How can a desktop app use a Windows Runtime object that infers UI context from its thread? The IInitializeWithWindow pattern

devblogs.microsoft.com/oldnewthing/20190412-00

April 12, 2019

Raymond Chen

Many objects in the Windows Runtime can be used from desktop apps. For today's example, we'll use the `FileOpenPicker`. This is a rather artificial example because you could just use the `IFileDialog` interface to get equivalent functionality in a desktop app, but I just picked it for use as an example.

Start with our scratch program and make these changes:

```
#include <winrt/windows.storage.pickers.h>

winrt::Windows::Foundation::IAsyncAction
ShowFilePickerAsync(HWND hwnd)
{
    auto picker = winrt::Windows::Storage::Pickers::FileOpenPicker();
    picker.FileTypeFilter().Append(L".jpg");
    auto file = co_await picker.PickSingleFileAsync();
}

winrt::fire_and_forget OnChar(HWND hwnd, TCHAR ch, int cRepeat)
{
    co_await ShowFilePickerAsync(hwnd);
}

// Add to WndProc
    HANDLE_MSG(hwnd, WM_CHAR, OnChar);
```

Run this program and press a key. The program will crash because the `FileOpenPicker` looks for a `CoreWindow` on the current thread to serve as the owner of the dialog. But we are a Win32 desktop app without a `CoreWindow`.

The solution is to use the `IInitializeWithWindow` interface. Many Windows Runtime objects which infer the `CoreWindow` from the current thread support the `IInitialize-WithWindow` interface to allow a Win32 desktop app to specify an explicit window.

Make the following changes to the program:

```
#include <shobjidl.h>
#include <winrt/windows.storage.pickers.h>

winrt::Windows::Foundation::IAsyncAction
ShowFilePickerAsync(HWND hwnd)
{
    auto picker = winrt::Windows::Storage::Pickers::FileOpenPicker();
    picker.as<IInitializeWithWindow>()->Initialize(hwnd);
    picker.FileTypeFilter().Append(L".jpg");
    auto file = co_await picker.PickSingleFileAsync();
}
```

This time, the File Open dialog opens because we explicitly provided a window handle to use as the owner.

The `IInitializeWindowWindow` pattern is used mostly in the case where an object is simply constructed. There is another pattern for the case where an object is obtained by calling a method: The interop pattern, which I covered some time ago.

Raymond Chen

Follow