Virtual desktops are an end-user window management feature, not a programmatic one

devblogs.microsoft.com/oldnewthing/20201123-00

November 23, 2020

Raymond Chen

Virtual desktops are a window management feature, not a security feature or a performance feature. Furthermore, they are for *end users* to organize their windows, not for other programs to organize the windows.

The purpose of the `IVirtualDesktopManager` interface is to allow applications to take the virtual desktop state into account in a few application scenarios.

One such scenario is deciding whether to reuse an existing window or to create a new one.

For example, suppose you're a multi-window tabbed application, and the user launches a new document. You have a few options.

- Create a new window with a single tab containing that document.
- Find an existing window and create a new tab within the existing window for the new document.

The recommendation is to stay on the current virtual desktop if possible.

The application can enumerate all of its windows and call `IVirtualDesktop-Manager:: IsWindowOnCurrentVirtualDesktop` for each one, so that it can build a set of target window candidates on the current virtual desktop. It can then choose one of those target window candidates (say, the one most recently active) as the one in which to create a new tab for the document. If there are no candidates on the current virtual desktop, then create a new window with a single tab for the document.

There's no reason for the application to know, say, the names of all of the virtual desktops. The goal is to avoid switching virtual desktops. It's not so an application can say, "Okay, when opening a document, I'll always open it on a virtual desktop named *Planning*." The application has no control over the virtual desktop names anyway. The user picked those. And you can't expect all users to have a virtual desktop called *Planning*."[1]

Another scenario that the `IVirtualDesktopManager` interface is intended to cover is that of an application that consists of multiple windows that operate together, like a main window and a bunch of auxiliary windows. The virtual desktop manager already recognizes many types of auxiliary windows and automatically keeps them on the same virtual desktop as the main window. But an application may have an unusual window structure that eludes this automatic detection.

For those cases, you can call `IVirtualDesktopManager::GetWindowDesktopId` for your main window to get an ID for its current virtual desktop, and then use `IVirtualDesktop-Manager::MoveWindowsToDesktop` to move the auxiliary windows to that same virtual desktop.

The purpose of `MoveWindowsToDesktop` is specifically *not* to let a program grab all of the user's windows and scatter them across all their virtual desktops. Note, for example, that the only thing you can do is make one window join another window's virtual desktop. You can bring windows together, but you can't, say, split a window into a new virtual desktop.

Because the point is to let you move your auxiliary windows so they follow the main window. It is not a general purpose interface for moving windows onto arbitrary virtual desktops. That's for the user to do.

[1] If you want, you can have the user pick a window and say "Use this window for opening new documents, even if it means switching virtual desktops." Then the user is still in control of when virtual desktop switches occur. (And they can move that window to another virtual desktop later.)

Raymond Chen

Follow