Querying information from an Explorer window

devblogs.microsoft.com/oldnewthing/20040720-00

July 20, 2004

Raymond Chen

Sometimes software development is inventing new stuff. But often, it's just putting together the stuff you already have. Today's puzzle is one of the latter type of problem.

Given a window handle, you can you determine (1) whether it is an Explorer window, and if so (2) what folder it is viewing, and (3) what item is currently focused.

This is not an inherently difficult task. You just have to put together lots of small pieces.

Start with the ShellWindows object which represents all the open shell windows. You can enumerate through them all with the Item property. This is rather clumsy from C++ because the ShellWindows object was designed for use by a scripting language like JScript or Visual Basic.

```
IShellWindows *psw;
if (SUCCEEDED(CoCreateInstance(CLSID_ShellWindows, NULL, CLSCTX_ALL,
                               IID_IShellWindows, (void**)&psw))) {
 VARIANT v;
 V_VT(&v) = VT_I4;
 IDispatch  *pdisp;
 BOOL fFound = FALSE;
 for (V_I4(&v) = 0; !fFound && psw->Item(v, &pdisp) == S_OK;
      V_I4(&v)++) {
   ...
   pdisp->Release();
 }
 psw->Release();
}
```

From each item, we can ask it for its window handle and see if it's the one we want.

```
IWebBrowserApp *pwba;
if (SUCCEEDED(pdisp->QueryInterface(IID_IWebBrowserApp, (void**)&pwba))) {
  HWND hwndWBA;
  if (SUCCEEDED(pwba->get_HWND((LONG_PTR*)&hwndWBA)) &&
    hwndWBA == hwndFind) {
    fFound = TRUE;
    ...
  }
  pwba->Release();
}
```

Okay, now that we have found the folder via its IWebBrowserApp, we need to get to the top shell browser. This is done by querying for the SID_STopLevelBrowser service and asking for the IShellBrowser interface.

```
IServiceProvider *psp;
if (SUCCEEDED(pwba->QueryInterface(IID_IServiceProvider, (void**)&psp))) {
  IShellBrowser *psb;
  if (SUCCEEDED(psp->QueryService(SID_STopLevelBrowser,
                      IID_IShellBrowser, (void**)&psb))) {
    ...
    psb->Release();
  }
  psp->Release();
}
```

From the IShellBrowser, we can ask for the current shell view via the QueryActiveShellView method.

```
IShellView *psv;
if (SUCCEEDED(psb->QueryActiveShellView(&psv))) {
  ...
  psv->Release();
}
```

Of course, what we really want is the IFolderView interface, which is the automation object that contains all the real goodies.

```
IFolderView *pfv;
if (SUCCEEDED(psv->QueryInterface(IID_IFolderView,
                                  (void**)&pfv))) {
  ...
  pfv->Release();
}
```

Okay, now we're golden. What do you want to get from the view? How about the location of the IShellFolder being viewed. To do that, we need to use IPersistFolder2::GetCurFolder. The GetFolder method will give us access to the shell folder, from which we ask for IPersistFolder2. (Most of the time you want the IShellFolder interface, since that's where most of the cool stuff hangs out.)

```
IPersistFolder2 *ppf2;
if (SUCCEEDED(pfv->GetFolder(IID_IPersistFolder2,
                             (void**)&ppf2))) {
  LPITEMIDLIST pidlFolder;
  if (SUCCEEDED(ppf2->GetCurFolder(&pidlFolder))) {
    ...
    CoTaskMemFree(pidlFolder);
  }
  ppf2->Release();
}
```

Let's convert that `pidl` into a path, for display purposes.

```
  if (!SHGetPathFromIDList(pidlFolder, g_szPath)) {
    lstrcpyn(g_szPath, TEXT("<not a directory>"), MAX_PATH);
  }
  ...
```

What else can we do with what we've got? Oh right, let's see what the currently-focused object is.

```
  int iFocus;
  if (SUCCEEDED(pfv->GetFocusedItem(&iFocus))) {
    ...
  }
```

Let's display the name of the focused item. To do that we need the item's pidl and the IShellFolder. (See, I told you the IShellFolder is where the cool stuff is.) The item comes from the Item method (surprisingly enough).

```
  LPITEMIDLIST pidlItem;
  if (SUCCEEDED(pfv->Item(iFocus, &pidlItem))) {
    ...
    CoTaskMemFree(pidlItem);
  }
```

(If we had wanted a list of selected items we could have used the Items method, passing SVGIO_SELECTION.)

After we get the item's pidl, we also need the IShellFolder:

```
  IShellFolder *psf;
  if (SUCCEEDED(ppf2->QueryInterface(IID_IShellFolder,
                                     (void**)&psf))) {
    ...
    psf->Release();
  }
```

Then we put the two together to get the item's display name, with the help of the GetDisplayNameOf method.

```
STRRET str;
if (SUCCEEDED(psf->GetDisplayNameOf(pidlItem,
                            SHGDN_INFOLDER,
                            &str))) {
  ...
}
```

We can use the helper function StrRetToBuf to convert the kooky STRRET structure into a boring string buffer. (The history of the kooky STRRET structure will have to wait for another day.)

```
  StrRetToBuf(&str, pidlItem, g_szItem, MAX_PATH);
```

Okay, let's put this all together. It looks rather ugly because I put everything into one huge function instead of breaking them out into subfunctions. In "real life" I would have broken things up into little helper functions to make things more manageable.

Start with the scratch program and add this new function:

```
#include <shlobj.h>
#include <exdisp.h>
TCHAR g_szPath[MAX_PATH];
TCHAR g_szItem[MAX_PATH];
void CALLBACK RecalcText(HWND hwnd, UINT, UINT_PTR, DWORD)
{
 HWND hwndFind = GetForegroundWindow();
 g_szPath[0] = TEXT('\0');
 g_szItem[0] = TEXT('\0');
 IShellWindows *psw;
 if (SUCCEEDED(CoCreateInstance(CLSID_ShellWindows, NULL, CLSCTX_ALL,
                                IID_IShellWindows, (void**)&psw))) {
  VARIANT v;
  V_VT(&v) = VT_I4;
  IDispatch  *pdisp;
  BOOL fFound = FALSE;
  for (V_I4(&v) = 0; !fFound && psw->Item(v, &pdisp) == S_OK;
       V_I4(&v)++) {
   IWebBrowserApp *pwba;
   if (SUCCEEDED(pdisp->QueryInterface(IID_IWebBrowserApp, (void**)&pwba))) {
     HWND hwndWBA;
     if (SUCCEEDED(pwba->get_HWND((LONG_PTR*)&hwndWBA)) &&
       hwndWBA == hwndFind) {
       fFound = TRUE;
       IServiceProvider *psp;
       if (SUCCEEDED(pwba->QueryInterface(IID_IServiceProvider, (void**)&psp))) {
         IShellBrowser *psb;
         if (SUCCEEDED(psp->QueryService(SID_STopLevelBrowser,
                                IID_IShellBrowser, (void**)&psb))) {
           IShellView *psv;
           if (SUCCEEDED(psb->QueryActiveShellView(&psv))) {
             IFolderView *pfv;
             if (SUCCEEDED(psv->QueryInterface(IID_IFolderView,
                                               (void**)&pfv))) {
               IPersistFolder2 *ppf2;
               if (SUCCEEDED(pfv->GetFolder(IID_IPersistFolder2,
                                            (void**)&ppf2))) {
                 LPITEMIDLIST pidlFolder;
                 if (SUCCEEDED(ppf2->GetCurFolder(&pidlFolder))) {
                   if (!SHGetPathFromIDList(pidlFolder, g_szPath)) {
                     lstrcpyn(g_szPath, TEXT("<not a directory>"), MAX_PATH);
                   }
                   int iFocus;
                   if (SUCCEEDED(pfv->GetFocusedItem(&iFocus))) {
                     LPITEMIDLIST pidlItem;
                     if (SUCCEEDED(pfv->Item(iFocus, &pidlItem))) {
                       IShellFolder *psf;
                       if (SUCCEEDED(ppf2->QueryInterface(IID_IShellFolder,
                                                          (void**)&psf))) {
                         STRRET str;
                         if (SUCCEEDED(psf->GetDisplayNameOf(pidlItem,
                                                             SHGDN_INFOLDER,
```

```
                                        &str))) {
                        StrRetToBuf(&str, pidlItem, g_szItem, MAX_PATH);
                      }
                      psf->Release();
                    }
                    CoTaskMemFree(pidlItem);
                  }
                }
                CoTaskMemFree(pidlFolder);
              }
              ppf2->Release();
            }
            pfv->Release();
          }
          psv->Release();
        }
        psb->Release();
      }
      psp->Release();
    }
   }
   pwba->Release();
  }
   pdisp->Release();
 }
 psw->Release();
}
InvalidateRect(hwnd, NULL, TRUE);
}
```

Now all we have to do is call this function periodically and print the results.

```
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
    SetTimer(hwnd, 1, 1000, RecalcText);
    return TRUE;
}
void
PaintContent(HWND hwnd, PAINTSTRUCT *pps)
{
  TextOut(pps->hdc, 0, 0, g_szPath, lstrlen(g_szPath));
  TextOut(pps->hdc, 0, 20, g_szItem, lstrlen(g_szItem));
}
```

We’re ready to roll. Run this program and set it to the side. Then launch an Explorer window and watch the program track the folder you’re in and what item you have focused.

Okay, so I hope I made my point: Often, the pieces you need are already there; you just have to figure out how to put them together. Notice that each of the pieces is in itself not very big. You just had to recognize that they could be put together in an interesting way.

Exercise: Change this program so it takes the folder and switches it to details view.

[Raymond is currently on vacation; this message was pre-recorded.]

Raymond Chen

Follow