемый файл resource.h содержит не только определения идентификаторов, но также дополнительную служебную информацию. Она расположена после директивы #ifdef APSTUDIO\_INVOKED и представляет собой ряд макроопределений. Данные макроопределения используются редактором ресурсов при создании новых идентификаторов.

Откройте страницу ClassView в окне Project Workspace. Обратите внимание, что если вы переместите окно Project Workspace к границе главного окна Visual C++, его заголовок пропадет и окно станет похоже на обычную панель управления. Если горизонтальный размер окна Project Workspace уменьшится, тогда исчезнут названия в закладках страниц и останутся только маленькие пиктограммы (рис. 2.31).

В ней отображаются два класса CMFDialogApp и CMyDialog. В главный класс приложения CMFDialogApp входит метод InitInstance. В класс CMyDialog входит конструктор CMyDialog, метод DoDataExchange XE "CWnd:DoDataExchange" , предназначенный для обмена данными между органами управления диалоговой панели и привязанных к ним переменных, а также методы OnOK XE "CDialog:OnOK" , OnCancel XE "CDialog:OnCancel" и OnDefault. Последние три метода вызываются когда пользователь нажимает на кнопки OK, Cancel и Default, расположенные в диалоговой панели. Кроме того, определена глобальная переменная MFDialogApp.

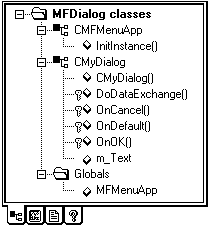


Рис. 2.31. Классы проекта MFDialog

Страница ResourceView окна Project Workspace показывает все ресурсы, входящие в проект (рис. 2.32). В приложении MFDialog определен только один ресурс - диалоговая панель, имеющая идентификатор “DIALOGPANEL”.



Рис. 2.32. Ресурсы проекта MFDialog

Постройте проект и запустите полученный выполнимый файл. На экране появится диалоговая панель My Dialog, определенная в файле ресурсов нашего проекта (рис. 2.33). Вначале поле редактирования Line Editor пустое. В нем вы можете ввести любой текст. Если вы нажмете на кнопку Default, тогда все, что вы ввели в поле редактирования Line Editor, пропадает и в нем отображается строка Start Text.

Кроме кнопки Default в диалоговой панели My Dialog находятся еще две кнопки OK и Cancel. Если вы нажмете кнопку OK, тогда диалоговая панель закрывается, а на экране появляется текстовое сообщение, содержащее текст, введенный вами в поле Line Editor. Если нажать кнопку Cancel, то диалоговая панель My Dialog также закроется, но сообщение с введенным вами текстом не появится.

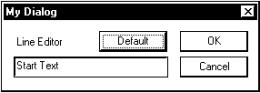


Рис. 2.33. Приложение MFDialog

Рассмотренное нами приложение MFDialog можно создать значительно быстрее, если воспользоваться средствами автоматизированной разработки приложений MFC AppWizard и ClassWizard. Используя эти средства, вы будете избавлены от необходимости вручную набивать в текстовом редакторе код приложения и сможете сконцентрировать свои усилия на реализации обработчиков сообщений от органов управления диалоговой панели.

**Главный класс приложения**

Теперь приступим к рассмотрению исходного текста приложения, представленного в листинге 1.1.

Сначала мы определяем главный класс приложения CMFDialogApp, наследованный от базового класса CWinApp XE "CWinApp" . Класс CMFDialogApp и его методы используются аналогично приложению MFHello. Отличие заключается в том, что переопределенный метод InitInstance вместо отображения на экране сообщения при помощи функции AfxMessageBox XE "AfxMessageBox" , отображает полноценную модальную диалоговую панель.

//=====================================================

// Метод InitInstance класса CMFDialogApp

//=====================================================

BOOL CMFDialogApp::InitInstance()

{

// Создаем объект класса CMyDialog

CMyDialog dlgTest;

m\_pMainWnd = &dlgTest;

// Отображаем на экране модельную диалоговую панель

dlgTest.DoModal();

// Отображаем на экране значение переменной m\_Text,

// входящей в класс CMyDialog

AfxMessageBox(dlgTest.m\_Text);

return FALSE;

}

Сначала создается объект dlgTest класса CMyDialog, который будет представлять диалоговую панель. Когда объект dlgTest создан, диалоговая панель еще не появляется на экране, для этого надо воспользоваться методом DoModal, определенным в классе CDialog.

Следующим оператором мы присваиваем адрес объекта диалоговой панели элементу данных m\_pMainWnd XE "CWinThread:m\_pMainWnd" , входящему в класс CWinApp. Элемент данных m\_pMainWnd определяет главное окно приложения. В нашем случае главное окно как таковое отсутствует и вместо него выступает диалоговая панель.

Чтобы отобразить диалоговую панель на экране, мы вызываем для объекта dlgTest метод DoModal XE "CDialog:DoModal" . На этом выполнение метода InitInstance приостанавливается, пока пользователь не закроет диалоговую панель.

Когда диалоговая панель закрыта, мы отображаем на экране состояние переменной dlgTest.m\_Text, которая соответствует полю ввода Edit диалоговой панели. Последний оператор метода return возвращает значение FALSE и приложение завершается.

**Класс диалоговой панели**

Второй класс, который мы используем в своем приложении, наследуется от базового класса CDialog и носит название CMyDialog. Этот класс предназначен для управления диалоговой панелью приложения.

//=====================================================

// Класс CMyDialog - класс диалоговой панели

//=====================================================

class CMyDialog : public CDialog

{

public:

CMyDialog();

CString m\_Text;

protected:

virtual void DoDataExchange(CDataExchange\* pDX);

// Обработчики сообщений от кнопок диалоговой панели

afx\_msg void OnDefault();

virtual void OnCancel();

virtual void OnOK();

DECLARE\_MESSAGE\_MAP()

};

Обратите внимание на то, как мы определяем конструктор класса CMyDialog. После названия конструктора стоит символ двоеточие и название конструктора класса CDialog. При этом в качестве параметра, конструктору CDialog передается идентификатор диалоговой панели "DIALOGPANEL":

// Конструктор класса CMyDialog

CMyDialog::CMyDialog() : CDialog("DIALOGPANEL")

{

// Инициализируем переменную m\_Text

m\_Text = "";

}

Основное назначение конструктора CMyDialog - вызвать конструктор класса CDialog, указав ему в качестве параметра идентификатор диалоговой панели "DIALOGPANEL". Именно конструктор класса CDialog выполняет создание диалоговой панели.

В конструкторе также инициализируется переменная m\_Text, входящая в класс CMyDialog. В нее записывается пустая строка.

**Обмен данными**

Виртуальный метод DoDataExchange XE "CWnd:DoDataExchange" , который мы переопределяем в классе диалоговой панели, первоначально определен в классе CWnd. Он служит для реализации механизмов автоматического обмена данными - Dialog Data Exchange XE "Dialog Data Exchange" (DDX XE "DDX" ) и автоматической проверки данных - Dialog Data Validation XE "Dialog Data Validation" (DDV XE "DDV" ).

Механизм автоматического обмена данными позволяет привязать к органам управления диалоговой панели переменные или элементы данных класса диалоговой панели. Ряд специальных функций, определенных в библиотеке MFC, вызываются методом DoDataExchange и выполняют обмен данными между органами управления диалоговой панели и соответствующими элементами данных класса диалоговой панели. Такой обмен работает в обоих направлениях. Информация из органов управления диалоговой панели может записываться в элементы данных класса, или в обратном направлении - информация из элементов данных класса может отображаться в диалоговой панели.

Названия всех функций, обеспечивающих обмен данными, начинаются символами DDX\_. Например определены функции DDX\_Text, DDX\_Radio, DDX\_Check, DDX\_Scroll и т. д. Практически каждый тип органов управления диалоговой панели имеет собственную функцию для выполнения процедуры обмена данными.

Все функции DDX\_ имеют три параметра. Первый параметр содержит указатель на объект класса CDataExchange. Этот объект определяет параметры обмена, в том числе направление, в котором надо выполнить обмен данными. Второй параметр определяет идентификатор органа управления, с которым выполняется обмен данными. Третий параметр содержит ссылку на элемент данные класса диалоговой панели, связанный с данным органом управления.

Ниже представлен прототип функции DDX\_, предназначенной для обмена информацией между полем редактирования диалоговой панели и элементом данных, входящим в класс диалоговой панели. Этот элемент имеет класс CString:

void AFXAPI

DDX\_Text(CDataExchange\* pDX, int nIDC, CString& value);

Метод DoDataExchange позволяет выполнять проверку данных, которые пользователь вводит в диалоговой панели. Для этого предназначен ряд функций DDV\_. Эти функции позволяют гарантировать, что данные, введенные пользователем в диалоговой панели, соответствуют определенным условиям.

Если вы используете функцию DDV\_ для проверки ввода в данном органе управления, вы должны вызвать ее сразу после вызова функции DDX\_ для этого же органа управления.

Если функция DDV\_ обнаруживает ошибку пользователя при вводе информации в органе управления диалоговой панели, она отображает сообщение и передает фокус ввода соответствующему органу управления.

В отличие от функций DDX\_, функции DDV\_, в зависимости от их предназначения, имеют различное количество параметров. Первый параметр, как и в случае функций DDX\_, содержит указатель на объект класса CDataExchange XE "CDataExchange" . Остальные параметры имеют различное назначение в зависимости от функции. Так, например, прототип функции, которая проверяет максимальную длину введенной текстовой строки, выглядит следующим образом:

void AFXAPI

DDV\_MaxChars(CDataExchange\* pDX,

CString const& value, int nChars);

Второй параметр содержит ссылку на элемент данных класса диалоговой панели, в который записана текстовая строка из поля редактирования панели. Третий параметр определяет максимально допустимую длину этой строки.

Другие функции DDV\_ имеют больше параметров. Так, функция DDV\_MinMaxInt, проверяющая, что введенное значение находится в определенных границах, имеет 4 параметра:

void AFXAPI

DDV\_MinMaxInt(CDataExchange\* pDX,

int value, int minVal, int maxVal);

Первый параметр, как всегда, содержит указатель на объект класса CDataExchange, второй параметр определяет проверяемую величину, а третий и четвертый параметры устанавливают минимальное и максимальное значения для проверяемой величины.

Приложение не должно напрямую вызывать метод DoDataExchange. Он вызывается через метод UpdateData, определенный в классе CWnd (рис. 2.34). Если вам надо выполнить обмен данными, между органами управления и элементами данных класса диалоговой панели, используйте метод UpdateData XE "CWnd:UpdateData" . Приведем прототип метода UpdateData:

BOOL UpdateData(BOOL bSaveAndValidate = TRUE);

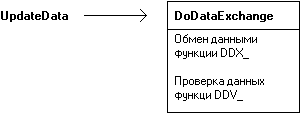


Рис. 2.34. Вызов метода DoDataExchange

Необязательный параметр bSaveAndValidate, определяет, как будет происходить обмен данными. Если метод UpdateData вызывается с параметром FALSE, выполняется инициализация диалоговой панели. Информация из элементов данных класса отображается в органах управления диалоговой панели.

В случае, если метод UpdateData вызван с параметром TRUE, данные перемещаются в обратном направлении. Из органов управления диалоговой панели они копируются в соответствующие элементы данных класса диалоговой панели.

Метод UpdateData возвращает ненулевое значение, если обмен данными прошел успешно и нуль в противном случае. Ошибка при обмене данными может произойти, если данные копируются из диалоговой панели в элементы класса диалоговой панели и пользователь ввел неправильные данные, отвергнутые процедурой автоматической проверки данных.

При создании модальной диалоговой панели перед тем как панель появится на экране, вызывается виртуальный метод OnInitDialog класса CDialog XE "CDialog:OnInitDialog" . По умолчанию OnInitDialog вызывает метод UpdateData и выполняет инициализацию органов управления. Если вы переопределили метод OnInitDialog в своем классе диалоговой панели, в первую очередь вызовите метод OnInitDialog класса CDialog.

Метод UpdateData также вызывается некоторыми другими методами класса CDialog. Так, метод UpdateData вызывается, когда пользователь закрывает модальную диалоговую панель, нажимая кнопку OK. Заметим, что кнопка OK должна иметь идентификатор IDOK XE "IDOK" . Если пользователь нажмет на кнопку Cancel, имеющую идентификатор IDCANCEL XE "IDCANCEL" , то диалоговая панель также закрывается, но метод UpdateData не вызывается и обмен данными не происходит.

//=====================================================

// Метод DoDataExchange класса CMyDialog

//=====================================================

void CMyDialog::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

DDX\_Text(pDX, IDC\_EDIT, m\_Text);

}

Методу DoDataExchange XE "CWnd:DoDataExchange" передается указатель pDX на объект класса CDataExchange. Этот объект создается, когда вы инициируете процесс обмена данными, вызывая метод UpdateData XE "CWnd:UpdateData" . Элементы данных класса CDataExchange определяют процедуру обмена данными, в том числе определяют, в каком направлении будет происходить этот обмен. Обратите внимание, что указатель pDX передается функциям DDX\_ и DDV\_.

В библиотеку MFC входит большое количество функций DDX\_ и DDV\_. Чтобы облегчить задачу написания метода DoDataExchange для класса вашей диалоговой панели, используйте ClassWizard XE "ClassWizard" . Он позволяет привязать к органам диалоговой панели элементы данных класса. При этом метод DoDataExchange создается ClassWizard автоматически. ClassWizard сам выбирает, какие функции DDX\_ и DDV\_ надо использовать для данного органа управления и связанного с ним элемента данных класса диалоговой панели. Подробно об использовании ClassWizard для разработки диалоговых панелей вы можете прочитать в главе “Приложение с главной диалоговой панелью”.

Класс диалоговой панели должен обрабатывать сообщения от своих органов управления, поэтому он должен иметь таблицу сообщений. В заголовке таблицы сообщений указывается имя класса CMyDialog и имя базового класса CDialog XE "CDialog" :

//=====================================================

// Таблица сообщений класса CMyDialog

//=====================================================

BEGIN\_MESSAGE\_MAP(CMyDialog, CDialog)

ON\_BN\_CLICKED(IDC\_DEFAULT, OnDefault)

END\_MESSAGE\_MAP()

Таблица сообщений содержит только одну строку, в которой обрабатывается сообщение с кодом извещения ON\_BN\_CLICKED XE "ON\_BN\_CLICKED" от кнопки “Default”. Когда пользователь нажимает кнопку, вырабатывается данное сообщение и вызывается его обработчик - метод OnDefault, определенный в классе CMyDialog:

//=====================================================

// Метод OnDefault класса CMyDialog

//=====================================================

void CMyDialog::OnDefault()

{

// TODO:

m\_Text = "Start Text";

UpdateData(FALSE);

MessageBeep(0);

}

Две другие кнопки диалоговой панели "DIALOGPANEL", OK и Cancel не представлены в таблице сообщений, но тем не менее в приложении определены методы OnOK и OnCancel, которые вызываются при нажатии на них. Оказывается для диалоговых панелей определены две стандартные кнопки OK и Cancel, которым присвоены специальные идентификаторы IDOK XE "IDOK" и IDCANCEL.

Базовый класс CDialog, также как и класс CMyDialog, содержит таблицу сообщений. Если вы установили библиотеку MFC вместе с исходными текстами, вы можете изучить реализацию класса CDialog в файле Dlgcore.cpp. Сам класс CDialog определен во включаемом файле Afxwin.h. Ниже представлена выдержка из таблицы сообщений, определенной в файле Dlgcore.cpp:

BEGIN\_MESSAGE\_MAP(CDialog, CWnd)

//{{AFX\_MSG\_MAP(CDialog)

ON\_WM\_CTLCOLOR()

ON\_COMMAND(IDOK, OnOK)

ON\_COMMAND(IDCANCEL, OnCancel)

ON\_MESSAGE(WM\_COMMANDHELP, OnCommandHelp)

ON\_MESSAGE(WM\_HELPHITTEST, OnHelpHitTest)

ON\_MESSAGE(WM\_QUERY3DCONTROLS, OnQuery3dControls)

ON\_MESSAGE(WM\_INITDIALOG, HandleInitDialog)

ON\_MESSAGE(WM\_SETFONT, HandleSetFont)

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

Среди прочих сообщений, в этой таблице определены командные сообщения с идентификаторами IDOK и IDCANCEL XE "IDCANCEL" . Для обработки этих командных сообщений определены виртуальные методы OnOK и OnCancel. Данные методы также определены в MFC (файл Dlgcore.cpp). Поэтому когда диалоговая панель содержит кнопки с идентификаторами IDOK и IDCANCEL, как правило нет необходимости создавать для них обработчики.

Так как в таблице сообщений класса CMyDialog отсутствуют макрокоманды для обработки сообщений от кнопок OK и Cancel, они передаются для обработки базовому классу CDialog. Здесь они обрабатываются виртуальными методами OnOK и OnCancel.

Метод OnOK, определенный в классе CDialog, копирует данные из полей диалоговой панели в связанные с ними переменные. Для этого вызывается метод UpdateData с параметром TRUE. Затем выполняется вызов метода EndDialog, который закрывает диалоговую панель и возвращает значение IDOK. После того как диалоговая панель закрыта, метод DoModal возвращает значение IDOK и продолжает работать метод InitInstance.

void CDialog::OnOK()

{

if(!UpdateData(TRUE))

{

// В процессе обмена данными произошла ошибка

TRACE0("UpdateData failed during dialog termination.\n");

return;

}

EndDialog(IDOK);

}

Метод OnCancel, определенный в классе CDialog, еще проще, чем OnOK. Он только закрывает диалоговую панель и возвращает значение IDCANCEL. Копирование данных из полей диалоговой панели не происходит, так как пользователь отменил изменения, нажав кнопку Cancel.

void CDialog::OnCancel()

{

EndDialog(IDCANCEL);

}

Так как методы OnOK и OnCancel определены в классе CDialog как виртуальные, вы можете переназначить их в своем классе CMyDialog. В этом случае управление получат переопределенные вами методы, а не методы класса CDialog. Методы базового класса CDialog можно вызвать, явно указав класс XE "CDialog:OnCancel" .

//=====================================================

// Метод OnCancel класса CMyDialog

//=====================================================

void CMyDialog::OnCancel()

{

// Подаем звуковой сигнал

MessageBeep(0);

// Вызываем метод OnCancel базового класса

CDialog::OnCancel();

}

Переопределенный нами метод OnCancel вызывает функцию программного интерфейса MessageBeep, которая подает звуковой сигнал, а затем вызываем метод OnCancel базового класса CDialog. Метод OnCancel базового класса CDialog вызывается в конце, так как он закрывает саму диалоговую панель.

Аналогично методу OnCancel мы переопределили метод OnOK XE "CDialog:OnOK" .

//=====================================================

// Метод OnOK класса CMyDialog

//=====================================================

void CMyDialog::OnOK()

{

// Вызываем метод OnOK базового класса

CDialog::OnOK();

// Подаем звуковой сигнал

MessageBeep(0);

}

Конечно, наша первая программа использует далеко не все возможности класса CDialog. Другие методы этого класса будут рассмотрены нами позже в главе посвященной автоматизированным средствам проектирования приложений.

**Приложение с немодальной диалоговой панелью**

Процедура создания немодальной диалоговой панели несколько отличается от процедуры создания модальной диалоговой панели. Как и в случае с модальной диалоговой панелью, в первую очередь вы должны создать шаблон диалоговой панели и добавить его в файл ресурсов приложения.

Затем надо создать класс управляющий диалоговой панелью - класс диалоговой панели. Этот класс наследуется непосредственно от базового класса CDialog. В классе диалоговой панели следует определить методы для обработки сообщений от органов управления диалоговой панели.

Следующие шаги выполняются только для немодальных диалоговых панелей. Для класса диалоговой панели надо определить конструктор, объявив его как public. Чтобы открыть немодальную диалоговую панель, создайте объект класса диалоговой панели, обратившись к определенному вами конструктору. В этот момент диалоговая панель еще не появляется на экране. Для этого надо вызвать метод Create XE "CDialog:Create" класса CDialog.

Вы можете вызывать метод Create либо непосредственно из конструктора класса диалоговой панели, либо уже после создания объекта. Если диалоговая панель имеет стиль WS\_VISIBLE, она сразу появится на экране. В противном случае для этого надо вызвать метод ShowWindow. Как видите, для отображения немодальной диалоговой панели мы не использовали метод DoModal.

В классе CDialog определены два прототипа метода Create. Один позволяет указать диалоговую панель через ее текстовое имя, а другой через числовой идентификатор:

BOOL

Create(LPCTSTR lpszTemplateName, CWnd\* pParentWnd = NULL);

BOOL

Create(UINT nIDTemplate, CWnd\* pParentWnd = NULL);

Параметр lpszTemplateName должен содержать указатель на строку, закрытую нулем, содержащую имя ресурса диалоговой панели. Если вы используете метод Create, имеющий второй прототип, вместо имени шаблона, надо указать в параметре nIDTemplate его идентификатор.

Параметр pParentWnd задает родительское окно диалоговой панели. Если в качестве pParentWnd указать NULL, то в качестве родительского окна выступает главное окно приложения.

Метод Create возвращает управление сразу после того, как диалоговая панель отобразилась на экране. Он возвращает ненулевое значение, если создание диалоговой панели завершилось успешно и нуль в противном случае.

Чтобы закрыть немодальную диалоговую панель, можно воспользоваться методом DestroyWindow (метод DestroyWindow определен в классе CWnd и вы можете вызвать его для объектов класса диалоговой панели).

Теперь вам остается удалить объект, управляющий диалоговой панелью. Для этого в классе диалоговой панели можно переопределить виртуальный метод PostNcDestroy XE "CWnd:PostNcDestroy" (этот метод первоначально определен в базовом классе CWnd). В нем вы можете вызвать оператор delete, передав ему в качестве параметра указатель на данный объект this XE "this" .