Cancelling the INamespaceWalk::Walk operation a little faster

devblogs.microsoft.com/oldnewthing/20171110-00

November 10, 2017

Raymond Chen

We saw last time that you can stop a `INamespaceWalk::Walk` operation by returning a COM error code from the `EnterFolder` or `FoundItem` callback. However, that may not be fast enough.

I noted some time ago that if you're going to enumerate the contents of a directory, you'd best do it all at once. And that's what `INamespaceWalk::Walk` does. After it enters a directory, it enumerates the whole thing at one shot, and then (optionally) sorts it, and then calls the `FoundItem` method for each item that was found.

If you happen to enter a large directory, then the "enumerate the whole thing at one shot" step can take a while. But there's a way to sneak in during the enumeration phase and cancel the operation: Implement the `IActionProgress` interface on your `INamespaceWalkCB` object. Note that this works only if you do *not* pass the `NSWF_SHOW_PROGRESS` flag. If you pass the `NSWF_SHOW_PROGRESS` flag, then the progress dialog's Cancel button controls the cancellation.

Assuming you don't pass the `NSWF_SHOW_PROGRESS` flag, the `INamespaceWalk::Walk` method will call `IActionProgress::Begin` to get the party started, and `IActionProgress::End` when it's all over. In between, it will call `IActionProgress::QueryCancel`. If your `IActionProgress::QueryCancel` method returns `*pfCancelled = TRUE`, then the `INamespaceWalk::Walk` operation will abandon the enumeration, unwind all the entered folders with `LeaveFolder`, and then return `HRESULT_FROM_WIN32(ERROR_CANCELLED)`.

Let's use this technique to cancel the `INamespaceWalk::Walk` operation a bit more quickly. Make the following changes to the program we had last time:

```
#define STRICT
#include <windows.h>
#include <shlobj.h>
#include <wrl/client.h>
#include <wrl/implements.h>
#include <stdio.h> // Horrors! Mixing stdio and C++!

namespace wrl = Microsoft::WRL;

class WalkCallback : public wrl::RuntimeClass<
  wrl::RuntimeClassFlags<wrl::ClassicCom>,
  INamespaceWalkCB,
  IActionProgress> // New interface!
{
public:
  // INamespaceWalkCB
  IFACEMETHODIMP FoundItem(IShellFolder *,
   PCUITEMID_CHILD) override
   { m_itemCount++; return TimeoutStatus(); }

  IFACEMETHODIMP EnterFolder(IShellFolder *,
   PCUITEMID_CHILD) override
   { m_folderCount++; return TimeoutStatus(); }

  IFACEMETHODIMP LeaveFolder(IShellFolder *,
   PCUITEMID_CHILD) override { return S_OK; }

  IFACEMETHODIMP InitializeProgressDialog(PWSTR *ppszTitle,
    PWSTR *ppszCancel) override
    { *ppszTitle = nullptr; *ppszCancel = nullptr;
      return E_NOTIMPL; }

  // IActionProgress - new interface!
  IFACEMETHODIMP Begin(SPACTION, SPBEGINF) override
  { return S_OK; }

  IFACEMETHODIMP UpdateProgress(ULONGLONG, ULONGLONG) override
  { return S_OK; }

  IFACEMETHODIMP UpdateText(SPTEXT, LPCWSTR, BOOL) override
  { return S_OK; }

  IFACEMETHODIMP QueryCancel(BOOL *pfCancelled) override
  { *pfCancelled = IsTimedOut(); return S_OK; }

  IFACEMETHODIMP ResetCancel() override { return S_OK; }
  IFACEMETHODIMP End() override { return S_OK; }

  int ItemCount() const { return m_itemCount; }
  int FolderCount() const { return m_folderCount; }

private:
```

```
  bool IsTimedOut()
    { return GetTickCount() - m_startTime > 1000; }

  HRESULT TimeoutStatus()
    { return IsTimedOut() ?
      HRESULT_FROM_WIN32(ERROR_CANCELLED) : S_OK; }

  DWORD m_startTime = GetTickCount();
  int m_itemCount = 0;
  int m_folderCount = 0;
};

int __cdecl wmain(int argc, PWSTR argv[])
{
  CCoInitialize coinit;

  wrl::ComPtr<INamespaceWalk> walk;
  CoCreateInstance(CLSID_NamespaceWalker, nullptr,
    CLSCTX_INPROC_SERVER, IID_PPV_ARGS(&walk));

  wrl::ComPtr<IShellItem> root;
  SHCreateItemFromParsingName(argv[1], nullptr,
    IID_PPV_ARGS(&root));

  auto callback = wrl::Make<WalkCallback>();

  HRESULT hr = walk->Walk(root.Get(), NSWF_DEFAULT,
    100, callback.Get());

  printf("Walk completed with result 0x%08x\n", hr);
  printf("Found %d items and %d folders\n",
   callback->ItemCount(), callback->FolderCount());

  return 0;
}
```

All we did was add `IActionProgress` support to our callback object. When asked if we want to cancel the operation, we report whether the operation has timed out.

Adding this extra support will not be noticeable when enumerating relatively small directories from relatively fast media.

Raymond Chen

Follow