在对话框中加入属性页

方案一

在对话框上放置一个 Tab Control 的控件，再在对话框上放置所需的控件(本例放置了 2 个按钮，试图在每个标签中显示一个）。然后利用 Class Wizard 来为 Tab Control 控件创建一个控件变量，该变量是 CTabCtrl 类的，再为其他控件也创建相应的控件类。 在主对话框的初始函数中 CProperty1Dlg::OnInitDialog()加入如下代码：

```
//本例插入两个标签，实际运用中可通过循环插入所需个数的标签，运行后默认第一个标签被选中
m_tab.InsertItem( 0, _T("Tab1") );
m_tab.InsertItem( 1, _T("Tab2") );
//将不是第一个标签的控件隐藏掉，只留下你要的控件
m_button2.ShowWindow( SW_HIDE );
```

再利用 ClassWizard 处理 Tab Control 的 TCN_SELCHANGE 的消息。在消息处理函数中，利用 CWnd::ShowWindow 来使相应的控件显示和隐藏。

```
void CProperty1Dlg::OnSelchangeTab1(NMHDR* pNMHDR, LRESULT* pResult)
{
//GetCurSel 返回当前被选中的标签的索引号（以 0 为基础算起）
int sel = m_tab.GetCurSel();

switch(sel)
{
case 0:
m_button1.ShowWindow( SW_SHOW );
m_button2.ShowWindow( SW_HIDE );
break;
case 1:
m_button2.ShowWindow( SW_SHOW );
m_button1.ShowWindow( SW_HIDE );
break;
}

*pResult = 0;
}
```

这样做以后就可以使界面上的控件在不同的标签中显示了，但是这个方案也有很多弊病。

所有的控件仍然在一个对话框内，在使用对话框编辑器进行编辑时，操作很不方便。
为了能分类显示控件，必须用 ClassWizard 为每一个控件创建一个控件变量，以便利用各控件变量的 CWnd 基类的 ShowWindow 函数来显示和隐藏。有时为了使用 DDX 和 DDV 机制来进行数据交换，还要创建一些存放值的变量，这样就使得整个对话框类变得相当庞大难以操作。
当然你也可以使用数组来存放那些控件变量或值变量，但是这样并不是最好，有时一些不相关的控件变量放入一个数组中，通过没有实际意义的数组索引号来访问控件，对程序的编写会造成麻烦。 最好能将所有控件进行分类，放入不通对话框类中，这些对话框作为子对话框出现在主对话框中。可以。现在看看方案二。

方案二

这个方案中，我将使用 MFC 中现成的 CPropertySheet 和 CPropertyPage 类来完成将控件分散到各个对话框类中。

首先加入两个（或数个）对话框资源。修改各对话框资源的属性，将对话框的 Caption 属性改为你要在标签上所显示的文字。将对话框的 Style 属性改为：Child， Border 属性改为：Thin， 只选中 Title Bar 复选框，去掉其他复选框。然后你可以在这些对话框中加入要分开显示的各个控件。

为上述对话框资源分别制作一个对话框类，该对话框类是从 CPropertyPage 继承。这样一来各子对话框类就好了，主对话框类可以直接使用 CPropertySheet 类。使用如下代码即可：

```
CPropertySheet sheet("属性页对话框");
CPage1 page1;
CPage2 page2;
//加入子对话框作为一个属性页
sheet.AddPage(&page1);
sheet.AddPage(&page2);
```

```
//产生一个模态对话框，也可以使用 Create 方法来产生一个非模态对话框（具体参见 MSDN）
sheet.DoModal();
```

这样这个对话框效果如下：

但是会有人问，如何在主对话框中放置其他控件呢？如果直接使用 CPropertySheet 的话，是不可以的，但是别忘了我们可以从 CPropertySheet 类继承自己的类啊！下面来看看方案三的做法。

方案三

首先还是要创建那些要在属性页中的显示的子对话框类，创建步骤和方案二一样，都是从 CPropertyPage 继承。

这次我们将从 CPropertySheet 类继承自己的类（假设类名为 CMySheet）。我们要在这里放上一个 button 控件。那么现在先在 CMySheet 中加入一个 CButton 类的成员变量 m_button。

在 CMySheet 类中的 OnInitDialog()函数里，这样写：

```
BOOL bResult =
CPropertySheet::OnInitDialog();

//取得属性页的大小
CRect rectWnd;
GetWindowRect(rectWnd);
//调整对话框的宽度
SetWindowPos(NULL, 0, 0,rectWnd.Width() +
100,rectWnd.Height(),SWP_NOMOVE |
SWP_NOZORDER | SWP_NOACTIVATE);
CRect rectButton(rectWnd.Width() + 25,
25,rectWnd.Width()+75, 75);
//用程序创建一个按钮
m_button.Create("Button", BS_PUSHBUTTON,
CRect(rectWnd.Width(),
25,rectWnd.Width()+75, 50) , this, 1);
//显示这个按钮
m_button.ShowWindow( SW_SHOW );
CenterWindow();
return bResult;
```

效果如下：

使用方案三虽然能在主对话框中加入控件，但是也比较麻烦，首先所加的控件只能在属性页的右边或下边。并且用程序来产生控件比较烦琐，位置与大小不易控制。那么还有其他方法，既能在对话框中加入属性页，又能在主对话框随意添加控件？还是有的，看看方案四。

方案四

这次我们不从 CPropertySheet 继承自己的类，还是直接使用它。各属性页的子对话框类还是需要的，创建方法和上述两个方案相同。

首先我们新建一个基于对话框的工程。在编辑已有的一个主对话框中可以自由加一些所需的控件，但是得留出一定的空间用于放置属性页。

在主对话框类里加入一个 CPropertySheet 类的一个成员变量（m_sheet）代表整个属性页。再加入一些各子对话框类的实例作为成员变量（m_page1、m_page2……）。

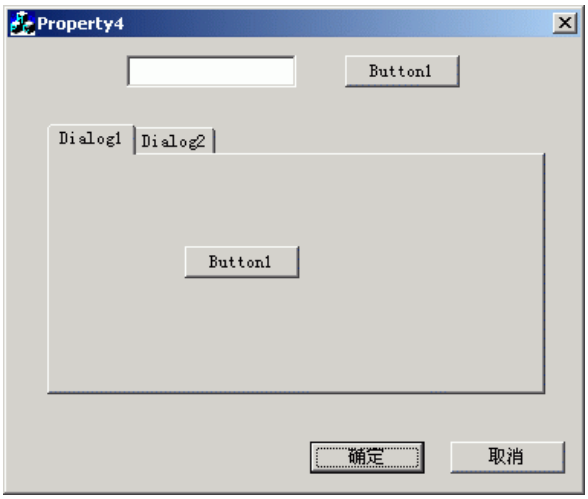
在主对话框类的 OnInitDialog()函数中加入：

```
//加入标签，标签名由各个子对话框的标题栏决定
m_sheet.AddPage(&m_page1);
m_sheet.AddPage(&m_page2);
//用 Create 来创建一个属性页
m_sheet.Create(this, WS_CHILD | WS_VISIBLE, WS_EX_CONTROLPARENT);

RECT rect;
m_sheet.GetWindowRect(&rect);
int width = rect.right - rect.left;
int height = rect.bottom - rect.top;

//调整属性页的大小和位置
m_sheet.SetWindowPos(NULL, 20, 50, 0, 0, SWP_NOSIZE | SWP_NOZORDER | SWP_NOACTIVATE);
```

效果如下：



这个方案可以自由在主对话框中加一些必要的控件，而且属性页中的控件也都分散在了各个子对话框类中，使用非常方便。

但是这样也有一些缺陷：主对话框不能处理属性页上标签的消息，即点击标签时无法通知主对话框。（可能笔者水平有限，理论上应该可以，但笔者尚未解决这个问题）

方案五

这次我们仍然要使用 Tab Control，并且从 CTabCtrl 控件类继承自己的类（CTabSheet）来处理。（此方法来自 CodeGuru 的一篇文章，本人稍做修改使其使用更简便）

首先我先介绍一下如何使用 CTabSheet。

先要制作子对话框类，这次的子对话框类不要从 CPropertyPage 继承，而是直接从 CDialog 继承。并且各个子对话框资源的属性应设置为：Style 为 Child， Border 为 None。

在主对话框资源中，加入一个 Tab Control，并且适当调整位置和大小。利用 ClassWizard 来为这个 Tab Control 创建一个 CTabSheet 的控件变量。

在主对话框的 OnInitDialog()加入：

```
m_sheet.AddPage("tab1", &m_page1, IDD_DIALOG1);
m_sheet.AddPage("tab2", &m_page2, IDD_DIALOG2);
m_sheet.Show();
```

就这样就可以在对话框上制作出一个完美的属性页了。效果和上图完全一样。

下面我就来讲讲 CTabSheet 类的细节内容。

CTabSheet 是从 CTabCtrl 继承来的，用于 Tab Control 的控件类。在类中有一个成员变量用来记录各子对话框的指针 CDialog* m_pPages[MAXPAGE]; MAXPAGE 是该类所能加载的标签的最大值。

类中有一个 AddPage 方法，用于记录子对话框的指针和所使用对话框资源的 ID 号。

```
BOOL CTabSheet::AddPage(LPCTSTR title, CDialog *pDialog,UINT ID)
{
if( MAXPAGE == m_nNumOfPages )
return FALSE;
//保存目前总的子对话框数
m_nNumOfPages++;
//记录子对话框的指针、资源 ID、要在标签上显示的文字
```

```
m_pPages[m_nNumOfPages-1] = pDialog;
m_IDD[m_nNumOfPages-1] = ID;
m_Title[m_nNumOfPages-1] = title;
return TRUE;
}
```

在使用 AddPage 加入了若干子对话框后，必须调用 CTabSheet 的 Show 方法来真正生成标签和子对话框。

```
void CTabSheet::Show()
{
//利用 CDialog::Create 来创建子对话框，并且使用 CTabCtrl::InsertItem 来加上相应的标签
for( int i=0; i < m_nNumOfPages; i++ )
{
m_pPages[i]->Create( m_IDD[i], this );
InsertItem( i, m_Title[i] );
}

//由于对话框显示时默认的是第一个标签被选中，所以应该让第一个子对话框显示，其他子对话框隐藏
m_pPages[0]->ShowWindow(SW_SHOW);
for( i=1; i < m_nNumOfPages; i++)
m_pPages[i]->ShowWindow(SW_HIDE);

SetRect();

}
```

生成好标签和子对话框后，调用 CTabSheet::SetRect 来计算并调整属性页的大小。

```
void CTabSheet::SetRect()
{
CRect tabRect, itemRect;
int nX, nY, nXc, nYc;

//得到 Tab Control 的大小
GetClientRect(&tabRect);
GetItemRect(0, &itemRect);

//计算出各子对话框的相对于 Tab Control 的位置和大小
nX=itemRect.left;
nY=itemRect.bottom+1;
nXc=tabRect.right-itemRect.left-2;
nYc=tabRect.bottom-nY-2;

//利用计算出的数据对各子对话框进行调整
m_pPages[0]->SetWindowPos(&wndTop, nX, nY, nXc, nYc, SWP_SHOWWINDOW);
for( int nCount=1; nCount < m_nNumOfPages; nCount++ )
m_pPages[nCount]->SetWindowPos(&wndTop, nX, nY, nXc, nYc, SWP_HIDEWINDOW);

}
```

在单击标签栏后，应该是相应的子对话框显示，正在显示的子对话框应该隐藏。因此利用 ClassWizard 来处理 WM_LBUTTONDOWN 消息。

```
void CTabSheet::OnLButtonDown(UINT nFlags, CPoint point)
{
CTabCtrl::OnLButtonDown(nFlags, point);

//判断是否单击了其他标签
if(m_nCurrentPage != GetCurFocus())
{
//将原先的子对话框隐藏
m_pPages[m_nCurrentPage]->ShowWindow(SW_HIDE);
m_nCurrentPage=GetCurFocus();
//显示当前标签所对应的子对话框
m_pPages[m_nCurrentPage]->ShowWindow(SW_SHOW);
}

}
```

这样利用 CTabSheet 这个类就可以轻松地在对话框上放置自己的属性页了，并且控件都分散在各子对话框类中，符合对象封装的思想。而且用这个方法来制作属性页就可以利用 ClassWizard 来轻松地生成消息映射处理 Tab Control 的消息了。例如：可以处理 TCN_SELCHANGE 消息来对切换了标签时进行一些动作。