# What’s the atom returned by RegisterClass useful for?

devblogs.microsoft.com/oldnewthing/20041011-00

October 11, 2004

Raymond Chen

The `RegisterClass` and `RegisterClassEx` functions return an `ATOM`. What is that `ATOM` good for?

The names of all registered window classes is kept in an atom table internal to USER32. The value returned by the class registration functions is that atom. You can also retrieve the atom for a window class by asking a window of that class for its class atom via `GetClassWord(hwnd, GCW_ATOM)`.

The atom can be converted to an integer atom via the `MAKEINTATOM` macro, which then can be used by functions that accept class names in the form of strings or atoms. The most common case is the `lpClassName` parameter to the `CreateWindow` macro and the `CreateWindowEx` function. Less commonly, you can also use it as the `lpClassName` parameter for the `GetClassInfo` and `GetClassInfoEx` functions. (Though why you would do this I can’t figure out. In order to have the atom to pass to `GetClassInfo` in the first place, you must have registered the class (since that’s what returns the atom), in which case why are you asking for information about a class that you registered?)

To convert a class name to a class atom, you can create a dummy window of that class and then do the aforementioned `GetClassWord(hwnd, GCW_ATOM)`. Or you can take advantage of the fact that the return value from the `GetClassInfoEx` function is the atom for the class, cast to a `BOOL`. This lets you do the conversion without having to create a dummy window. (Beware, however, that `GetClassInfoEx`‘s return value is **not** the atom on Windows 95-derived operating systems.)

But what good is the atom?

Not much, really. Sure, it saves you from having to pass a string to functions like `CreateWindow`, but all it did was replace a string with with an integer you now have to save in a global variable for later use. What used to be a string that you could hard-code is now an atom that you have to keep track of. Unclear that you actually won anything there.

I guess you could use it to check quickly whether a window belongs to a particular class. You get the atom for that class (via `GetClassInfo`, say) and then get the atom for the window and compare them. But you can't cache the class atom since the class might get unregistered and then re-registered (which will give it a new atom number). And you can't prefetch the class atom since the class may not yet be registered at the point you prefetch it. (And as noted above, you can't cache the prefetched value anyway.) So this case is pretty much a non-starter anyway; you may as well use the `GetClassName` function and compare the resulting class name against the class you're looking for.

In other words, window class atoms are an anachronism. Like replacement dialog box classes, it's one of those generalities of the Win32 API that never really got off the ground, but which must be carried forward for backwards compatibility.

But at least now you know what they are.

[Typos fixed October 12.]

Raymond Chen

**Follow**