How does a shell namespace extension provide icons for virtual items that track the standard icons set by the user's file associations?

devblogs.microsoft.com/oldnewthing/20151009-00

October 9, 2015

Raymond Chen

A customer asked, "What is the correct way to retrieve the icon associated with a file extension? We are writing a shell namespace extension that holds virtual file content, and we want to show the icon that would have been shown if the file were a physical file on disk rather than a virtual one. We tried using `SHGetFileInfo`, expecting it to return the icon location and index, but the `szDisplayName` comes out as a blank string. (See sample program attached.) What's the right way to get the location so we can return it in our own `GetUIObjectOf(IExtractIcon)` handler?"

```
#include <windows.h>
#include <iostream>

int main()
{
 SHFILEINFOW info;
 ::CoInitializeEx(NULL, COINIT_APARTMENTTHREADED);
 ::SHGetFileInfoW(L".txt", FILE_ATTRIBUTE_NORMAL,
    &info, sizeof(info),
    SHGFI_ICONLOCATION | SHGFI_USEFILEATTRIBUTES);
 std::wcout << info.szDisplayName << std::endl;
 std::wcout << info.iIcon << std::endl;
 return 0;
}
```

The location is coming out blank because the file location returned is `GIL_NOTFILENAME` so there is no file name to return.

But let's look past the question to the problem. The problem is that you want to implement `IShellFolder::GetUIObjectOf(IExtractIcon)` for your shell namespace extension. Your plan is to create a custom implementation of `IExtractIcon` and tell it to report the information you obtained from `SHGetFileInfo`. The catch is that this information is lossy because `IExtractIcon::GetIconLocation` returns additional information that is not captured by `SHGetFileInfo`.

Avoid the loss of fidelity by removing the middle man. Just ask for the standard icon extractor and return *that*.

We start with a helper function that takes its inspiration from `GetUIObjectOfFile` but applies a little seasoning from `CreateSimplePidl`:

```
HRESULT GetUIObjectOfVirtualFile(HWND hwnd, LPCWSTR pszPath,
    REFIID riid, void **ppv)
{
  *ppv = nullptr;

  WIN32_FIND_DATAW fd = {};
  fd.dwFileAttributes = FILE_ATTRIBUTE_NORMAL;
  CComHeapPtr<ITEMIDLIST_ABSOLUTE> spidlSimple;
  HRESULT hr = CreateSimplePidl(&fd, pszPath, &spidlSimple);
  if (FAILED(hr)) return hr;

  CComPtr<IShellFolder> spsf;
  PCUITEMID_CHILD pidlChild;
  hr = SHBindToParent(spidlSimple, IID_PPV_ARGS(&spsf), &pidlChild);
  if (FAILED(hr)) return hr;

  return spsf->GetUIObjectOf(hwnd, 1, &pidlChild, riid, NULL, ppv);
}
```

This helper function is like `GetUIObjectOfFile` except that it uses a simple pidl to get the UI object for a file that doesn't actually exist.

We can use this function to get the icon extractor for an arbitrary file extension.

```
HRESULT GetIconExtractorForExtension(
    HWND hwnd,
    PCWSTR pszExtension,
    REFIID riid,
    void **ppv)
{
 *ppv = nullptr;

 wchar_t szPath[MAX_PATH];
 HRESULT hr = StringCchPrintfW(szPath, ARRAYSIZE(szPath),
                               L"C:\\a%ls", pszExtension);
 if (FAILED(hr)) return hr;

 return GetUIObjectOfVirtualFile(hwnd, szPath, riid, ppv);
}
```

and then use this function when handling the request for `IExtractIcon`.

```
if (interfaceId == IID_IExtractIconW ||
    interfaceId == IID_IExtractIconA)
{
  return GetIconExtractorForExtension(hwnd, L".txt", riid, ppv);
}
```

Raymond Chen

Follow