How do I create an IShellItemArray from a bunch of file paths?

devblogs.microsoft.com/oldnewthing/20140314-00

March 14, 2014

Raymond Chen

The `IFileOperation` interface accepts bulk operations in the form of an `IShellItemArray`. So how do you take a list of file names and convert them into an `IShellItemArray`?

There is no `SHCreateShellItemArrayFromPaths` function, but there is a `SHCreateShellItemArrayFromIDLists`, and we know how to convert a path to an ID list, namely via `SHParseDisplayName`. So lets snap two blocks together.

```
#define UNICODE
#define _UNICODE
#define STRICT
#define STRICT_TYPED_ITEMIDS
#include <windows.h>
#include <shlobj.h>
#include <wrl/client.h>
// class CCoInitialize incorporated by reference
template<typename T>
HRESULT CreateShellItemArrayFromPaths(
    UINT ct, T rgt[], IShellItemArray **ppsia)
{
 *ppsia = nullptr;
 PIDLIST_ABSOLUTE *rgpidl = new(std::nothrow) PIDLIST_ABSOLUTE[ct];
 HRESULT hr = rgpidl ? S_OK : E_OUTOFMEMORY;
 int cpidl;
 for (cpidl = 0; SUCCEEDED(hr) && cpidl < ct; cpidl++)
 {
  hr = SHParseDisplayName(rgt[cpidl], nullptr, &rgpidl[cpidl], 0, nullptr);
 }
 if (SUCCEEDED(hr)) {
  hr = SHCreateShellItemArrayFromIDLists(cpidl, rgpidl, ppsia);
 }
 for (int i = 0; i < cpidl; i++)
 {
  CoTaskMemFree(rgpidl[i]);
 }
 delete[] rgpidl;
 return hr;
}
```

The `CreateShellItemArrayFromPaths` template function takes an array of paths and starts by creating a corresponding array of ID lists. (If you're feeling fancy, you can use a file system bind context to make simple ID lists.) It then pumps this array into the `SHCreate-ShellItemArrayFromIDLists` function to get the item array.

Using a template allows you to pass an array of *anything* as the array of paths, as long as it has a conversion to `PCWSTR`. So you can pass an array of `PCWSTR` or an array of `PWSTR` or an array of `BSTR` or an array of `CComHeapPtr<wchar_t>` or an array of `CStringW` or whatever else floats your boat.

Let's take this function out for a spin.

```
int __cdecl wmain(int argc, wchar_t **argv)
{
 CCoInitialize init;
 Microsoft::WRL::ComPtr<IShellItemArray> spsia;
 Microsoft::WRL::ComPtr<IFileOperation> spfo;
 if (SUCCEEDED(CreateShellItemArrayFromPaths(
                     argc - 1, argv + 1, &spsia)) &&
     SUCCEEDED(CoCreateInstance(__uuidof(FileOperation), nullptr,
                     CLSCTX_ALL, IID_PPV_ARGS(&spfo)))) {
  spfo->DeleteItems(spsia.Get());
  spfo->PerformOperations();
 }
 return 0;
}
```

The main program first treats the command line arguments as a list of absolute file paths and uses our new helper function to create a shell item array from them. It then passes the shell item array to the `IFileOperation::DeleteItems` method to delete all the items.

No magic here. Just taking the pieces available and combining them in a relatively obvious way.

Raymond Chen

Follow