Adding a lookup control to the dictionary: Just getting it on the screen

devblogs.microsoft.com/oldnewthing/20050811-08

August 11, 2005

Raymond Chen

When we last left the dictionary project, we were able to display the dictionary entries but hadn't yet gotten around to searching it. Today, we'll place the lookup control, though we won't hook it up until next time.

First, we give the edit control an ID and create some member variables to keep track of it.

```
class RootWindow : public Window
{
 …
 enum {
  IDC_LIST = 1,
  IDC_EDIT = 2,
 };
 …
private:
 HWND m_hwndLV;
 HWND m_hwndEdit;
 int  m_cyEdit;
 COLORREF m_clrTextNormal;
 Dictionary m_dict;
};
```

Of course, we need to create the edit control, too.

```
LRESULT RootWindow::OnCreate()
{
 …
 ListView_SetItemCount(m_hwndLV, m_dict.Length());


 m_hwndEdit = CreateWindow(TEXT(“edit”), NULL,
                  WS_VISIBLE | WS_CHILD | WS_TABSTOP |
                  ES_LEFT | ES_AUTOHSCROLL,
                  0, 0, 0, 0,
                  m_hwnd,
                  (HMENU)IDC_EDIT,
                  g_hinst,
                  NULL);
 if (!m_hwndEdit) return -1;


 HFONT hfLV = GetWindowFont(m_hwndLV);
 SetWindowFont(m_hwndEdit, hfLV, FALSE);


 m_cyEdit = 0;
 HDC hdc = GetDC(m_hwndEdit);
 if (hdc) {
  HFONT hfPrev = SelectFont(hdc, hfLV);
  if (hfPrev) {
   SIZE siz = { 0, 0 };
   if (GetTextExtentPoint32(hdc, TEXT(“0”), 1, &siz)) {
     RECT rc = { 0, 0, siz.cx, siz.cy };
     AdjustWindowRectEx(&rc, GetWindowStyle(m_hwndEdit), FALSE,
                             GetWindowExStyle(m_hwndEdit));
     m_cyEdit = rc.bottom – rc.top;
   }
   SelectFont(hdc, hfPrev);
  }
  ReleaseDC( m_hwndEdit, hdc);
 }
 if (!m_cyEdit) return -1;


 return 0;
}
```

After creating it, we give it the same font that the listview is using, so that they match. We then measure that font to figure out how big the edit control needs to be in order to accomodate the text.

We use this size information to guide how we lay out our window.

```
LRESULT RootWindow::HandleMessage(
                          UINT uMsg, WPARAM wParam, LPARAM lParam)
{
 …
  case WM_SIZE:
   if (m_hwndEdit) {
    SetWindowPos(m_hwndEdit, NULL, 0, 0,
                 GET_X_LPARAM(lParam), m_cyEdit,
                 SWP_NOZORDER | SWP_NOACTIVATE);
   }
   if (m_hwndLV) {
    SetWindowPos(m_hwndLV, NULL, 0, m_cyEdit,
                 GET_X_LPARAM(lParam),
                 GET_Y_LPARAM(lParam) – m_cyEdit,
                 SWP_NOZORDER | SWP_NOACTIVATE);
   }
   return 0;
 …
}
```

The edit control goes at the top of our client area, and the listview goes directly below it.

Finally, we add a call to the `IsDialogMessage` function to our message loop, making the dialog manager do the heavy lifting of navigating around our window via the Tab and Shift+Tab keys.

```
  RootWindow *prw = RootWindow::Create();
  if (prw) {
   ShowWindow(prw->GetHWND(), nShowCmd);
   MSG msg;
   while (GetMessage(&msg, NULL, 0, 0)) {
    if (IsDialogMessage(prw->GetHWND(), &msg)) {
     /* processed */
    } else {
     TranslateMessage(&msg);
     DispatchMessage(&msg);
    }
   }
  }
```

When you run this program, observe that the edit control and listview position themselves correctly as you resize the window, and that you can Tab between them. But there's still something wrong: Focus always returns to the listview when you switch away and back. That's because I missed a spot.

```
class RootWindow : public Window
{
private:
 HWND m_hwndLV;
 HWND m_hwndEdit;
 HWND m_hwndLastFocus;
 int  m_cyEdit;
 …
};


LRESULT RootWindow::OnCreate()
{
 …
 if (!m_cyEdit) return -1;


 m_hwndLastFocus = m_hwndEdit;


 return 0;
}


LRESULT RootWindow::HandleMessage(
                          UINT uMsg, WPARAM wParam, LPARAM lParam)
{
 …
  case WM_SETFOCUS:
   if (m_hwndLastFocus) {
    SetFocus(m_hwndLastFocus);
   }
   return 0;


  case WM_ACTIVATE:
   if (wParam == WA_INACTIVE) {
    m_hwndLastFocus = GetFocus();
   }
   break;
 …
}
```

The new member variable keeps track of which control had focus last. We update it when we lose activation and restore it when we regain focus. (Its initial value is set at creation so we know whom to give focus to when the window is shown for the first time.)

Okay, that was an awful lot of typing without very much payoff. Next time, we'll start searching the dictionary.

Raymond Chen

Follow