Using the DisplayInformation class from a desktop Win32 application, part 1

devblogs.microsoft.com/oldnewthing/20240320-00

March 20, 2024

Raymond Chen

For UWP apps, the Windows Runtime `DisplayInformation` class provides information about the display that the app is running one. In particular, the `Orientation` property tells you whether the screen is running in landscape, portrait, landscape-flipped, or portrait-flipped orientation.

For Win32 desktop apps, you can obtain a `DisplayInformation` object corresponding to a window handle or a display monitor with the assistance of the `IDisplayInformationStatics-Interop` interface. This interface follows the usual pattern for interop interfaces: Query for the interface from the class activation factory and then use the desired `GetFor...` method.

Let's try that in our scratch program.

```
#include <windows.graphics.display.interop.h>
#include <winrt/Windows.Graphics.Display.h>

namespace winrt
{
    using namespace winrt::Windows::Graphics::Display;
}

winrt::DisplayInformation g_info{ nullptr };

BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs) try
{
    winrt::check_hresult(
        winrt::get_activation_factory<winrt::DisplayInformation,
            ::IDisplayInformationStaticsInterop>()->GetForWindow(
        hwnd, winrt::guid_of<decltype(g_info)>(),
        winrt::put_abi(g_info)));
    return TRUE;
}
catch (...)
{
    return FALSE;
}
```

We are following the principle of starting by doing nothing: Our first step is simply to create the `DisplayInformation`. We don't do anything with it.

Our `OnCreate` can be simplified by taking advantage of the `capture_interop` helper.

```
BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs) noexcept
{
    g_info = wil::capture_interop<winrt::DisplayInformation>
        (&IDisplayInformationStaticsInterop::GetForWindow, hwnd);

    return TRUE;
}
catch (...)
{
    return FALSE;
}
```

And when the window is destroyed, we clean up the `DisplayInformation`.

```
void
OnDestroy(HWND hwnd)
{
    g_info = nullptr;
    PostQuitMessage(0);
}
```

If we run this program, we get a "not implemented" exception when we call `GetForWindow()`. Phew, good thing we didn't get in too deep. We added only a few lines of code, so it's easy to narrow in on the lines that are the source of the problem. And if it turns out that the entire undertaking is not implemented, well, we saved ourselves a lot of trouble: It took us only a few lines of code to discover this.

If you watch the debugger, you see a helpful message:

```
onecoreuap\windows\wgi\winrt\display\displayinformation.cpp(101)\Windows.Graphics.dll!
00007FFFAB1BC87B: (caller: 00007FFFAB1BB741) ReturnHr(2) tid(6f04) 80004001 Not implemented
    Msg:[A DispatcherQueue is required for DisplayInformation created from an HWND]
```

Aha, so the problem is that we need the thread to have a dispatcher queue. We'll take up the story next time.