How do I get the tabbed dialog effect on my own custom tabbed dialog?

devblogs.microsoft.com/oldnewthing/20121105-00

November 5, 2012

Raymond Chen

CJ observed that the standard tabbed dialogs provide an effect on the tab pages and want to know how to get the tabbed dialog effect on custom dialogs. fds242 noted this as well and wanted to know why the automatic tabbed dialog effect doesn't kick in until you put a static control on the child dialog box.

Let's look at the first question first. To get the tabbed dialog effect, you can call `EnableThemeDialogTexture` with the `ETDT_ENABLETAB` flag. The best time to do this is in the `WM_INITDIALOG` handler, but you can also do it immediately after the dialog has been created. (Basically, you want to do this before the dialog paints for the first time, so as to avoid flicker.)

Here's a sample program that shows a dummy dialog with the tabbed dialog texture enabled.

```
// Hereby incorporated by reference:
// dialog template and DlgProc function from this sample program
// Comctl32 version 6 manifest from this sample program
int WINAPI WinMain(HINSTANCE hinst, HINSTANCE hinstPrev,
                   LPSTR lpCmdLine, int nShowCmd)
{
    return DialogBox(hinst, MAKEINTRESOURCE(1), 0, DlgProc);
}
```

If you run this program, you get the expected dialog box without the tabbed dialog effect. But you can turn on the effect by calling the `EnableThemeDialogTexture` function:

```
#include <uxtheme.h>
 case WM_INITDIALOG:
  CheckRadioButton(hdlg, 100, 102, 100);
  EnableThemeDialogTexture(hdlg, ETDT_ENABLETAB);
  return TRUE;
```

Now, when you run the program, you get the tabbed dialog effect. It looks kind of weird when it's not in a tabbed dialog, but presumably you're going to put this dialog inside your own custom tabbed dialog control, so everything will look right when it's all finished.

Now the second half of the question: Why doesn't the automatic tabbed dialog effect kick in until you put a static control on the child dialog box?

If you look closely at the `ETDT_ENABLETAB` flag, you'll see that it's really two flags: `ETDT_USETABTEXTURE` and `ETDT_ENABLE`. The first flag says, "I would like to get the tab texture, if enabled"; the second flag says "Enable it." In other words, in order to get the tab texture, the tab texture needs to be both *used* and *enabled*.

Originally, `ETDT_ENABLETAB` was just a single bit. Setting the bit turned on the tab texture. But it turns out that some programs didn't look good with the tab texture, and the common reason was that they created a dialog with no standard controls at all and then did custom drawing all over it. Therefore, the algorithm for enabling the tab texture was changed to the two-step version. The property sheet manager turned on the `ETDT_USETABTEXTURE` flag, and the button and static controls turned on the `ETDT_ENABLE` flag. Therefore, if your property sheet page has a button or a static, the second bit got turned on, and the tab texture became visible. On the other hand, if you didn't have any buttons or statics, then the assumption is that you're one of those programs that does custom dialog drawing, and the tab texture remains disabled.

Let's watch it in action:

```
1 DIALOGEX 32, 32, 200, 76
STYLE DS_MODALFRAME
CAPTION "Sample"
FONT 8, "MS Shell Dlg"
BEGIN
    // nothing!
END
INT_PTR CALLBACK DlgProc(HWND hdlg, UINT uMsg,
                         WPARAM wParam, LPARAM lParam)
{
 switch (uMsg) {
 case WM_INITDIALOG:
  return TRUE;
 }
 return FALSE;
}
int WINAPI WinMain(HINSTANCE hinst, HINSTANCE hinstPrev,
                   LPSTR lpCmdLine, int nShowCmd)
{
 PROPSHEETPAGE psp = { sizeof(psp) };
 psp.hInstance = hinst;
 psp.pszTemplate = MAKEINTRESOURCE(1);
 psp.pfnDlgProc = DlgProc2;
 PROPSHEETHEADER psh = { sizeof(psh) };
 psh.dwFlags = PSH_PROPSHEETPAGE;
 psh.nPages = 1;
 psh.ppsp = &psp;
 return PropertySheet(&psh);
}
```

If you run this program, you'll see that there is no tabbed dialog texture. As we saw earlier, the reason there is no tabbed dialog texture is that the system is afraid that you're one of those programs that custom-draws their dialog boxes, so it's staying out of your way.

But add this line:

```
 case WM_INITDIALOG:
  EnableThemeDialogTexture(hdlg, ETDT_ENABLETAB);
  return TRUE;
```

The property sheet manager was afraid to give you that texture by default, but adding that line just adds the texture manually.

This time, when you run the program, you get the happy tabbed dialog texture because you added it explicitly.

I will leave you to answer fds242's final question: "Why do Windows Task Manager's tab pages still have the gray background."

Raymond Chen

Follow