**创建模态对话框**

CDialog::DoModal

virtual int DoModal();  
  
返回值：  
整数值，指定了传递给CDialog::EndDialog 的nResult参数值。该函数用于关闭对话框。如果函数不能创建对话框，则返回-1；如果出现其它错误，则返回IDABORT。  
  
说明：  
调用该成员函数使用模态对话框并返回对话框结果。当对话框处于活动状态时，该函数处理与用户的交互。这使得对话框是模态的，使用户在关闭对话框之前不能与其它窗口交互。

如果用户单击了对话框中的按钮，如OK或Cancel，那么消息处理函数如OnOK或OnCancel被调用，从而关闭对话框。缺省的OnOK成员函数会对对话框数据进行有效性检验和更新，并关闭它得到结果IDOK。缺省OnCancel函数关闭对话框得到结果IDCANCEL，而不对对话框数据检验或更新，可以覆盖这些消息函数并改变它们的行为。注意 目前PreTransMessage被调用来处理模态对话框的消息。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**模态对话框**（Modal Dialogue Box，又叫做[模式对话框](http://baike.baidu.com/view/3121766.htm)），是指在用户想要对对话框以外的应用程序进行操作时，必须首先对该对话框进行响应。如单击【确定】或【取消】按钮等将该对话框关闭。

一般来说，Windows应用程序中，对话框分为模态对话框和[非模态对话框](http://baike.baidu.com/view/3798288.htm)两种。二者的区别在于当对话框打开时，是否允许用户进行其他对象的操作。

模态对话框垄断了用户的输入。当一个模态对话框打开时，用户只能与该对话框进行交互，而其他用户界面对象收不到输入信息。模态对话框下，用户需要操作目标对话框就必须先操作模态对话框。用户对模态对话框的处理如图所示。

模态对话框的处理过程

[非模态对话框](http://baike.baidu.com/view/3798288.htm)(Nonmodal Dialogue Box，又叫做[无模式对话框](http://baike.baidu.com/view/3121786.htm))，与模态对话框不同，当用户打开非模态对话框时，依然可以操作其他窗口。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**创建非模态的对话框**

用MFC可以创建非模态的对话框，只是创建方式不同，模态对话框用 dlg.DoModel()调用，而非模态对话框要用create函数创建调用。  
/\*假设IDD\_TEST\_DLG为已经定义的对话框资源的ID号\*/  
  
CTestDlg \*dlg=new CTestDlg;  
dlg->Create(IDD\_TEST\_DLG,NULL);  
dlg->ShowWindow(SW\_SHOW);  
在上面的代码中我们新生成了一个对话框对象，而且在退出函数时并没有销毁该对象。因为如果此时销毁该对象（对象被销毁时窗口同时被销毁），而此时对话框还在显示就会出现错误。那么这就提出了一个问题：什么时候销毁该对象。可以用如下方法：  
  
在对话框退出时销毁自己：在对话框中重载OnOK与OnCancel在函数中调用父类的同名函数，然后调用DestroyWindow()强制销毁窗口，在对话框中映射WM\_DESTROY消息，在消息处理函数中调用delete this;强行删除自身对象。相关代码如下：  
void CTestDlg1::OnOK()  
{  
CDialog::OnOK();  
DestroyWindow();  
}  
  
void CTestDlg1::OnCancel()  
{  
CDialog::OnCancel();  
DestroyWindow();  
}  
  
void CTestDlg1::OnDestroy()   
{  
CDialog::OnDestroy();  
   delete this;  
}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**函数Invalidate**

void Invalidate( BOOL bErase = TRUE );

参数: bErase 决定了是否要在WM\_PAINT消息前发送WN\_ERASEBKGND

该函数的作用是使整个窗口客户区无效。窗口的客户区无效意味着需要重绘，例如，如果一个被其它窗口遮住的窗口变成了前台窗口，那么原来被遮住的部分就是无效的，需要重绘。这时Windows会在应用程序的[消息队列](http://baike.baidu.com/view/262473.htm)中放置WM\_PAINT消息。[MFC](http://baike.baidu.com/view/10918.htm)为[窗口类](http://baike.baidu.com/view/1938381.htm)提供了WM\_PAINT的消息处理函数OnPaint，OnPaint负责重绘窗口。视图类有一些例外，在视图类的OnPaint函数中调用了[OnDraw](http://baike.baidu.com/view/5260871.htm)函数，实际的重绘工作由OnDraw来完成。参数bErase为TRUE时，重绘区域内的背景将被擦除，否则，背景将保持不变。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**详细：**

**MFC非模态对话框的创建:**

|  |
| --- |
| 用MFC创建非模态的对话框，和模态对话框创建方式不同，模态对话框用 dlg.DoModel()调用，而非模态对话框要用create函数创建调用。  CTestDlg \*dlg=new CTestDlg;  dlg->Create(IDD\_TEST\_DLG,NULL);  dlg->ShowWindows(SW\_SHOW);//SW\_SHOW是对话框的显示方式 |

   MFC非模态对话框的销毁:

非模态对话框相对于模态对话框，他的创建和销毁过程和模态对话框有一定的区别，先看一下MSDN的原文：When   you   implement   a   modeless   dialog   box,   always   override   the   OnCancel   member  function   and   call   DestroyWindow   from   within   it.   Don’t   call   the   base   class  CDialog::OnCancel,   because   it   calls   EndDialog,   which   will   make   the   dialog   box   invisible   but   will   not   destroy   it.   You   should   also   override   PostNcDestroy  for   modeless   dialog   boxes   in   order   to   delete   this,   since   modeless   dialog  boxes   are   usually   allocated   with   new.   Modal   dialog   boxes   are   usually  constructed   on   the   frame   and   do   not   need   PostNcDestroy   cleanup.

MS的指示：非模态对话框需要重载函数OnCanel，并且在这个函数中调用DestroyWindow。并且不能调用基类的OnCancel，因为基类的OnCancel调用了EndDialog这个函数，这个函数是针对模态对话框的。  
还有一个必须重载的函数就是PostNcDestroy，这也是一个虚函数，通常的非模态对话框是用类的指针，通过new创建的，这就需要在PostNcDestroy函数中delete掉这个指针。

了解了理论过后，下面我们就可以用代码实现一下非模态对话框的创建和销毁过程:  
建立：  
//主框架中，即调用/创建非模态对话框的对话框/窗体：  
CTestDlg \*pDlg=new CTestDlg;  
pDlg->Create(IDD\_TESTDLG,this);  
pDlg->ShowWindow(SW\_SHOW);  
//非模态对话框中：  
void CTestDlg::OnCancel()  
{  
     DestroyWindow();  
}  
void CTestDlg::PostNcDestroy()  
{  
     CDialog::PostNcDestroy();  
     delete this;  
}

如果要在点击按钮的情况下，销毁非模态对话框，只需要把按钮的事件映射到OnCancel函数即可。

以下是一点资料供参考，非模态对话框的销毁顺序：

**MFC应用程序中处理消息的顺序**  
1.AfxWndProc()       该函数负责接收消息，找到消息所属的CWnd对象，然后调用AfxCallWndProc  
2.AfxCallWndProc()   该函数负责保存消息(保存的内容主要是消息标识符和消息参数)供应用程序以后使用，然后调用WindowProc()函数  
3.WindowProc()       该函数负责发送消息到OnWndMsg()函数，如果未被处理，则调用DefWindowProc()函数  
4.OnWndMsg()         该函数的功能首先按字节对消息进行排序，对于WM\_COMMAND消息，调用OnCommand()消息响应函数，对于WM\_NOTIFY消息调用OnNotify()消息响应函数。任何被遗漏的消息将是一个窗口消息。OnWndMsg()函数搜索类的消息映像，以找到一个能处理任何窗口消息的处理函数。如果OnWndMsg()函数不能找到这样的处理函数的话，则把消息返回到WindowProc()函数，由它将消息发送给DefWindowProc()函数  
5.OnCommand()       该函数查看这是不是一个控件通知(lParam参数不为NULL，如果lParam参数为空的话，说明该消息不是控件通知)，如果它是，OnCommand()函数会试图将消息映射到制造通知的控件； 如果他不是一个控件通知(或者如果控件拒绝映射的消息)OnCommand()就会调用OnCmdMsg()函数  
6.OnCmdMsg()         根据接收消息的类，OnCmdMsg()函数将在一个称为命令传递(Command Routing)的过程中潜在的传递命令消息和控件通知。例如：如果拥有该窗口的类是一个框架类，则命令和通知消息也被传递到视图和文档类，并为该类寻找一个消息处理函数

**MFC应用程序创建窗口的过程**  
1.PreCreateWindow()   该函数是一个重载函数，在窗口被创建前，可以在该重载函数中改变创建参数 (可以设置窗口风格等等)  
2.PreSubclassWindow() 这也是一个重载函数，允许首先子分类一个窗口  
3.OnGetMinMaxInfo()   该函数为消息响应函数，响应的是WM\_GETMINMAXINFO消息，允许设置窗口的最大或者 最小尺寸  
4.OnNcCreate()         该函数也是一个消息响应函数，响应WM\_NCCREATE消息，发送消息以告诉窗口的客户区即将被创建  
5.OnNcCalcSize()       该函数也是消息响应函数,响应WM\_NCCALCSIZE消息，作用是允许改变窗口客户区大小  
6.OnCreate()           该函数也是一个消息响应函数，响应WM\_CREATE消息，发送消息告诉一个窗口已经被创建  
7.OnSize()             该函数也是一个消息响应函数，响应WM\_SIZE消息，发送该消息以告诉该窗口大小已经发生变化  
8.OnMove()             消息响应函数，响应WM\_MOVE消息，发送此消息说明窗口在移动  
9.OnChildNotify()     该函数为重载函数，作为部分消息映射被调用，告诉父窗口即将被告知一个窗口刚刚被创建

**MFC应用程序关闭窗口的顺序(非模态窗口)**  
1.OnClose()       消息响应函数，响应窗口的WM\_CLOSE消息，当关闭按钮被单击的时候发送此消息  
2.OnDestroy()     消息响应函数，响应窗口的WM\_DESTROY消息，当一个窗口将被销毁时，发送此消息  
3.OnNcDestroy()   消息响应函数，响应窗口的WM\_NCDESTROY消息，当一个窗口被销毁后发送此消息  
4.PostNcDestroy() 重载函数，作为处理OnNcDestroy()函数的最后动作，被CWnd调用

**MFC应用程序中打开模式对话框的函数调用顺序**  
1.DoModal()             重载函数，重载DoModal()成员函数  
2.PreSubclassWindow()   重载函数，允许首先子分类一个窗口  
3.OnCreate()             消息响应函数，响应WM\_CREATE消息，发送此消息以告诉一个窗口已经被创建  
4.OnSize()               消息响应函数，响应WM\_SIZE消息，发送此消息以告诉窗口大小发生变化  
5.OnMove()               消息响应函数，响应WM\_MOVE消息，发送此消息，以告诉窗口正在移动  
6.OnSetFont()           消息响应函数，响应WM\_SETFONT消息，发送此消息，以允许改变对话框中控件的字体  
7.OnInitDialog()         消息响应函数，响应WM\_INITDIALOG消息，发送此消息以允许初始化对话框中的控件，或者是创建新控件  
8.OnShowWindow()        消息响应函数，响应WM\_SHOWWINDOW消息，该函数被ShowWindow()函数调用  
9.OnCtlColor()           消息响应函数，响应WM\_CTLCOLOR消息，被父窗口发送已改变对话框或对话框上面控件的颜色  
10. OnChildNotify()     重载函数，作为WM\_CTLCOLOR消息的结果发送

**MFC应用程序中关闭模式对话框的顺序**  
1.OnClose()         消息响应函数，响应WM\_CLOSE消息，当"关闭"按钮被单击的时候，该函数被调用  
2.OnKillFocus() 消息响应函数，响应WM\_KILLFOCUS消息，当一个窗口即将失去键盘输入焦点以前被发送  
3.OnDestroy()       消息响应函数，响应WM\_DESTROY消息，当一个窗口即将被销毁时，被发送  
4.OnNcDestroy()     消息响应函数，响应WM\_NCDESTROY消息，当一个窗口被销毁以后被发送  
5.PostNcDestroy()   重载函数，作为处理OnNcDestroy()函数的最后动作被CWnd调用

**打开无模式对话框的顺序**  
1.PreSubclassWindow()     重载函数，允许用户首先子分类一个窗口  
2.OnCreate()             消息响应函数，响应WM\_CREATE消息，发送此消息以告诉一个窗口已经被创建  
3.OnSize()               消息响应函数，响应WM\_SIZE消息，发送此消息以告诉窗口大小发生变化  
4.OnMove()               消息响应函数，响应WM\_MOVE消息，发送此消息以告诉窗口正在移动  
5.OnSetFont()          消息响应函数，响应WM\_SETFONT消息，发送此消息以允许改变对话框中控件的字体

以上这些的执行都是按给定的顺序执行！

引用:

<http://hi.baidu.com/scutsolo/blog/item/8a3780162f334a4c20a4e980.html>

<http://blog.csdn.net/bitxinhai/archive/2008/04/20/2309939.aspx>

以前学VC的时候觉得模态与非模态很简单，两个的创建方式不同，一个是Create一个是DoModel，然后如果是模态的就很霸道，不让你去点后面的父窗口，如果是非模态就可以点击后面的父窗口，但是在实际的应用发现一点很需要注意的地方：

我们如果便写过window SDK，可以知道，其实窗口的创建不关事父窗口还是子窗口，不管是模态还是非模态，其最基本的的调用的都是CreateWindow或者CreateWindowEx，但是就是这两个函数创建的窗口使用起来就很大不同，我们一点一点的说。

首先，一般来说我们都有一个主窗口，这个很简单，在MFC中，这个主窗口可能是一个基于Dialog的，也可能是一个基于View的，这个都比较简单。

好，现在我们要使用模态对话框， 非模态的对话框是独立的，就是说和主程序可以同时交换数据，而模态的对话框则只能和自己交换。换句话说，模态对话框很霸道，你开了它就无法在同一个应用程序中的其他地方干活了，比如那个经常用的openfile dialog。非模态对话框就很随和，开了它你还可以爱干嘛干嘛。

我们创建的代码是：   
  CMyDlg   dlg；    
  dlg.Domdal();

这样你的模态对话框就霸道的出来了，注意这里的CMyDlg 可以是你的成员变量，在这个对话框不关掉之前，你是不能操作其他的东东的，我们可以猜测一下，这个函数干了什么，首先create一个window，然后不让你点击其他的区域。

好，现在我们创建一个非模态的对话框，代码如下;

CMyDlg\*   pMainWnd   =   new   CMyDlg;    
  pMainWnd->Create();

有人或许会问，我能不能把CMyDlg声明为一个变量，而不是一个指针，我试过，同样可以创建，但是发现了一个问题，我的这个对话框就没有独立性了，它就变成了一个子窗口，从属与父窗口，这个从属关系，要比一般的对话框要紧密的多，就相当于一个子窗口控件，可能说的还不是很清楚，子窗口+父窗口=1个窗口，对画框+父窗口=2个窗口，虽然两个窗口关系很紧密（如何紧密我们后面讲），首先，什么是子窗口，子窗口创建在父窗口上，就完全成为了父窗口的一部分了，父窗口说，我的这块地就给你，我不管了，以后你想怎么干久怎么干，干好了跟我说一下就可以了(发送消息)，什么是对话框呢，这里有两块地，一块是我的，一块是你的，两个人都可以想怎么干就怎么干，但是我比较狠，你在一定程度上还得听我的，这个就是对话框，其实就是一块地与两块地的区别，有人可能会说，这个区别有什么影响，我们来说一下影响？最常见的是窗口刷新的时候，如果是子窗口的情况，当父窗口要刷新子窗口的时候，什么事情都不做，对子窗口说，起来了，把你负责的那一块刷一下；如果是对话框，父窗口首先把自己的这一块刷了一下，然后就对对话框说，既然我是老大，你看我都刷新了，是不是你也应该刷新了呢？然后对话框就闷闷不乐的刷新他的那一块。

这点区别有什么用呢？首先，当我们在子窗口中贴一张透明的图的时候，在擦出背景消息里面直接返回的时候，会发现我靠父窗口啥也没有干，显示的是桌面；但是如果是对话框，对话框里面贴的是一张透明的图的时候，会发现我不干，你也还是干了，看到的就不是桌面，而是父窗口对应的区域。

这个问题困扰我很久，感觉一个是指针一个是变量，创建的窗口就完全不同了，郁闷。

下面，我们来说一下模态与非模态的区别，模态很霸道，他把焦点始终强制转移到他自己的身上，当你点击其他区域的时候，会发出警告，提示你必须这个对话框销毁你才能够做别的事情。如果是非模态，如果点击对话框区域，对话框就获得焦点，如果不是，那么焦点就给了其他区域，有人或许会问，这个时候这个对话框会不会被窗口掩盖在下面，答案是不会，虽然不是亲生的父子关系，没有血融于水，但是怎么说也是父子关系，除非是平级的窗口才会这样，当一个窗口获得焦点的时候其他的窗口都在他的下面。

还有一个区别在于内存分配,new在堆中,另一在栈中,一般说两者一样,但如果你在对话框中开了很大的空间做成员,第二种就会有问题了,因为栈空间是受限的。但据说解决栈溢出用static申明，在全局数据区分配内存就可以了（我没试过）!模态消息不进message   loop;非模态消息进入message   loop。

如何使用vc的非模态对话框  
1、非模态对话框的模板必须具有Visible风格，否则对话框将不可见，而模态对话框则无需设置该项风格。更保险的办法是调用CWnd::ShowWindow(SW\_SHOW)来显示对话框，而不管对话框是否具有Visible风格。  
2**、非模态对话框对象是用new操作符在堆中动态创建的，而不是以成员变量的形式嵌入到别的对象中或以局部变量的形式构建在堆栈上。** 通常应在对话框的拥有者窗口类内声明一个指向对话框类的指针成员变量，通过该指针可访问对话框对象。  
3、\*通过调用CDialog::Create函数来启动对话框，而不是CDialog::DoModal，这是模态对话框的关键所在。由于Create函数不会启动新的消息循环，对话框与应用程序共用同一个消息循环，这样对话框就不会垄断用户的输入。Create在显示了对话框后就立即返回，而DoModal是在对话框被关闭后才返回的。众所周知，在MFC程序中，窗口对象的生存期应长于对应的窗口，也就是说，不能在未关闭屏幕上窗口的情况下先把对应的窗口对象删除掉。由于在Create返回后，不能确定对话框是否已关闭，这样也就无法确定对话框对象的生存期，因此只好在堆中构建对话框对象，而不能以局部变量的形式来构建之。

4、代码示例：  
对话框资源ID为：IDD\_MYDIALOG  
对话框的类定义为：CMyDialog  
CMyView的成员变量为：CMyDialog\* m\_myDlg  
  
CMyView::OnOpenDialogButton()  
{  
  CMainFrame\* pWnd=(CMainFrame\*)AfxGetMainWnd();  
  ASSERT\_VALID(pWnd);    //定义父窗口指针pWnd  
  m\_myDlg=new CMyDialog(pWnd);    //堆分配非模态对话框内存空间，   //CMyDialog(CWnd\* pParent=NULL)为构造函数  
  m\_myDlg->Create(IDD\_MYDIALOG,pWnd);  
  m\_myDlg->ShowWindow(SW\_SHOW);  
  //可用this指针代替pWnd指针，则省略头两行  
}

5、必须有一个标志表明非模态对话框是否是打开的。这样做的原因是用户有可能在打开一个模态对话框的情况下，又一次选择打开命令。程序根据标志来决定是打开一个新的对话框，还是仅仅把原来打开的对话框激活。通常可以用拥有者窗口中的指向对话框对象的指针作为这种标志，当对话框关闭时，给该指针赋NULL值，以表明对话框对象已不存在了。  
[注]：在C++编程中，判断一个位于堆中的对象是否存在的常用方法是判断指向该对象的指针是否为空。这种机制要求程序员将指向该对象的指针初始化为NULL值，在创建对象时将返回的地址赋给该指针，而在删除对象时将该指针置成NULL值。

6、必须调用CWnd::DestroyWindow而不是CDialog::EndDialog来关闭非模态对话框。调用CWnd::DestroyWindow是直接删除窗口的一般方法。由于缺省的CDialog::OnOK和CDialog::OnCancel函数均调用EndDialog，故程序员必须编写自己的OnOK和OnCancel函数并且在函数中调用DestroyWindow来关闭对话框。

7、因为是用new操作符构建非模态对话框对象，因此必须在对话框关闭后，用delete操作符删除对话框对象。在屏幕上一个窗口被删除后，框架会调用CWnd::PostNcDestroy，这是一个虚拟函数，程序可以在该函数中完成删除窗口对象的工作，具体代码如下  
void CModelessDialog::PostNcDestroy  
{  
delete this; //删除对象本身  
}  
这样，在删除屏幕上的对话框后，对话框对象将被自动删除。拥有者对象就不必显式的调用delete来删除对话框对象了。也可以通过由的DestroyWindow()引起的WM\_DESTROY消息处理函数OnDestroy（）种实现delete this操作。

void CModelessDlg::OnDestroy()  
{  
 CDialog::OnDestroy();   
 delete this;   
}