Strange uses for window class atoms

devblogs.microsoft.com/oldnewthing/20080501-00

May 1, 2008

Raymond Chen

When you register a window class (with the `RegisterClass` or `RegisterClassEx` function), you get an `ATOM` back. What use is this atom?

Not much.

You can use this atom in many places where a window class name can be used; just convert it to a string with the `MAKEINTATOM` macro. Let's change our scratch program to illustrate:

```
ATOM g_atmClass;
BOOL
InitApp(void)
{
    ...
    g_atmClass = RegisterClass(&wc);
    if (!g_atmClass) return FALSE;
    ...
}
int WINAPI WinMain(HINSTANCE hinst, HINSTANCE hinstPrev,
                   LPSTR lpCmdLine, int nShowCmd)
{
    ...
        hwnd = CreateWindow(
            MAKEINTATOM(g_atmClass),        /* Class Name */
            "Scratch",                      /* Title */
            WS_OVERLAPPEDWINDOW,            /* Style */
            CW_USEDEFAULT, CW_USEDEFAULT,   /* Position */
            CW_USEDEFAULT, CW_USEDEFAULT,   /* Size */
            NULL,                           /* Parent */
            NULL,                           /* No menu */
            hinst,                          /* Instance */
            0);                             /* No special parameters */
    ...
}
```

We save the atom returned by the `RegisterClass` function and use it (in the form of a `MAKEINTATOM`) in place of the class name. if you run this program, you'll see that it works exactly the same as the old version that used the class name. The class atom is valid as long as the class remains registered.

Functions that accept a `MAKEINTATOM` as the class name include `CreateWindow`, `FindWindow`, `GetClassInfo`, and `UnregisterClass` (and the `Ex` versions of them).

Why would you do this?

Well, there really isn't much reason. The string name works just as well as the atom, so the atom is just one more thing to keep track of. However, even though you don't use it, you have to be aware that other people might. For example, the `lpszClass` member of the `CREATESTRUCT` structure is usually a pointer to a string, but it could be a `MAKEINTATOM` if somebody decided to pass an atom instead of a string to `CreateWindow`. Those of you who've read the first Bonus Chapter of my book are already familiar with the program that crashed when somebody created a window via an atom.

There is one interesting thing you can do with the atom: If you have a valid class atom, you can quickly tell whether a window belongs to that class by checking the window word for the atom:

```
if (GetWindowWord(hwnd, GWW_ATOM) == atom) ...
```

This technique saves you the trouble of calling `GetClassName` and then doing a string comparison, reducing it instead to an integer comparison. This technique makes it very simple to write a `TestIfWndIsDialog` function:

```
BOOL TestIfWndIsDialog(HWND hwnd)
{
  return GetWindowWord(hwnd, GWW_ATOM) == (ULONG_PTR)WC_DIALOG;
}
```

Exercise: Discuss the limitations of the above `TestIfWndIsDialog` function.

Raymond Chen

Follow