Subject: Re: IDL objected oriented question
Posted by pashas77 on Wed, 09 Apr 2003 13:40:15 GMT
View Forum Message <> Reply to Message

David Fanning <david@dfanning.com> wrote in message
news:<MPG.18fcb0a0da584bdf989b3a@news.frii.com>...
> Sabir Pasha (pashas77@yahoo.com) writes:
>
>> I'm a relative newbie to IDL. I'm working on with classes right now.
>> I have a class which has objects as member variables. At runtime via
>> the famous Info structure, I find that I need to use the objects
>> member functions. But lo and behold, encapsulation is implemented in
>> IDL 5.6(I don't believe that it was implemented in 5.5...correct me if
>> I'm wrong).
>
> You're wrong. :-)
>
>> Basically
>>
>> define = { ClassA, $
>>
>> ObjectB: Obj_New()}
>>
>> END
>>
>> the object gets defined in
>> ObjectB = Obj_New("ClassB")
>>
>> And somewhere we define ObjectA
>>
>> ObjectA = Object_New("ClassA)
>>
>> and now in an event handler far far away
>>
>> Sinfo.objectA.objectB->member function
>>
>> doesnt' work because we cannot access Objects A's member variables
>> only member functions.
>
> Exactly.
>
> Perhaps you meant to INHERIT objectB, in which case
> you could use all its methods and data directly in objectA.
> But perhaps not. There are good reasons sometimes to simply
> have objects as members of other objects.
>
> Working with member objects in event handlers is tough,
> because, of course, you have to have some way to *get*

> the object you are interested in manipulating.
>
> One way to do this is like this:
>
> info.objectA -> GetProperty, ObjectB=objectB
>
> Now you can call the methods on objectB directly:
>
> objectB -> DoYourThing
>
> This sort of defeats the purpose of object encapsulation,
> but there you are. :-)
>
> I would argue that ObjectA is the only one who is suppose to
> know anything about ObjectB (since it is member data for
> ObjectA), so anything that is done to it should be done
> in an ObjectA method. This means you don't have to get
> ObjectB, since it is already there:
>
> PRO ObjectA::SomeMethod
>
> self.objectB -> DoYourThing
>
> END
>
> The problem you have is that you are not in objectA's methods,
> but in an event handler. A bummer. :-)
>
> Dave Burridge and I have solved this problem with our Catalyst
> Object Library by wrapping all widgets up as objects. Then widget
> events automatically get sent to event handler *methods* rather
> than event handler procedures. This makes it possible to write
> widget programs in the normal way, but you get to take advantage
> of the many lovely properties of objects, too. It is the best
> of both worlds, really.
>
> Another huge advantage of our library is that it is based on
> object containment hierarchies, which means objects get cleaned
> up and destroyed almost magically. You almost never have to worry
> about leaking memory, one of the most annoying problems with writing
> large object programs. Objects can have many "parents", or objects
> that care about them (three different views of a volumetric data object,
> for example), but an object will only be destroyed when all the
> parents have died. In our Catalyst world, children *always* outlive
> their parents. :-)
>
>> Is there a equivalent to the "public" keyword in C++.
>

> No, probably in IDL 6.1. :-)
>
> (I don't know this, I only mention it for the amusement of
> the IDL newsgroup regulars.)
>
>> So I wanted to ask the IDL gurus out there, how you overcome these
>> problems in very large IDL programs.
>
> For very large programs, I use our Catalyst Library. I wouldn't
> think of using anything else. For one thing, it reduces development
> time by at least 25-50% by already providing a framework for building
> large applications, not to mention the sizeable library of
> building blocks that grow daily.
>
> Cheers,
>
> David

Thanks all for the prompt reply. Yes, I thought about obtaining the objectB via a member function of Object A, but exactly as Mr. Fanning said, that would defeat the point of encapsulation. Inheritance, I think would be inappropriate in this case, because again, Object A does not need access to all of Objects B's member variables, thus breaking encapsulation again. Those event handlers are sometimes quite the monkey's wrench.

I think I'll end up using this method:

```
PRO ClassA::SomeMethod

    self.objectB -> DoYourThing

END
```

I guess i'll have to wait until the IDL includes the "public" keyword(don't hold my breath, I'm guessing??) Or as was subtly mentioned, get the Catalyst library....:)

Thanks again for the help, much obliged.

Sabir Pasha