What if my application is really two applications bundled into a single file, and I want them collected into two groups on the taskbar in Windows 7?

devblogs.microsoft.com/oldnewthing/20120817-00

August 17, 2012

Raymond Chen

A customer wanted to prevent multiple copies of their program from being grouped on the taskbar. They didn't give an explanation why, but let's assume that they are doing this for honorable purposes rather than as a way to annoy the user. For example, maybe their program is really multiple applications bundled inside a single EXE file for convenience.

The information you need to do this is in MSDN under Application User Model IDs, specifically in the *Where to assign an AppUserModelID* section. I'll assume you've read the guidance there, and I'm just going to dive into the implementation.

Suppose our scratch program can serve both as a floor wax and as a dessert topping. It decides on the mode based on a command line switch.

```
#include <shlobj.h>
int WINAPI WinMain(HINSTANCE hinst, HINSTANCE hinstPrev,
                   LPSTR lpCmdLine, int nShowCmd)
{
    MSG msg;
    HWND hwnd;
    g_hinst = hinst;
    if (!InitApp()) return 0;
    BOOL fDessert = strcmp(lpCmdLine, "-dessert") == 0;
    SetCurrentProcessExplicitAppUserModelID(fDessert ?
            L"Contoso.LitWare.DessertTopping" :
            L"Contoso.LitWare.FloorWax");
    if (SUCCEEDED(CoInitialize(NULL))) {/* In case we use COM */
        hwnd = CreateWindow(
            TEXT("Scratch"),                /* Class Name */
            fDessert ? TEXT("Dessert topping") : TEXT("Floor wax"),
            WS_OVERLAPPEDWINDOW,            /* Style */
            CW_USEDEFAULT, CW_USEDEFAULT,   /* Position */
            CW_USEDEFAULT, CW_USEDEFAULT,   /* Size */
            NULL,                           /* Parent */
            NULL,                           /* No menu */
            hinst,                          /* Instance */
            0);                             /* No special parameters */
    ...
}
```

Run this program a few times, some with the `-dessert` switch and some without. Observe that the dessert versions and non-dessert versions group separately.

The next level of fancy-pants behavior is to give different AppIDs to different windows within a single process. You might do this if your combination floor wax/dessert topping program actually runs both modes inside the same process. Something like this:

```
#include <shellapi.h>
#include <propkey.h>
#include <propvarutil.h>
#include <shlobj.h>
int g_cWindows = 0;
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
  ++g_cWindows;
  return TRUE;
}
void
OnDestroy(HWND hwnd)
{
  if (--g_cWindows == 0) PostQuitMessage(0);
}
HWND
CreateTaskWindow(BOOL fDessert, int nShowCmd)
{
  HWND hwnd = CreateWindow(
      TEXT("Scratch"),                /* Class Name */
      fDessert ? TEXT("Dessert topping") : TEXT("Floor wax"),
      WS_OVERLAPPEDWINDOW,            /* Style */
      CW_USEDEFAULT, CW_USEDEFAULT,   /* Position */
      CW_USEDEFAULT, CW_USEDEFAULT,   /* Size */
      NULL,                           /* Parent */
      NULL,                           /* No menu */
      g_hinst,                        /* Instance */
      0);                             /* No special parameters */
  if (hwnd) {
    IPropertyStore *pps;
    HRESULT hr = SHGetPropertyStoreForWindow(hwnd, IID_PPV_ARGS(&pps));
    if (SUCCEEDED(hr)) {
      IPropertyStore_SetValue(pps, PKEY_AppUserModel_ID,
            fDessert ?
            L"Contoso.LitWare.DessertTopping" :
            L"Contoso.LitWare.FloorWax");
      pps->Release();
    }
    ShowWindow(hwnd, nShowCmd);
  }
  return hwnd;
}
void
OnChar(HWND hwnd, TCHAR ch, int cRepeat)
{
    switch (ch) {
    case 'd': CreateTaskWindow(TRUE, SW_SHOWNORMAL); break;
    case 'f': CreateTaskWindow(FALSE, SW_SHOWNORMAL); break;
    }
}
    HANDLE_MSG(hwnd, WM_CHAR, OnChar);
```

```
int WINAPI WinMain(HINSTANCE hinst, HINSTANCE hinstPrev,
                   LPSTR lpCmdLine, int nShowCmd)
{
    MSG msg;
    HWND hwnd;
    g_hinst = hinst;
    if (!InitApp()) return 0;
    BOOL fDessert = strcmp(lpCmdLine, "-dessert") == 0;
    // SetCurrentProcessExplicitAppUserModelID(...);
    if (SUCCEEDED(CoInitialize(NULL))) {/* In case we use COM */
        hwnd = CreateTaskWindow(fDessert, nShowCmd);
        // ShowWindow(hwnd, nShowCmd);
    ...
}
```

This time, instead of setting the application ID globally, we set it on a per-window basis. When you run this program, you can press "f" to open a new floor wax window or "d" to open a new dessert topping window. As before, observe that the two types of windows group separately.

The last detail is setting the `System.AppUserModel.ID` property on the shortcuts used to launch these programs. You can do this from MSI by adding an entry to your `MsiShortcut-Property` table, or if you create your shortcuts programmatically, you do this by setting the property yourself:

```
CComPtr<IShellLink> spsl;
spsl.CoCreateInstance(CLSID_ShellLink);
spsl->SetPath(TEXT("C:\\Path\\to\\scratch.exe"));
CComQIPtr<IPropertyStore> spps(spsl);
IPropertyStore_SetValue(spps, PKEY_AppUserModel_ID,
                        L"Contoso.LitWare.FloorWax");
spps->Commit();
CComQIPtr<IPersistFile>(spsl)->Save(L"LitWare Floor Wax.lnk", TRUE);
```

Next time, we'll look at another reason you might want to customize how your application group on the taskbar in Windows 7.

Raymond Chen

Follow