Using the wrong HINSTANCE in RegisterClass is like identity theft

devblogs.microsoft.com/oldnewthing/20110715-00

July 15, 2011

Raymond Chen

Last year, I left you with a teaser for a problem that resulted in the `CResourceException` being thrown.

Studying the function that threw the exception revealed that it was thrown due to a failed call to `RegisterClass`. And studying the parameters that were passed to `RegisterClass` revealed that `HINSTANCE` parameter did not match the DLL. Instead of being the instance handle of the DLL, it was the instance handle of the host application.

Okay, now let's apply what we learned a few years ago about the significance of the `HINSTANCE` parameter passed to the `RegisterClass` function. By passing the `HINSTANCE` of the host application, the class was registered against the namespace of the host rather than the namespace of the DLL. It's like signing up for a credit card using somebody else's name or checking a book out of the library with somebody else's library card.

In this case, the module in question was a plug-in. It tried to register a class called, say, `MyClass`, and instead of registering against itself, it registered against the host application. Fortunately, the host application didn't have a class called `MyClass`, so the incorrect registration didn't cause a conflict. The book got checked out to the wrong person, but as far as the library can tell, nothing has gone wrong. It merely looks like the host application checked out a book.

So why did the call to `RegisterClass` fail? Because *some other plug-in made the same mistake*. Plug-in B also registered its class against the host application, and by an amazing coincidence, its class was also called `MyClass`. (If you look at how MFC auto-generates class names, you can see that this name collision can happen quite easily.) If both plug-ins had registered their classes properly, there would have been no problem, because each class would have been registered against their respective DLLs, and no conflict would have arisen. But instead, two wrongs make a wronger, and since both plug-ins incorrectly registered their classes against the host, the first plug-in to register succeeds, and the second one crashes.

(One might argue that this is another special case of *What if two programs did this?*)

Both plug-ins tried to sign up for a credit card in the name of the host application. The first one got the card, and the second one was informed by the credit card company, "Your application was denied because you already have a card from us."

Why did these plug-ins register against the host application instead of against their own library cards? I don't know for sure, but my guess is that it was due to ignorance.[1] When reading the documentation, they found that they needed to fill in the `hInstance` member of the `WNDCLASS` structure. Gosh, where do I get an `HINSTANCE` from? Oh wait, I found a function that returns an `HINSTANCE`: `GetModuleHandle`. And hey look, if I pass `NULL`, a valid `HINSTANCE` comes out. I'll just set `wndclass.hInstance = GetModuleHandle(NULL);` and try it. Hey, look, it works!

Footnote

[1] This reminds me of a story that took place at an administrative hearing. The government agency representative presented as evidence that the other party admitted in a telephone conversation to being ignorant of the applicable regulations.

The other party angrily interrupted.

"I'm not ignorant! I simply didn't know what the rules were."

The judge patiently explained, "That's what the word *ignorant* means."

Raymond Chen

Follow