The SetClientGuid method of the common file and folder dialogs lets you give names to those dialogs, too

devblogs.microsoft.com/oldnewthing/20200527-00

May 27, 2020

Raymond Chen

We saw a little while ago that the SettingsIdentifier property of the various file pickers lets you give names to your pickers. Those work for the Windows Runtime pickers, but what if you're using the old-fashioned Win32 pickers?

There is an analogous feature in the Win32 file and folder pickers: You can call the `IFileDialog::SetClientGuid` method with a GUID that names your dialog.[1] As before, you can use different GUIDs for different scenarios in your program, and each of those scenarios will retain their settings separately.

The `IFileDialog` is a base interface of both the `IFileOpenDialog` and `IFileSaveDialog` interfaces, so you get this functionality in file open and save dialogs, as well as the folder picker dialog, which is just a file picker that has been configured with the `FOS_PICKFOLDERS` option.

On the Win32 side, you also have the bonus method `IFileDialog::ClearClientData()` which tells the system to forget the settings that were associated with the client GUID. The next time a dialog with that GUID is displayed, it will show up with the defaults.

[1] Unlike the Windows Runtime case, where names from different apps are kept separate, in the Win32 case, the GUID is kept in a global namespace, so make sure to use some GUID unique to your program, rather than some value that might collide with one chosen by another program.[2]

[2] On the other hand, if you have multiple programs that work together, you might intentionally use the same GUID in all of them, so that they share state. For example, if you have a suite of programs that all have an "Export as Contoso Archive" command, maybe you want them all to export to the same place.

Raymond Chen

Follow