Even if you have code to handle a message, you’re allowed to call DefWindowProc, because you were doing that anyway after all

devblogs.microsoft.com/oldnewthing/20090105-00

January 5, 2009

Raymond Chen

Just because you write `case WM_SOMETHING:` doesn’t mean that you have to handle all possible parameters for the `WM_SOMETHING` message. You’re still allowed to call the `DefWindowProc` function. After all, that’s what you did when you didn’t have a `case WM_SOMETHING:` statement in the first place.

```
switch (uMsg) {
case WM_CHAR:
    OnChar(…);
    return 0;


default:
    return DefWindowProc(…);
}
```

The above code fragment doesn’t handle the `WM_SOMETHING` message at all. Suppose the `WM_SOMETHING` message uses the `wParam` parameter to specify what type of something occurred, and you only want to override the default processing in the case where `wParam` has the value of 4. What do you do with the other values?

```
switch (uMsg) {
case WM_CHAR:
    OnChar(…);
    return 0;


case WM_SOMETHING:
    if (wParam == 4) { DoSomething4(…); }
    else … ????? …
    return 0;


default:
    return DefWindowProc(…);
}
```

If the value is 4, then you do your special “something 4” processing, but what about all the other values? How do you handle them?

Well, think about it: How did you handle them before? The original code, before you added a `WM_SOMETHING` handler, was equivalent to this:

```
switch (uMsg) {
case WM_CHAR:
    OnChar(…);
    return 0;


case WM_SOMETHING:
    return DefWindowProc(…);


default:
    return DefWindowProc(…);
}
```

In the original code, since there was no explicit handler for the `WM_SOMETHING` message, control is transferred to the `default` case handler, which just calls the `DefWindowProc` function. If you really want to, you can expand the case out a bit more:

```
switch (uMsg) {
case WM_CHAR:
    OnChar(…);
    return 0;


case WM_SOMETHING:
    if (wParam == 4) return DefWindowProc(…);
    else return DefWindowProc(…);


default:
    return DefWindowProc(…);
}
```

Because if the `wParam` is 4, the original code just called `DefWindowProc`. And if the `wParam` was something other than 4, the original code still just called `DefWindowProc`.

Of course, I expanded the block in precisely this way so it matches up with the case we started writing when we decided to handle the `WM_SOMETHING` method. Written out this way, it becomes obvious what to write for the question marks.

```
switch (uMsg) {
case WM_CHAR:
    OnChar(…);
    return 0;


case WM_SOMETHING:
    if (wParam == 4) { DoSomething4(…); }
    else return DefWindowProc(…);
    return 0;


default:
    return DefWindowProc(…);
}
```

Just because you have a `case WM_SOMETHING` statement doesn't mean you have to handle all the cases; you can still call `DefWindowProc` for the cases you don't want to handle.

Armed with this information, you can help commenter Norman Diamond handle the `VK_F10` key in his `WM_SYSKEYDOWN` message handler without having to "start handling a bunch of keys that really are system keys, that I didn't want to bother with."

Raymond Chen

Follow