第4章对话框

教学内容

- MFC应用程序
- 对话框的使用
- 消息对话框
- 通用对话框

4.1 MFC应用程序

- MFC基本数据类型

Windows数据类型	对应的基本数据类型	说　明
BOOL	bool	布尔值
BSTR	unsigned short *	32位字符指针
BYTE	unsigned char	8位无符号整数
COLORREF	unsigned long	用作颜色的32位值
DWORD	unsigned long	32位无符号整数
LONG	long	32位有符号整数
LPSTR	char *	指向字符串的32位指针
LPARAM	long	作为参数传递给窗口过程函数32位值
LPVOID	void *	指向未定义类型的32位指针
LRESULT	long	来自窗口过程函数的32位返回值
UINT	unsigned int	32位无符号整数
WORD	unsigned short	16位无符号整数
WPARAM	unsigned short	作为参数传递给窗口过程函数32位值
POSITION		用于标记集合中一个元素的位置
LPCRECT		指向一个RECT结构体常量的32位指针

4.1 MFC应用程序

- 句柄

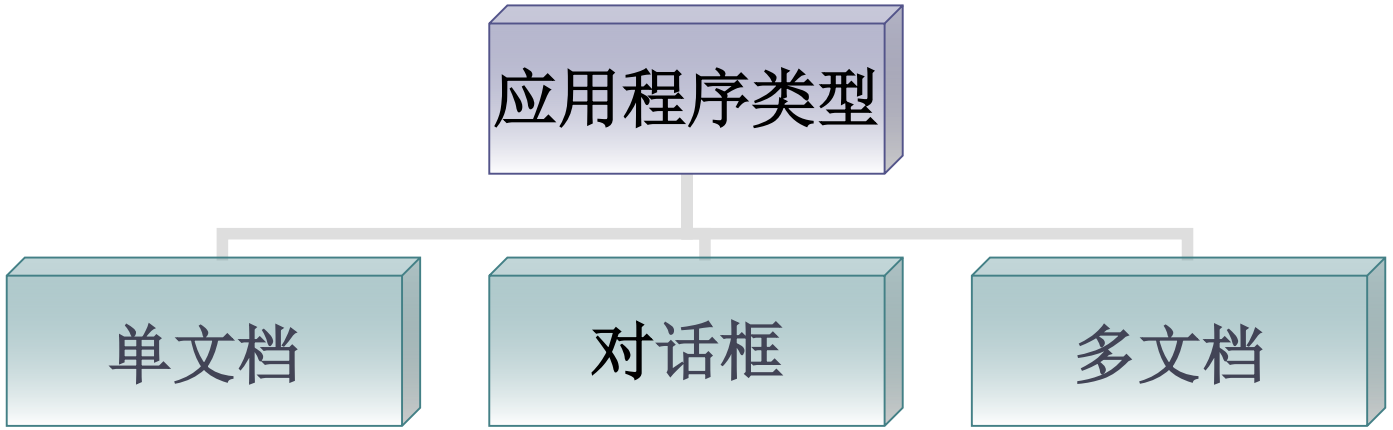
句柄是一个标识**Windows**资源**(**如选单、图标、窗口等**)**和设备等变量。应用程序通过句柄获得相应的对象信息，句柄的值是一个**4**字节长的数值。

句柄类型	说明	句柄类型	说明
HWND	窗口句柄	**HDC**	设备环境句柄
HINSTANCE	运行实例句柄	**HFONT**	字体句柄
HCURSOR	光标句柄	**HPEN**	画笔句柄
HICON	图标句柄	**HBITMAP**	位图句柄
HMENU	菜单句柄	**HBRUSH**	画刷句柄
HFILE	文件句柄		

4.1 MFC应用程序

- **MFC应用程序类型**

MFC AppWizard(exe)为一般的**windows**应用程序框架.包含最常用、最基本的三种应用程序类型：



4.2 对话框的使用

- **对话框概述**

对话框模板–定义对话框特性及对话框控件

对话框类–用来对对话框资源进行管理，提供编程接口

- **创建基于对话框的应用程序**
- **对话框编辑器的使用**

添加控件

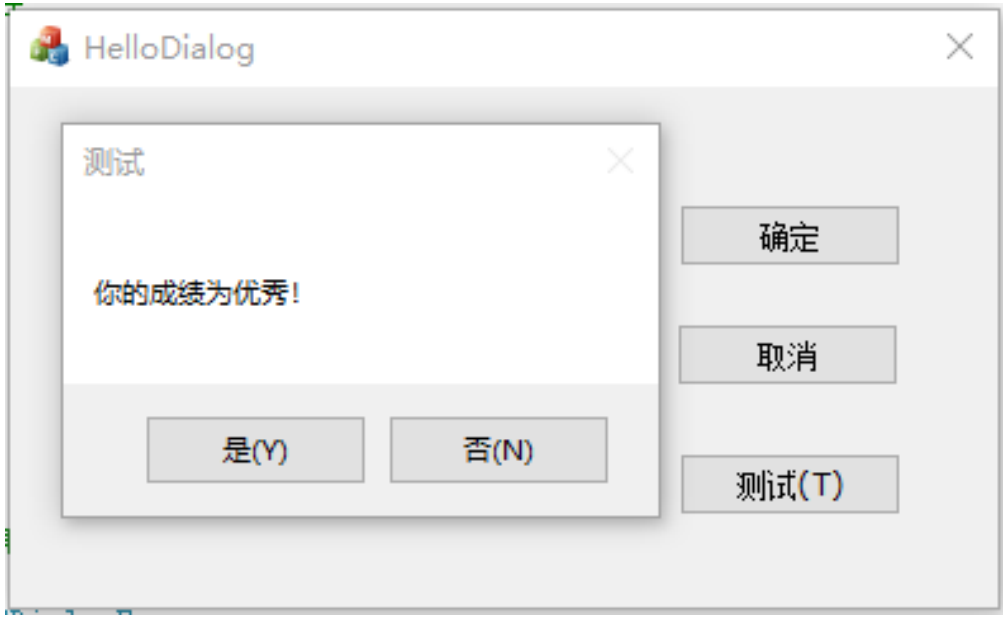
属性设置

添加消息处理函数

为控件添加代码

编译并运行程序

【例4.1】创建一个基于对话框的应用程序，单击“测试”按钮弹出一个消息对话框。



操作一：

创建一个基于对话框的应用程序，具体步骤如下：

① 选择“文件|新建”菜单命令，打开“新建”对话框，选择“项目”标签下的“MFC 应用程序”选项，在“名称”编辑框中输入相应程序项目名称Ex_4_1，在“位置”编辑框中输入相应的文件名和文件路径，单击“确定”按钮。

② 在“MFC AppWizard-MFCApplication1”向导页上选择基于对话框的选项，其它采用默认值。

③ 使用“Ctrl+F5”可编译/链接/运行该项目显示一个空白的，无任何功能的项目。

操作二：

对话框编辑器的使用

1．添加控件

单击控件工具栏按钮控件，并按住鼠标键不放，拖到对话框模板中。

2．设置控件属性

设置按钮ID为“IDC_BUTTONTEST”，标题Caption为“测试(&T)”，Alt+T为选中按钮的快捷键，其它为默认值。

操作步骤三：

建立消息映射(Message Map)

消息映射（Message Map）机制，是指通过一条消息映射宏将一个Windows消息和其消息处理函数联结起来，一一对应。映射一个消息的过程分为以下三步：

① 在处理消息的类中，使用消息宏DECLARE_MESSAGE_MAP()声明对消息映射的支持，并在该宏之前声明消息处理函数。

② 使用BEGIN_MESSAGE_MAP()和END_MESSAGE_MAP()宏来定义消息映射，所有的消息映射宏都添加在它们中间。

③ 定义消息处理函数。

在资源视图上双击测试按钮，或右键添加事件处理程序，选择BN_CLICKED，编辑代码。会自动生成上述三段代码。

操作步骤三：

建立消息映射(Message Map)

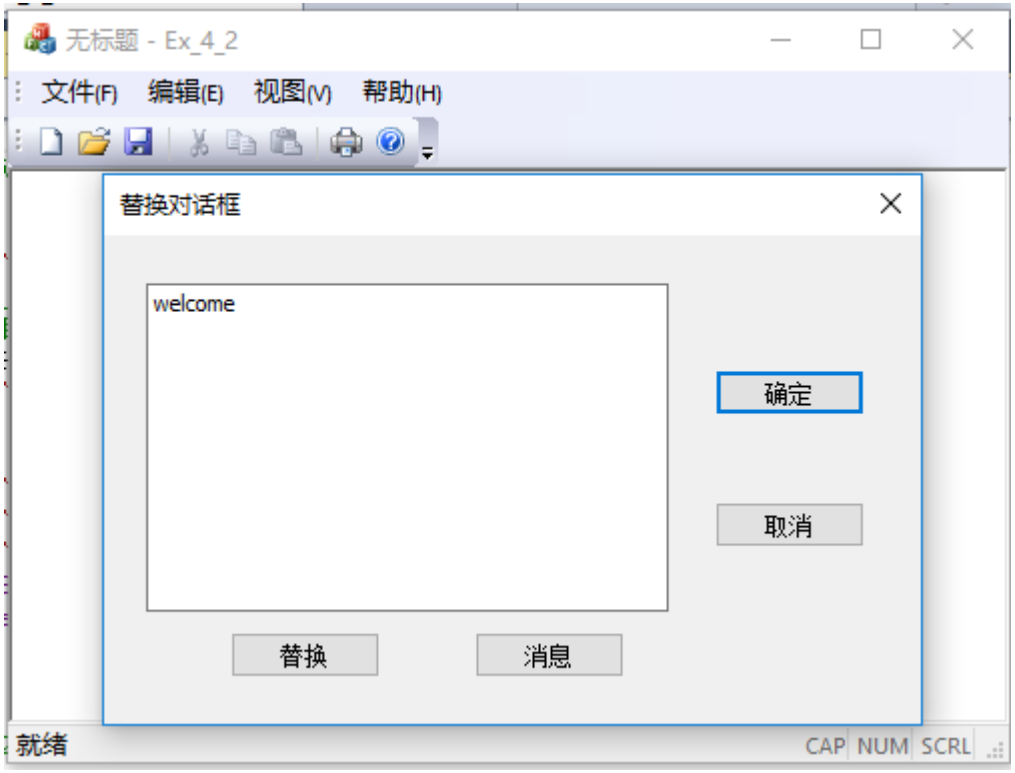
4.为控件添加单击事件处理代码

```
void CEx_4_1Dlg::OnBnClickedButtontest()
{
   // TODO: 在此添加控件通知处理程序代码
   MessageBox(_T("你的成绩为优秀！"),_T("测试
   "),MB_YESNO);
}
```

5. 编译并运行程序

- 添加并使用对话框

【例4.2】创建一个单文档应用程序，并添加对话框资源。



操作步骤：

1.创建一个基于单文档的应用程序Ex_4_2

2.添加对话框资源

选择“视图|资源视图”或者直接点击“解决方案管理器|资源文件| Ex_4_2.rc”，打开资源视图，选中Dialog，右键插入对话框， ID标识为IDD_DIALOG1 。

3.设置对话框属性

将对话框属性ID设置为IDD_DIALOG_REPLACE，标题Caption设置为“替换对话框”，单击字体“Font”按钮，在弹出的字体对话框中设置文本为“宋体，9”。

4.为对话框添加并布局控件

添加2个Button按钮，设置ID属性IDC_BUTTONREPLACE和IDC_BUTTONMAG，和1个编辑框Edit，属性使用默认值。

5. 创建一个对话框类

选择项目|类向导，或者在对话框资源中点击右键，选择类向导，打开类向导对话框，选中添加类，也可以直接在对话框资源中右键点击添加类。

6. 为控件关联变量

打开类向导对话框，选中类名CReplaceDlg，选择成员变量选项卡，为控件ID关联成员变量。

- 控件的数据交换和数据验证

DDX用于初始化对话框中的控件，获取用户的数据输入，其特点是将数据成员变量同相关控件相连接，实现数据在控件之间的传递；

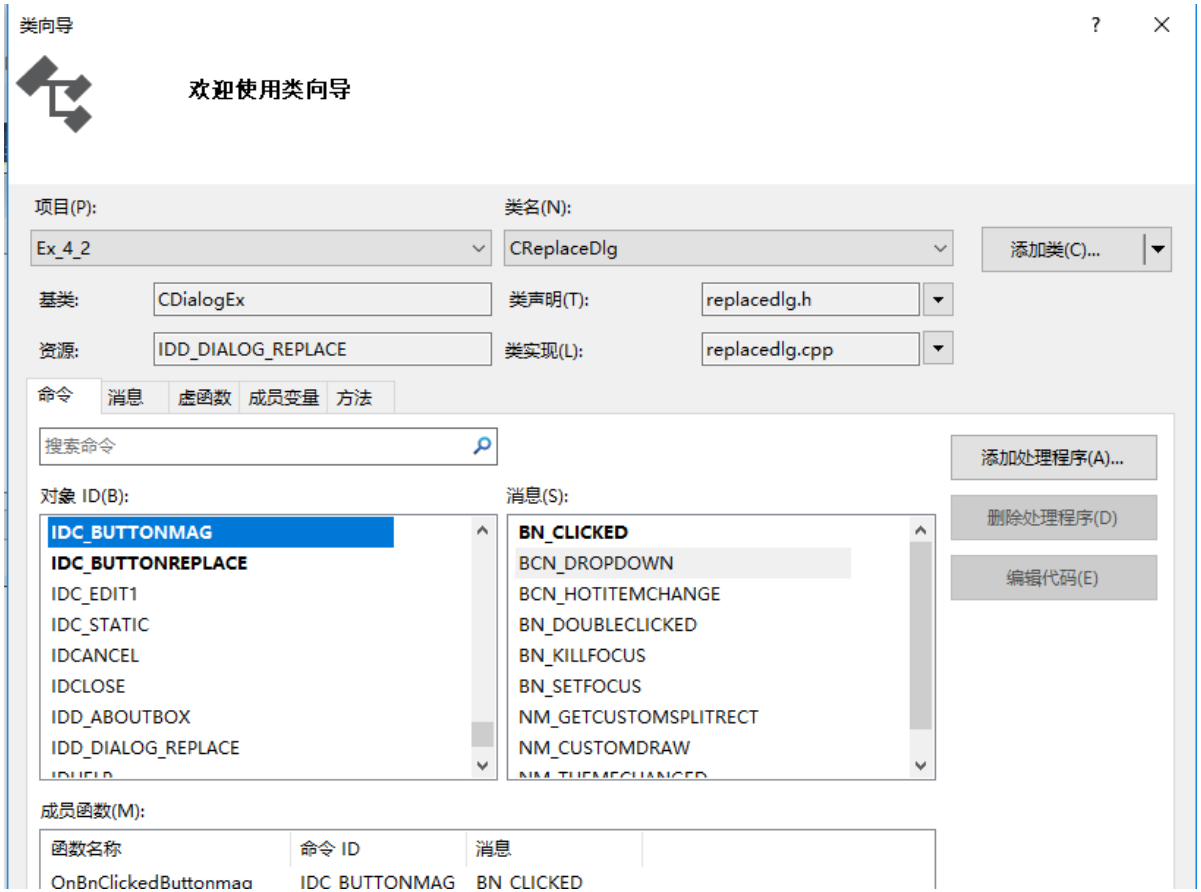
DDV用于验证数据输入的有效性，其特点是自动验证数据成员变量的类型和数值的范围，并发出相应的警告。

完成关联变量后，可以在头文件和源文件中查看生成的代码：

```
void CReplaceDlg::DoDataExchange(CDataExchange* pDX)
{
CDialogEx::DoDataExchange(pDX);
DDX_Control(pDX, IDC_EDIT1, m_Edit1);
DDX_Text(pDX, IDC_EDIT1, m_str);
}
```

7. 添加消息映射代码

打开类向导对话框，选中类名CReplaceDlg，选中对象ID,添加点击时候的消息处理代码。



8.为控件添加代码

```
void CReplaceDlg::OnBnClickedButtonreplace()
{
// TODO: 在此添加控件通知处理程序代码
m_Edit1.ReplaceSel(0,20);
m_Edit1.ReplaceSel(_T("Hello Dialog!"));
}


void CReplaceDlg::OnBnClickedButtonmag()
{
// TODO: 在此添加控件通知处理程序代码
MessageBox(_T("Hello Dialog!"));
}
```

9.对话框的调用

(1) 对话框通过主框架窗口调用，对话框之间也可以相互调用。

用户在Ex_4_2.cpp文件的InitInstance()初始化函数体的“return TRUE;”语句之前加入以下代码：

```
CReplaceDlg d;
d.DoModal();
```

并在Ex_4_2.cpp文件的前面，加入包含CReplaceDlg类的头文件：

#include "ReplaceDlg.h", 将创建的对话框类包含进来。

编译并运行程序单击“替换”按钮，替换的内容在编辑框中显示，单击“消息”按钮，弹出一个消息对话框。

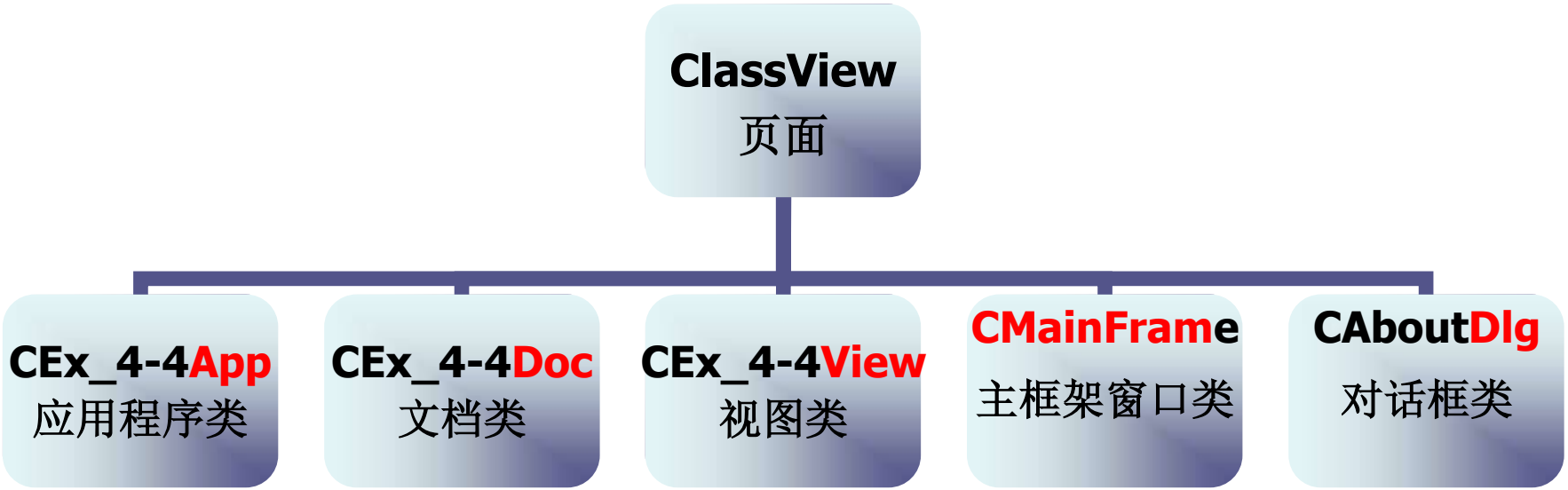
(2) 对话框还可以通过菜单命令调用，具体操作如下：

选择“视图|资源视图”或者直接点击“解决方案管理器|资源文件| Ex_4_2.rc”，打开资源视图，双击Menu下的IDR_MAINFRAME，打开菜单编辑器窗口，在“视图”菜单下添加“调用对话框”菜单，ID为ID_VIEW_DLG。右键添加事件处理程序。

编辑代码：

```
CReplaceDlg d;
d.DoModal();
//也需要添加ReplaceDlg.h
```

- 单文档应用程序结构分析



编写MFC的应用程序，按如下步骤编写程序
① 利用AppWizard应用程序向导创建一个框架。
② 利用资源编辑器为程序添加资源（菜单、对话框、控件等资源）。
③ 利用类向导或手工添加类、类的成员变量和类的函数。
④ 编写需要的代码。
⑤ 编译、链接程序。