

Re: MFC and c++ problems

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2006-02/msg00478.html>

- *From:* Daniel James <wastebasket@xxxxxxxxxxxxxxxxxxx>
 - *Date:* Sat, 04 Feb 2006 14:13:38 GMT
-

In article news:<fa63u1pin4p62hk7umrn7bvrjvhlm3tk29@xxxxxxx>, Joseph M. Newcomer wrote:

So a control subclass becomes a completely self-contained class in my world.... It cannot look into any other class that might contain it, including CWinApp, CView, or CDialog classes that might contain it. GetParent()->SendMessage is the interface of choice, and this can usually be concealed with some reasonable programming techniques.

I agree with most of what you wrote in that long post, joe, but I have to play Devil's Advocate with you here for a moment ...

You say that a control must make no assumptions about its parent.

But, if a control sends a message to its parent, is it not making the assumption that its parent can handle that message? How is this any better than calling a method defined by the parent?

Semantically, of course, the parent is free not to handle any message that it doesn't understand. Your program containing a control that sends messages to its parent will compile and run ... but may not WORK, because the control may require the parent to handle the messages it sends, and make some response.

I agree wholeheartedly with you that it is a great strength of OO programming that there is no coupling between unrelated program components ... but here we're dealing with components which do have some logical (i.e. real-world) coupling. We can't hide from that, and we should be prepared to allow it to be manifest in our expressions of the program.

If MyControl calls methods in its parent directly there is a visible connection between control and parent that documents the real-world connection between their behaviours. If you place that control in a parent that doesn't provide the methods that the control needs to call your program will fail to compile ... which is rather better, in my book, than having it compile and run but not work.

One could write an interface class that supported the method(s) needed by

Re: MFC and c++ problems

the control, and make sure that any parent class using that control inherited from the interface. One would pass the control's constructor a pointer/reference to the parent *as an interface object* through which the calls could be made ... that way the control does not need to know any more about the parent class than is necessary, and the compiler can still catch the error if an attempt is made to place the control in an inappropriate parent.

The trouble with using SendMessage is firstly that it *cannot* be made typesafe in this way, and secondly that it imposes a (small) runtime overhead.

Having made those points ... I must say that for most scenarios I can think of I would actually use the SendMessage technique; for the usual commonplace control-notifies-parent-of-user-action situations it is what is expected. Windows uses messages for notifications of this kind all the time, and allows messages to be spied upon, hooked, redirected, etc.; and supports subclassing of the message sender/receiver without breaking the overall model.

It's not a completely cut-and-dry issue, though ...

Cheers,
Daniel.

.