How can you use both versions 5 and 6 of the common controls within the same module?

devblogs.microsoft.com/oldnewthing/20140508-00

May 8, 2014

Raymond Chen

Commenter Medinoc was baffled by the statement that the decision to use the visual-styles-enabled version of the common controls library is done on a window-by-window basis. “ Isn’t it rather on a per-module basis, depending on each module’s manifest? If it is indeed on a per-window basis, how does one choose?”

Whether a particular call to `CreateWindow` (or one of its moral equivalents) gets the classic version of the control or the visual-styles-enabled version of the control depends on which activation context is active at the point of the call. If an activation context with version 6 of the common controls is active, then you get the control from version 6 of the common controls. Otherwise, you get the classic control.

If you use the `ISOLATION_AWARE_ENABLED` macro, then including `commctrl.h` turns on a bunch of macros that take all your calls to `CreateWindow` and related functions, and converts them into something like this:

```
HWND CreateWindow_wrapped(... parameters ...)
{
 HWND hwnd = nullptr;
 ULONG_PTR ulCookie;
 if (ActivateActCtx(ModuleContext, &ulCookie)) {
  hwnd = CreateWindow(... parameters ...);
 DeactivateActCtx(0, ulCookie);
 }
 return hwnd;
}
```

where `ModuleContext` is a global variable that holds the activation context you specified in your manifest.

In other words, any time your code tries to create a window, the wrapper macros activate your v6 manifest, create the window, then deactivate the manifest.

Remember that nobody walks the stack looking to see who the caller is. The return address is not reliable. (And checking the return address doesn't help for dynamically-generated code anyway.) The way to know which activation context is active is for somebody to actually come out and set it.

Back to the question: The way you choose whether you want a classic or a visual-styles-enabled version of a control is by deciding whether or not to have the v6 manifest active when you call `CreateWindow`.

A common mistake is that people will call a function that requires a v6 manifest, such as `TaskDialog`, but they will forget to activate the v6 manifest before calling. The result is that they call into version 6 of the common controls, but when the common controls library tries to create its task dialog, *it fails* because the v5 context is active, and the v5 context does not have a task dialog control.

Raymond Chen

Follow