When should I use CS_GLOBALCLASS?

devblogs.microsoft.com/oldnewthing/20230310-00

March 10, 2023

Raymond Chen

When you register a window class, one of the class styles is `CS_GLOBALCLASS`. When should you use this style?

The documentation says that the `CS_GLOBALCLASS` style creates an application global class, which is a window class that is available to all other modules in the process.

Specifically, it means that anybody in the process can call `CreateWindow` with your class name and summon your window class, even if they get the `HINSTANCE` wrong.

Recall that window classes are looked up by the pair (`HINSTANCE`, class name). The purpose of the `HINSTANCE` is to give each module its own private namespace, so that one DLL's `MyWindow` doesn't collide with another DLL's `MyWindow`. In order to create a window, you must specify both the `HINSTANCE` and the class name.

If a class is registered as a "global class", then it becomes possible to create a window of that class even if the `HINSTANCE` does not match. The primary audience for this is control libraries: If a control library registers its controls as global, then the controls can be created from a client DLL's dialog template: When a dialog is created from a template, the `HINSTANCE` passed to the dialog box function is combined with the class name in the dialog template to create each window. Normally, this means that you can create only controls that were registered to that `HINSTANCE`. But if the class cannot be found via (`HINSTANCE`, class name) lookup, then the system will look for a global class with the same class name. Registering your controls library controls as global classes therefore allows them to be created from any dialog template.

Some people read that description and conclude, "Sounds like global classes are all upside, no downside, so I'll register all of my classes as global classes."

But there *is* a downside. It's the tragedy of the commons.

If you decide to register all of your classes as global, then your classes are no longer scoped to your `HINSTANCE` and instead go into the list of global classes. Your `MyWindow` class, which you thought was private to your module, is now being exposed to all modules, and anybody

can

```
HWND hwnd = CreateWindowW(L"MyWindow", L"Any title",
    WS_CHILD | WS_BORDER,
    x, y, width, height, parentWindow, nullptr,
    hinstAnything, lpParam);
```

to create your window with an arbitrary `lpCreateParams`. The code that did this probably meant to create a window from their own `MyWindow` class, but they had a bug where they passed an incorrect `hinstAnything` (maybe the variable was uninitialized), and they ended up creating your window instead. You put your window class in the town square for anyone to use.

Things get even worse if *two* people did this. Now you have two modules dumping their private business in the town square, and if both of them happen to choose the same name for a window class, the first one will register the global class successfully, and the second one will fail with `ERROR_CLASS_ALREADY_EXISTS`. The second module will probably stop working at that point.

Now, even if you register all your classes to your own `HINSTANCE`, other people can still create it if they have both the instance handle and the class name. You can document the class name in your library's header file, and people can create it via `CreateWindow`:

```
HINSTANCE hinstContoso = LoadLibraryW(L"ContosoControls.dll");
HWND hwnd = CreateWindow(WC_CONTOSOGRID, L"Grid title",
    WS_CHILD | CGS_GRIDLINES,
    x, y, width, height, parentWindow, nullptr,
    hinstContoso, lpParam);
```

You need `CS_GLOBALCLASS` only if you want outsiders to be able to create your control from a dialog template:

```
IDD_RESULTS DIALOG 32, 32, 160, 280
CAPTION "Query results"
BEGIN
    CONTROL "", IDC_GRID, WC_CONTOSOGRID, CGS_GRIDLINES,
            4, 4, 152, 272
END
```

In order to look up `IDD_RESULTS`, the `HINSTANCE` must be the module that contains the dialog template. But that's not the module that contains the registration for `WC_CONTOSO-GRID`. Registering `WC_CONTOSOGRID` as a global class works around that problem.

TL;DR: Use `CS_GLOBALCLASS` only for window classes that are intended to be created by others via dialog box templates.