第 6 章　属性单和属性页

在 Visual C++ 中，如果选择“工具”→“定制”菜单命令，就会打开如图 6-1 所示的定制对话框，这个对话框就是一个典型的属性单，也称为标签对话框，它的每一个选项卡就是一个属性页，包含一套不同的设置选项。通过标签或按钮来激活各个页面，可以有效地解决大量信息无法在一个对话框上显示的问题，并能提供分类组织管理信息的功能。在程序设计中，可以把相关的选项放在一个属性页中。

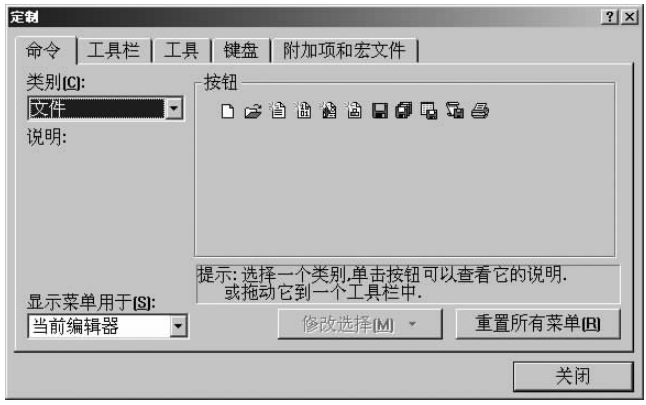

图 6-1　属性单对话框示例

属性单主要分为一般属性单对话框和向导对话框两类。在一般属性单对话框中，页面的切换通过单击不同的标签实现，而在向导对话框中，页面的选择是通过单击“上一步”、“下一步”、“完成”和“取消”等按钮实现的。

6.1　属性单和属性页相关类

MFC 库提供了两个重要的支持属性单的类：CPropertySheet 类和 CPropertyPage 类，它们分别对属性单和属性页进行了封装。其中 CPropertySheet 类是 CWnd 类的一个派生类，作为属性页对话框的窗口框架出现，主要实现管理各个属性页面的作用。一个属性单可以包含一个 CPropertySheet 类（或者其派生类）的对象和多个 CPropertyPage 类（或者其派生类）的对象。

1. CPropertySheet 类

CPropertySheet 类的对象表示属性单，通常由一个或多个属性页对象组成。虽然 CPropertySheet 类不是 CDialog 类的派生类，但其使用却和 CDialog 类非常相似：首先运行 CPropertySheet 类的构造函数，然后调用 DoModal()函数实现一个模式属性单对话框，或者调用 Create()函数实现一个非模式属性单对话框。

CPropertySheet 类提供了很多有用的成员函数，其名称及说明如下：

1）CPropertySheet 函数

其原型为：

```
CPropertySheet();
CPropertySheet(UINT nIDCaption,CWnd * pParentWnd=NULL,UINT iSelectPage=0);
CPropertySheet(LPCTSTR pszCaption,CWnd * pParentWnd=NULL,UINT iSelectPage=0);
```

说明：CPropertySheet 类的构造函数，可以指定父级窗口，并选择当前页面和设置属性页的标题。

2）Construct 函数

其原型为：

```
void Construct(UINT nIDCaption,CWnd * pParentWnd=NULL,UINT iSelectPage=0);
void Construct(LPCTSTR pszCaption,CWnd * pParentWnd=NULL,UINT iSelectPage=0);
```

说明：CPropertySheet 类的另一个构造函数。当定义了 CPropertySheet 类的对象而没有调用构造函数时，如定义了 CPropertySheet 类的对象数组时，必须显式地调用 Construct 构造函数。

3）GetActivePage 函数

其原型为：

```
CPropertyPage * GetActivePage() const;
```

说明：用来返回当前激活的属性页的指针。

4）SetActivePage 函数

其原型为：

```
BOOL SetActivePage(int nPage);
BOOL SetActivePage(CPropertyPage * pPage );
```

说明：用来将指定索引号或将指针的属性页设置为激活页。

5）DoModal 函数

其原型为：

```
virtual int DoModal();
```

说明：显示一个模式属性页。对于一般属性页，返回值为 IDOKIDCANCEL 或者 0；对于向导对话框返回值为 ID_WIZFINISH 或 IDCANCEL。

6）Create 函数

其原型为：

```
BOOL Create(CWnd * pParentWnd=NULL,DWORD dwStyle=(DWORD)-1,DWORD dwExStyle=0);
```

说明：显示一个非模式属性页。

7）AddPage 函数

其原型为：

```
void AddPage(CPropertyPage * pPage);
```

说明：往属性单中增加属性页，并按照调用 AddPage 函数的顺序从左至右显示属性页。在调用 AddPage 函数后，实际上并没有为该页创建相应的窗口，只有当属性页被激活时才为该属性页创建窗口。

8)RemovePage 函数

其原型为：

```
void RemovePage(CPropertyPage * pPage);
void RemovePage(int nPage);
```

说明：从属性单中去除一个属性页，同时删除与之关联的窗口，但是 CPropertyPage 类的对象还存在，只有在属性框窗口被关闭后，CPropertyPage 类的对象才被删除。

9)EndDialog 函数

其原型为：

```
void EndDialog(int nEndID);
```

说明：关闭属性单。

除了这些成员方法之外，该类还有一个 propsheetheader 类型的结构成员 m_psh，该结构定义了属性单的框架和属性页。

2. CPropertyPage 类

该类是从对话框类派生的，它的对象表示单个的属性页，使用方法同使用标准的对话框一样，进行数据交换时，也是通过变量和属性页控件进行数据交换来完成的。

该类实现了对属性页的封装，同时还提供了一些非常有用的成员函数和成员变量，通过这些成员，可以很方便地操作和定制属性页的各个特性，下面重点讲解几个重要的函数：

1)CPropertyPage 函数

其原型为：

```
CPropertyPage();
CPropertyPage(UINT nIDTemplate, UINT nIDCaption=0);
CPropertyPage(LPCTSTR lpszTemplateName, UINT nIDCaption=0);
```

说明：CPropertyPage 类的构造函数。

2)Construct 函数

其原型为：

```
void Construct(UINT nIDTemplate, UINT nIDCaption=0);
void Construct(LPCTSTR lpszTemplateName, UINT nIDCaption=0);
```

说明：用来构造一个 CPropertyPage 类的对象。和 CPropertysheet 类一样，当定义 CPropertyPage 类的对象数组时也要调用 Construct 构造函数。

3)OnApply 函数

其原型为：

```
virtual BOOL OnApply()
{
        ASSERT_VALID(this);
        OnOK();
        return TRUE;
}
```

说明：当单击“应用”(Apply Now)按钮后调用 OnApply 函数，该函数又调用了 OnOK 函数，然后返回 TRUE。在实际编程过程中，常常需要重载此函数，以完成不同的任务。

4)OnOK 函数

其原型为：

```
virtual void OnOK();
{
      ASSERT_VALID(this);
}
```

说明：当单击“确定(OK)”按钮、“应用(Apply Now)”按钮或“关闭(Close)”按钮后调用 OnOK 函数。该函数并没有做任何有意义的工作，只是简单将页标示成干净页。该函数可以被重载。

5)OnSetActive 函数

其原型为：

```
virtual BOOL OnSetActive();
```

说明：当属性页被激活时，调用 OnSetActive 函数。该函数可以被重载。

6)OnKillActive 函数

其原型为：

```
virtual BOOL OnKillActive();
```

说明：当前属性页不再是被激活页时，调用 OnKillActive 函数。该函数可以被重载。

6.2 创建属性单和属性页程序

属性对话框能同时提供多个属性页，而每页都可以由资源编辑器以编辑对话框的方式进行编辑，这样给界面开发带来了方便。

创建属性单和属性页程序的步骤如下：

(1)利用资源编辑器创建属性页资源。

(2)运行 CPropertySheet 类的构造函数。

(3)调用 DoModal()函数实现一个模式属性页对话框，或者调用 Create()函数实现一个非模式属性页对话框。

本节以创建一个单文档应用程序实现画图功能为例，讲解创建属性单和属性页程序的各个步骤，要求如下：

创建一个单文档应用程序实现画圆功能。在主菜单上添加菜单项“设置”，当单击此菜单项时弹出“设置”属性单对话框，该属性单对话框共有三个页面，第一页设置圆的半径；第二页设置圆的边框颜色；第三页设置圆的填充颜色。

注意：要求在初次运行程序时显示一个黑色边框、填充颜色为白色、半径为 50 的圆。

6.2.1 创建属性页资源

创建属性页资源可以通过在“插入资源”对话框中选择属性页来实现，如图 6-2 所示，也可以选择插入一个普通对话框，然后修改其属性使其成为一个属性页。普通对话框和属性页的主要区别有以下两点：

(1)在样式上，普通对话框的样式为“弹出”式，边框为“对话框架”，而属性页的样式为

“子窗口”,边框为“细小”。

(2)在“更多样式”选项卡中,普通对话框的“已禁止”没有选中,这个选项主要控制对话框初始状态下是否可用。

根据这两个不同点进行修改,就可以把一个普通对话框变为一个属性页对话框。

图 6-2　“插入资源”对话框

在属性页上添加布局控件的方法和在普通对话框操作的方法一样,可以根据需要添加不同的控件并为控件添加相关联的对话框成员变量,以实现对话框数据交换功能,完成数据的输入和输出。

针对本例,要完成题目要求的功能,首先需要做如下工作:

(1)利用 MFC AppWizard 向导生成单文档(SDI)应用程序 PropertySheetTest。在工作区的 ResourceView 页面中选择 Menu 并展开,双击 IDR_MAINFRAME 项,弹出菜单资源编辑器,添加“测试”主菜单,其对应的 ID 标识为 ID_DIALOG_TEST。

(2)打开“插入资源”对话框,在 Dialog 下选择 IDD_PROPPAGE_MEDIUM 选项,然后单击“新建”按钮,完成第一个属性页对话框的插入,按照相同的方法添加第二个属性页。

(3)按照要求添加各个页面的主要控件,控件属性如表 6-1 所示。

表 6-1　对话框资源以及控件属性表

对 话 框	控件类型	控 件 ID	设置的非默认属性
设置半径	静态文本	IDC_STATIC	标题为“请输入半径:”
	编辑框	IDC_INPUT_RADIUS	
边框颜色 (填充颜色)	组框	IDC_BORDER_COLOR	标题为“边框颜色:”
	静态文本	IDC_STATIC	标题为“红色(0～255)”
	静态文本	IDC_STATIC	标题为“绿色(0～255)”
	静态文本	IDC_STATIC	标题为“蓝色(0～255)”
	编辑框	IDC_RED	
	编辑框	IDC_GREEN	
	编辑框	IDC_BLUE	

“设置边框颜色”和“设置填充颜色”对话框中的控件基本一致，3 个对话框的 ID 分别为：IDD_DLGRADIUS、IDD_DLGBORDER、IDD_DLGFILL。将 3 个对话框的标题分别设置为“设置半径”、“设置边框颜色”和“设置填充颜色”，这样就完成了对话框资源的创建。使用 Ctrl＋T 组合键查看运行状态，如图 6-3、6-4 和 6-5 所示。

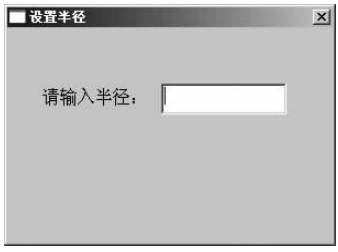

图 6-3 “设置半径”属性页对话框

图 6-4 “设置边框颜色”属性页对话框

图 6-5 “设置填充颜色”属性页对话框

6.2.2 创建 CPropertyPage 类

建立好属性页资源之后，就可以根据这个对话框资源生成一个新类了。此时可以利用 MFC 的 ClassWizard 来实现，但是需要注意的是，创建类时应选择基类 CPropertyPage，ClassWizard 会自动生成相关的代码。

可以通过“查看”→“建立类向导”命令，也可以使用 Ctrl＋W 组合键来打开类向导。由于此时添加的属性页对话框资源并没有相应的类，因此会弹出添加类对话框，如图 6-6 所示，选择“OK”按钮后设置对话框类以及该类的基类即可，如图 6-7 所示。

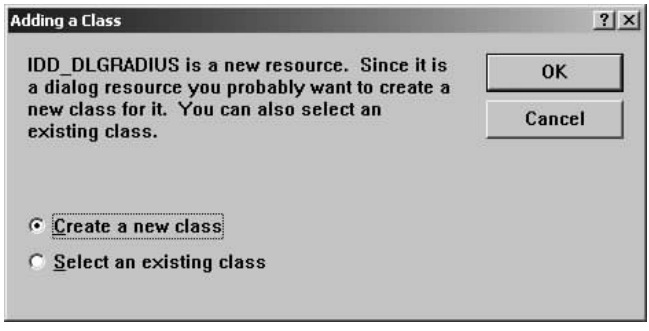

图 6-6 “Adding a Class”对话框

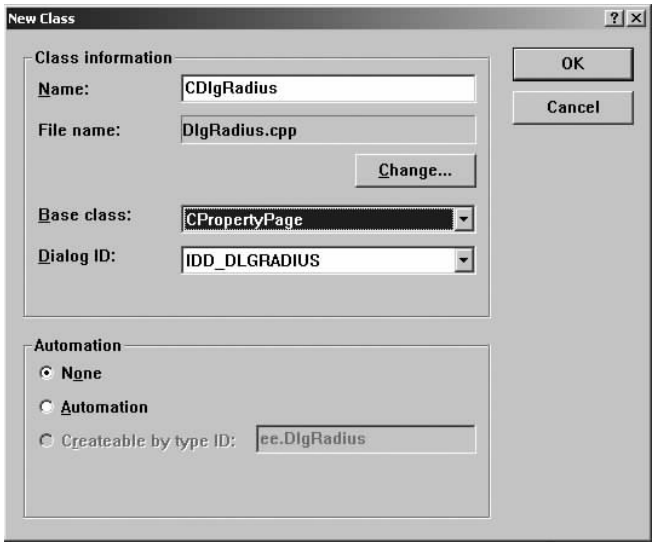

图 6-7　"New Class"对话框

这里"设置半径"对话框的类名称为"CDlgRadius",基类为"CPropertyPage",用同样的方法建立另外两个属性页对话框对应的类"CDlgBorder"和"CDlgFill"。

为了进行数据交换,还需要为每个对话框中需要传递数据的控件添加对话框成员变量,可以通过 Ctrl+W 组合键打开对应的类向导,在 Member Variables 页面添加成员变量,各成员变量属性如表 6-2 所示。

表 6-2　属性页对话框成员变量表

对 话 框	控 件 ID	成员变量	变量类型
设置半径	IDC_INPUT_RADIUS	m_nRadius	UINT
边框颜色 (填充颜色)	IDC_RED	m_nRed	UINT
	IDC_GREEN	m_nGreen	UINT
	IDC_BLUE	m_nBlue	UINT

"填充颜色"对话框和"边框颜色"对话框中的变量一致,另外,为了防止输入颜色值时出现错误,可以通过使用对话框数据验证方式,设置颜色值的最大值为 255,最小值为 0。

6.2.3　创建 CPropertySheet 类

有了属性页资源对话框和对应的类之后,还需要一个对话框框架,用来管理所有的属性页对话框,即属性单。这就需要创建 CPropertySheet 类了。

可以通过单击"插入"→"类"命令来添加新的类,也可以使用 Ctrl+W 组合键打开类向导来添加新的类。两者虽然都可以添加新的类,但最主要的不同点在于:

(1)前者建立的类可以以 CObject 为基类,而后者不可以。

(2)如果编程时需要添加一个不以任何类为基类的新类时,使用前者实现非常方便。

使用两者均可以添加 CPropertySheet 类,可以选择其中的任意一种来完成此工作,使用前者添加时的具体设置方法如图 6-8 所示。

图 6-8 创建 CPropertySheet 类

为了能在属性单中显示已经建立好的 3 个属性页对话框,需要使用属性单的成员函数将属性页添加到属性单中。为了实现这一目的,在 CPropSheet 类中引入 3 个属性页类的头文件。

```
#include "DlgRadius.h"
#include "DlgBorder.h"
#include "DlgFill.h"
```

并且在该类声明中加入 3 个成员变量。

```
public:
    CDlgRadius m_pageRadius;
    CDlgBorder m_pageBorder;
    CDlgFill m_pageFill;
```

在属性单 CPropSheet 的构造函数中将属性页添加上去,可以看到该类有两个构造函数,分别添加如下代码:

```
CPropSheet::CPropSheet(UINT nIDCaption, CWnd * pParentWnd, UINT iSelectPage)
:CPropertySheet(nIDCaption, pParentWnd, iSelectPage)
{
    this->AddPage(&m_pageRadius);
    this->AddPage(&m_pageBorder);
    this->AddPage(&m_pageFill);
}
CPropSheet::CPropSheet(LPCTSTR pszCaption, CWnd * pParentWnd, UINT iSelectPage)
:CPropertySheet(pszCaption, pParentWnd, iSelectPage)
{
      this->AddPage(&m_pageRadius);
      this->AddPage(&m_pageBorder);
      this->AddPage(&m_pageFill);
}
```

6.2.4　显示属性单

要显示属性单，可以调用 DoModal()函数实现一个模式属性单对话框，也可以调用 Create()函数实现一个非模式属性单对话框。使用模式对话框方式实现本例的方法如下：

(1)添加对菜单项“测试”的菜单命令响应函数。打开“ClassWizard”类向导对话框，在“Class name”中选择“CPropertySheetTestView”类，在“Object IDs”中选择“ID_DIALOG_TEST”，添加对 Command 消息的响应，将函数 OnDialogTest()映射到视图类中。

(2)为了保存在属性页对话框中选择的半径和颜色值，在视图类头文件中添加成员变量。

```
...
private:
      UINT m_nRadiusView;
      COLORREF m_borderColor;
      COLORREF m_fillColor;
...
```

在视图类中添加对属性单类的引用。

```
#include "PropSheet.h"
```

(3)使用 CPropertySheet 类的构造函数定义一个类对象，然后调用 DoModal()函数显示模式属性单，并将属性页中输入的值传递给视图类的成员变量。为完成这项功能，需要将下面这段代码添加在 OnDialogTest()函数中。

```
void CPropertySheetTestView::OnDialogTest()
{
    //标题为“属性单窗口”,父窗口是视图类窗口,属性单第一个页面是设置半径属性页
    CPropSheet propSheet("属性单窗口",this,0);
    if (propSheet.DoModal()==IDOK)//模式属性单
    {
        m_nRadiusView=propSheet.m_pageRadius.m_nRadius;     //半径
        m_borderColor=RGB(propSheet.m_pageBorder.m_nRed,    //边框颜色
        propSheet.m_pageBorder.m_nGreen,
        propSheet.m_pageBorder.m_nBlue);
        m_fillColor=RGB(propSheet.m_pageFill.m_nRed,        //填充颜色
        propSheet.m_pageFill.m_nGreen,
        propSheet.m_pageFill.m_nBlue);
        Invalidate();//刷新视图类窗口
    }
}
```

(4)为了能完成初次运行时的画圆要求，还需要在视图类中对半径、边框颜色和填充颜色变量进行初始化，同时要实现按照设置的半径和颜色绘制圆形，此外需要在 OnDraw()函数中添加画圆的程序代码。

```
CPropertySheetTestView::CPropertySheetTestView()
```

```
{
    m_nRadiusView=50;//初始圆半径为 50
    m_borderColor=RGB(0,0,0);//初始圆边框颜色为黑色
    m_fillColor=RGB(255,255,255);//初始圆填充颜色为白色
}
...
void CPropertySheetTestView::OnDraw(CDC * pDC)
{
    CPropertySheetTestDoc * pDoc=GetDocument();
    ASSERT_VALID(pDoc);
    CPen penNew, * pPenOld;
    penNew.CreatePen(PS_SOLID,1,m_borderColor);//创建新画笔
    pPenOld=pDC->SelectObject(&penNew);
    CBrush brushNew, * pBrushOld;
    brushNew.CreateSolidBrush(m_fillColor);
    pBrushOld=pDC->SelectObject(&brushNew);
    pDC->Ellipse(0,0, 2 * m_nRadiusView,2 * m_nRadiusView);
    pDC->SelectObject(pPenOld);
    pDC->SelectObject(pBrushOld);
    penNew.DeleteObject();
    brushNew.DeleteObject();
}
```

程序运行效果如图 6-9 所示。

图 6-9 属性单程序运行效果

可以看到,在程序运行效果图中,除了“确定”和“取消”按钮之外,还有一个“应用”按钮,它们都属于属性单窗口。“应用”按钮在没有添加此按钮消息映射时为灰色,表示不可用,用户可以根据需要添加对此按钮的响应函数。

6.3　创建向导属性单应用程序

向导是属性单的一种，但与普通属性单和属性页不同的是，其多个属性页对话框并不是并列显示，而是有先后顺序的，通常由“上一步”、“下一步”、“取消”和“完成”等按钮指示用户完成设置的过程。使用 MFC AppWizard 创建应用程序的对话框就是一个创建向导的例子，每设置完一个属性页，单击“下一步”按钮即可进行后边的设置，直到单击“完成”按钮结束设置过程。

要实现向导属性单对话框，有几个重要的函数需要调用，其功能说明如下：

1. SetWizardMode 函数

其原型为：

```
void SetWizardMode();
```

说明：用于设置属性页对话框为向导对话框模式。应该在调用 DoModal 函数之前调用 SetWizardMode 函数。

2. SetWizardButtons 函数

其原型为：

```
void SetWizardButtons(DWORD dwFlags);
```

说明：用于在向导对话框中设置按钮的显示方式。在调用 DoModal 函数之后才可以调用 SetWizardButtons 函数。在属性页中可以通过调用 CPropertyPage::OnSetActive 函数来判断是否可调用。

3. SetFinishText 函数

其原型为：

```
void SetFinishText(LPCTSTR lpszText);
```

说明：在向导对话框中，当完成了所有操作时，在“完成”(Finish)按钮上设置需要显示的文字，同时隐藏“上一步(Back)”按钮和“下一步(Next)”按钮。

下面对上节的例子进行修改，使当单击“测试”菜单命令时，弹出向导属性单对话框，设置完成后，在视图类窗口中画出圆形，其他要求不变。

(1)首先生成单文档应用程序，然后添加“测试”菜单。添加 3 个属性页对话框资源和相关控件，生成对应的类后添加和控件相关联的成员变量。

(2)新建属性单类 CWizardSheet，基类为 CPropertySheet，按照上节的内容在该类的头文件中加入 3 个属性页的包含文件，并在类的声明文件中加入属性页成员，最后，在构造函数中使用 AddPage()函数添加 3 个属性页。

(3)为应用程序视图类添加 3 个变量：m_nRadiusView、m_borderCorlor 和 m_fillColor，分别表示半径、边框颜色和填充颜色，并按照题目要求进行初始化工作。

(4)当属性页被激活时，调用 CPropertyPage 类的 OnSetActive 函数，可以在该函数中添加每个属性页的向导按钮。该函数是一个虚函数，所以可以对其重载，添加方法是在要重载虚函数的类上右击，在弹出的菜单中选择“Add Virtual Function...”命令，弹出如图 6-10 所示的对话框，选择 OnSetActive 函数后单击“Add and Edit”按钮即可。

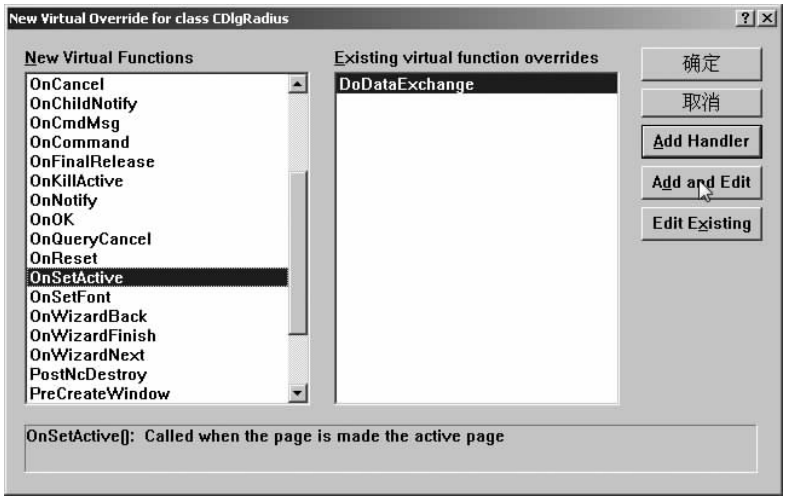

图 6-10 重载虚函数对话框

将“设置半径”属性页设为首页,使该对话框“下一步”按钮和“取消”按钮有效,“上一步”按钮不可用,代码如下:

```
BOOL CDlgRadius::OnSetActive()
{
    //TODO: Add your specialized code here and/or call the base class
    CPropertySheet * pParent=(CPropertySheet * )GetParent();
    ASSERT(pParent->IsKindOf(RUNTIME_CLASS(CPropertySheet)));
    pParent->SetWizardButtons(PSWIZB_NEXT);
    return CPropertyPage::OnSetActive();
}
```

该对话框运行结果如图 6-11 所示。

图 6-11 “设置半径”对话框

“设置边框颜色”对话框为第二页,该页“上一步”、“下一步”和“取消”按钮均可用,代码如下:

```
BOOL CDlgBorder::OnSetActive()
{
    //TODO: Add your specialized code here and/or call the base class
```

```
    CPropertySheet * pParent=(CPropertySheet * )GetParent();
    ASSERT(pParent->IsKindOf(RUNTIME_CLASS(CPropertySheet)));
    pParent->SetWizardButtons(PSWIZB_BACK|PSWIZB_NEXT);
    return CPropertyPage::OnSetActive();
}
```

该对话框运行结果如图 6-12 所示。

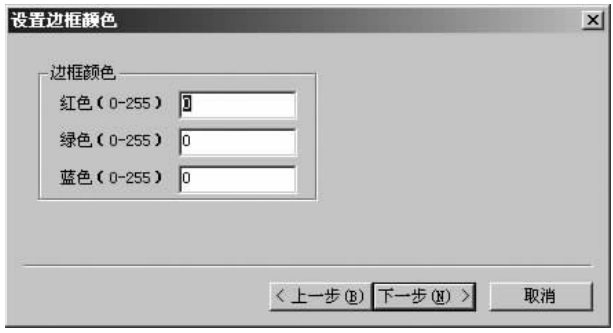

图 6-12　“设置边框颜色”对话框

“设置填充颜色”对话框为最后一页,该页“上一步”和“完成”按钮可用,代码如下:

```
BOOL CDlgFill::OnSetActive()
{
    //TODO: Add your specialized code here and/or call the base class
    CPropertySheet * pParent=(CPropertySheet * )GetParent();
    ASSERT(pParent->IsKindOf(RUNTIME_CLASS(CPropertySheet)));
    pParent->SetWizardButtons(PSWIZB_BACK|PSWIZB_FINISH);
    return CPropertyPage::OnSetActive();
}
```

该对话框运行结果如图 6-13 所示。

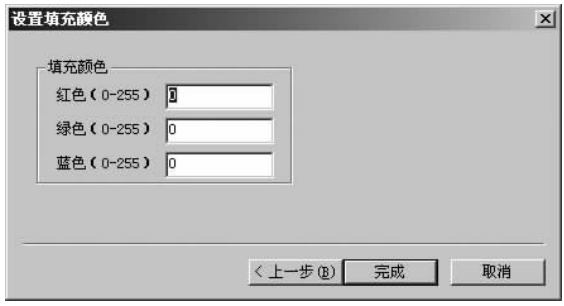

图 6-13　“设置填充颜色”对话框

(5)在视图类头文件中引用 CWizardSheet 类的头文件。

```
#include "WizardSheet.h"
```

(6)为了显示向导属性单对话框,还要对“测试”菜单项添加菜单命令响应函数,并添加如下代码:

```
void CPropertySheetTestView::OnDialogTest()
```

```
    {
        //TODO: Add your command handler code here
        CWizardSheet wizardSheet("向导对话框示例",this,0);
        wizardSheet.SetWizardMode();//设置向导对话框模式
        int nResult=wizardSheet.DoModal();//模式显示向导对话框,得到返回值
        if (nResult==ID_WIZFINISH)//如果在最后一个属性页单击"完成"
        {
            //获取设置半径对话框指针
            CDlgRadius * pageRadius=(CDlgRadius * ) wizardSheet.GetPage(0);
            m_nRadiusView=pageRadius->m_nRadius;//得到设置的半径值
            CDlgBorder * pageBorder=(CDlgBorder * ) wizardSheet.GetPage(1);
             m_borderColor=RGB(pageBorder->m_nRed,pageBorder->m_nGreen,
pageBorder->m_nBlue);
            int zz=propSheet.GetPageCount();
            CDlgFill * pageFill=(CDlgFill * ) wizardSheet.GetPage(2);
            m_fillColor=RGB(pageFill->m_nRed,pageFill->m_nGreen,pageFill->
m_nBlue);
            Invalidate();//刷新窗口
            }
        else
        {
            AfxMessageBox("取消设置");//如果单击取消,弹出消息提示框
        }
    }
```

(7)为了在视图窗口中画出圆形,还需要在 CPropertySheetTestView 类 OnDraw()函数中添加如下代码:

```
    void CPropertySheetTestView::OnDraw(CDC * pDC)
    {
            CPropertySheetTestDoc * pDoc=GetDocument();
            ASSERT_VALID(pDoc);
            //TODO: add draw code for native data here
            CPen penNew, * pPenOld;
            penNew.CreatePen(PS_SOLID,1,m_borderColor);//创建新画笔
            pPenOld=pDC->SelectObject(&penNew);
            CBrush brushNew, * pBrushOld;
            brushNew.CreateSolidBrush(m_fillColor);//创建新画刷
            pBrushOld=pDC->SelectObject(&brushNew);
            pDC->Ellipse(0,0,2 * m_nRadiusView,2 * m_nRadiusView);//根据给定半
径绘制圆形
            pDC->SelectObject(pPenOld);
```

```
        pDC->SelectObject(pBrushOld);
        penNew.DeleteObject();
        brushNew.DeleteObject();
}
```

(8)如果此时运行程序进行向导设置,会发现半径和边框颜色将按照设置生效,填充颜色却始终无法起作用。这是因为当单击最后一页“设置填充颜色”对话框中的“完成”按钮时,前面两页的数据交换已经完成,但最后一页尚未完成。要解决这一问题,必须重载最后一个属性页的 OnWizardFinish()函数,以便完成最后一页的数据交换工作,从而将用户的设置更新到应用程序中,代码如下:

```
BOOL CDlgFill::OnWizardFinish()
{
        UpdateData(TRUE);//数据交换,将控件中颜色数据传递给对话框成员变量
        return CPropertyPage::OnWizardFinish();
}
```

整个程序运行结果如图 6-14 所示。

图 6-14　程序运行结果

在向导对话框设置的任何一个环节如果单击“取消”按钮,就会弹出消息提示框,说明取消了所做的设置,该对话框如图 6-15 所示。

图 6-15　取消设置对话框

6.4　编程实例——在属性单中设置字体

1. 编程要求

实现一个关于属性单的例子,要求通过该属性单可更改字体设置,从而实现对字体、简

单样式以及字体大小等属性的编辑功能。

2. 编程实现

(1)首先创建一个 SDI 项目 FontSet,按向导操作,在步骤 6 对应的对话框中的“基类”下拉列表框中选择“CEditView”,如图 6-16 所示,其他选项均保持默认,单击“下一步”即可,这样可以使程序运行后在客户区出现闪动的输入光标,允许输入字符。

图 6-16 选择视图基类

(2)添加“字体选择”、“字体效果”和“字体大小”3 个对话框模板,并按照图 6-17、图 6-18 和图 6-19 所示的布局为对话框添加控件资源。其中“字体选择”对话框的 ID 为 IDD_FONT_CHOOSE,“字体效果”对话框的 ID 为 IDD_FONT_EFFECT,“字体大小”对话框的 ID 为 IDD_FONT_SIZE。对话框中各个控件的 ID 及对应的成员变量如表 6-3 所示。

图 6-17 “字体选择”对话框

图 6-18 “字体效果”对话框

图 6-19 “字体大小”对话框

表 6-3　属性页对话框成员变量表

对话框	控件类型	控件 ID	成员变量	变量类型	设置的非默认属性
字体选择	静态文本	IDC_STATIC			标题为“系统字体”
	列表框	IDC_FONT_LIST	m_lstFont	CListBox	
字体效果	组框	IDC_STATIC		Caption 为“效果”	
	复选框	IDC_BOLD	m_bBold	BOOL	标题为“粗体”
	复选框	IDC_ITALIC	m_bItalic	BOOL	标题为“斜体”
字体大小	静态文本	IDC_STATIC		Caption 为“高度:”	
	编辑框	IDC_FONT_HEIGHT	m_nHeight	UINT	
	静态文本	IDC_STATIC		Caption 为“宽度:”	
	编辑框	IDC_FONT_WIDTH	m_nWidth	UINT	

(3)添加 3 个对话框模板对应的类,方法参见 6.2.2 节中的内容。需要注意的是,这些类都应该从 CPropertyPage 类中派生。3 个类的名称分别为 CFontChoose、CFontEffect 和 CFontSize。

(4)添加属性单类 CPropSheet,添加方法参见 6.2.3 节中的内容。

为了能在属性单中显示我们已经建立好的 3 个属性页对话框,需要使用属性单的成员函数将属性页添加到属性单中,因此,在 CPropSheet 类中引入 3 个属性页类的头文件:

```
#include "FontChoose.h"
#include "FontEffect.h"
#include "FontSize.h"
```

并且在该类声明中加入 3 个成员变量:

```
public:
    CFontChoose m_pageFont;
    CFontEffect m_pageEffect;
    CFontSize m_pageSize;
```

在属性单 CPropSheet 的构造函数中将属性页添加上去,可以看到该类有两个构造函数,分别添加如下代码:

```
CPropSheet::CPropSheet(UINT nIDCaption,CWnd * p ParentWnd,UINT iSelectPage)
:CPropertySheet(nIDCaption,pParentWnd, iSelectPage)
{
    this->AddPage(&m_pageFont);
    this->AddPage(&m_pageEffect);
    this->AddPage(&m_pageSize);
}
CPropSheet:: CPropSheet ( LPCTSTR pszCaption, CWnd * pParentWnd, UINT iSelectPage)
:CPropertySheet(pszCaption, pParentWnd, iSelectPage)
```

```
{
    this->AddPage(&m_pageFont);
    this->AddPage(&m_pageEffect);
    this->AddPage(&m_pageSize);
}
```

(5)为“字体选择”属性页中的列表框添加备选的字体名称,这个工作可以放在该类的OnInitDialog 函数中来实现。

```
BOOL CFontChoose::OnInitDialog()
{
    CPropertyPage::OnInitDialog();
    //设置列表框显示内容时所用的字体为"楷体_GB2312"
    CFont font;
    font.CreatePointFont(120,"楷体_GB2312");
    m_lstFont.SetFont(&font);
    //添加 4 种字体名称
    m_lstFont.AddString("Courier");
    m_lstFont.AddString("Time New Roman");
    m_lstFont.AddString("Arial");
    m_lstFont.AddString("楷体_GB2312");
    //设置列表框初始选中的字体为第一种字体,即"Courier"字体
    m_lstFont.SetCurSel(0);
    return TRUE; //return TRUE unless you set the focus to a control
    //EXCEPTION: OCX Property Pages should return FALSE
}
```

(6)在工作区的 ResourceView 页面中选择 Menu 并展开,双击 IDR_MAINFRAME 项,弹出菜单资源编辑器,添加“字体设置”主菜单,该菜单对应的 ID 标识为 ID_FONT_SET。

(7) 添加对菜单项“字体设置”的菜单命令响应函数。打开“ClassWizard”类向导对话框,在“Class name”中选择“CFontSetView”类,在 Object IDs 中选择“ID_FONT_SET”,添加对 Command 消息的响应,将函数 OnFontSet()映射到视图类中。

(8)为了保存在属性页对话框中选择的字体,为视图类头文件添加成员变量。

```
...
private:
      CFont * pNewFont;
...
```

在视图类中添加对属性单类的引用。

```
#include "PropSheet.h"
```

(9)使用 CPropertySheet 类的构造函数定义一个类对象,然后调用 DoModal()函数显示模式属性单,并将属性页中输入的值传递给视图类的成员变量,这段代码添加在OnFontSet()函数中,在该函数中还要获取视图窗口包含的编辑控件,并设置生效。

```
void CFontSetView::OnFontSet()
{
    //设置属性单显示时的标题栏
    CPropSheet propSheet("字体设置",this,0);
    if (propSheet.DoModal()==IDOK)//模式属性单
    {
        CString str;//用来获取列表框中的字体名称字符串
        CEdit &MessageBody=GetEditCtrl();//获取视图窗口包含的编辑控件
        //LOGFONT 结构用于说明字体的所有属性,该变量来记录各属性页的输入值
        LOGFONT logft;
        logft.lfHeight=propSheet.m_pageSize.m_nHeight;//获取字体高度
        logft.lfWidth=propSheet.m_pageSize.m_nWidth;//获取字体宽度
        logft.lfEscapement=0;//文本行的倾斜度
        logft.lfWeight=propSheet.m_pageEffect.m_bBold? FW_BOLD:FW_NORMAL;//是否
粗体
        logft.lfItalic=propSheet.m_pageEffect.m_bItalic;//是否斜体
        logft.lfUnderline=0;//不创建下划线
        logft.lfStrikeOut=FALSE;//不创建删除线
         propSheet.m_pageFont.m_lstFont.GetText(propSheet.m_pageFont.m_
lstFont.GetCurSel(),str);//获取字体列表框中选中的字体名称
        strcpy(logft.lfFaceName,str);//将选中的字体名称放入 LOGFONT 结构体变
量中
        //创建新字体
        pNewFont=new CFont;
        pNewFont->CreateFontIndirect(&logft);
        MessageBody.SetFont(pNewFont);//设置编辑控件字体,使生效
        }
}
```

(10)为了使字体大小属性页中字体的宽度和高度有初始值,可以在该属性页对应的类中修改其构造函数。这里将初始值均设置为 10,代码如下:

```
CFontSize::CFontSize() : CPropertyPage(CFontSize::IDD)
{
      //{{AFX_DATA_INIT(CFontSize)
      m_nHeight=10;
      m_nWidth=10;
      //}}AFX_DATA_INIT
}
```

(11)因为在 OnFontSet()函数中创建了新的字体,所以在程序结束时应清除该字体指针。这个清除工作可以在 DestroyWindow()函数中完成。添加 DestroyWindow()的方法是右击 CFontSetView 类,在弹出的菜单中选择“Add Virtual Function...”选项,弹出如

图 6-20所示的对话框，在左边的窗口中选择“DestroyWindow”，然后单击“Add and Edit”按钮进行代码编辑。

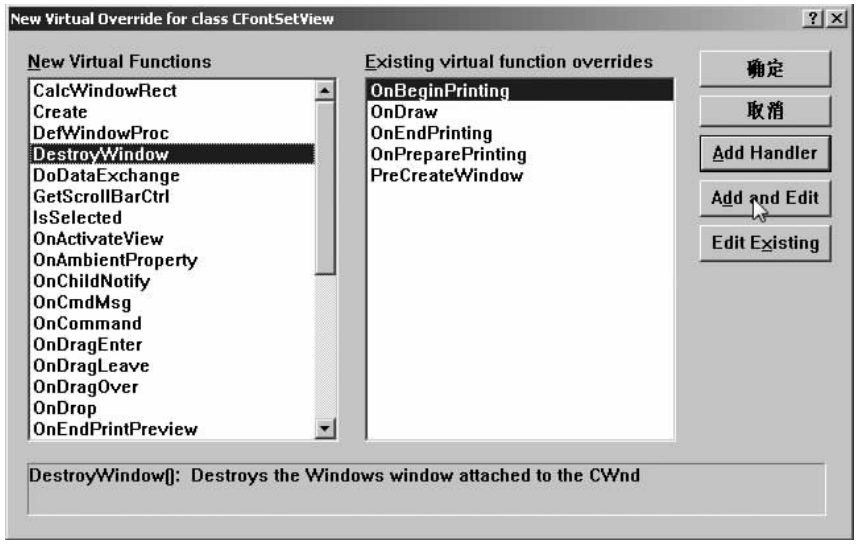

图 6-20　添加 DestroyWindow()函数对话框

```
BOOL CFontSetView::DestroyWindow()
{
        if (pNewFont)
        delete pNewFont;
        return CEditView::DestroyWindow();
}
```

程序整体运行效果如图 6-21 所示。

图 6-21　属性页中设置字体示例图

本 章 小 结

本章详细介绍了 MFC 中属性表的概念、创建以及处理等内容，并对每个环节结合实例进行讲解，以便达到举一反三的效果。通过对 CPropertySheet 类和 CPropertyPage 类成员函数的讲解以及具体实例演示步骤的分析及阐述，让读者了解属性单和属性页应用程序的各层知识，以达到使读者掌握其基本用法的目的。

习　题　6

一、填空题

1. 属性单主要分为_____对话框和_____对话框两类。

2. MFC 库提供了两个支持属性单的重要类，即_____和_____类，分别对属性单和属性页进行了封装。

3. 可以调用_____函数实现一个模式属性单对话框，或者调用_____ 数实现一个非模式属性单对话框。

4. 在属性单编程中，往属性单中增加属性页使用_____ 函数实现。

二、简答题

1. 简述创建属性单和属性页程序的步骤。

2. 普通对话框和属性页对话框有什么样的区别？

3. 请列举 3 个以上 CPropertySheet 类的成员函数，并说明它们所能实现的功能。

4. 请列举 3 个以上 CPropertyPage 类的成员函数，并说明它们所能实现的功能。

三、上机操作

编写属性单对话框应用程序，要求属性单对话框包括 3 个属性页：第 1 个属性页选择职业，第 2 个属性页选择工资水平，第 3 个属性页选择爱好，然后根据选择在视图窗口中显示一句对所做选择的评语。