Subject: Object Madness or Restoring Nightmares
Posted by David Fanning on Wed, 03 Mar 2004 00:08:57 GMT
View Forum Message <> Reply to Message

Folks,

Don't you just hate it when you think you understand
something, only to realize (usually at a critical time)
that you don't?

Maybe I've been doing too much programming and not playing
enough tennis lately, but I feel t-i-r-e-d. Good thing
the ol' physical is tomorrow. Maybe I ask for some of those
tiny blue programming pills. :-)

Anyway, here's the deal. I have an object containing
several other objects. All of these objects know how to
clean themselves up. If I destroy the main object, let's
call it the "study object", then all is fine. No memory
leakage.

Now, I want to save this object in my IDL application.
The idea is that I can have several sessions hanging
around and I can restore and continue working on any
of several different studies. So no problem saving
the session as an IDL save file:

```
currentStudy = self.currentStudy
Save, Filename='somename.sav', currentStudy
```

And I can restore it OK:

```
Obj_Destroy, self.currentStudy
Restore, Filename='somename.sav'
self.currentStudy = curentStudy
```

This works great....*except* when I restore like this
and exit my application (thereby doing an Obj_Destroy
on self.currentStudy), I am left with *lots* of leaking
memory. I don't know why. (Or, more accurately, I think
I *do* know why, I just can't remember it.)

I've proved that it is not the save/restore cycle that is
doing this, because if the study contains just non-objects,
say images, then there is no memory leakage. Only when the
study contains objects do I leak.

Any good ideas?

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Object Madness or Restoring Nightmares
Posted by David Fanning on Thu, 04 Mar 2004 15:43:14 GMT
View Forum Message <> Reply to Message

Tom McGlynn writes:

> Forgive me if I'm asking stupid questions... (OK the if is superfluous!)

Oh, Lord, let us all not become pious in Your presence,
for the wailing and gnashing of teeth would be a din unknown
on this good earth! Amen.

> Clearly each object contains pointers to all of its children
> so if you save the parent all the objects contained in it
> are saved. But I don't see why a child (still taking about the containment
> hierarchy, not the inheritance tree) needs to point to its
> parent? Where is that pointer coming from and what is it doing?

Well, maybe this is the design issue that needs examining. :-)

Our notion was that this would be an object hierarchy similar
in design and operation to a widget hierarchy. That is to say,
each object would be able to identify its parent and its children.
Primarily, this is for communication sake. We ourselves want to
traipse around the hierarchy looking for particular objects.
For example, our objects generate "events" or "messages" that
can traverse the object hierarchy, just as widget events do.
If an "event" gets to an object, and that object doesn't have
an EventHandler method, then the event is passed on to the parent
of that object and so on.

So every object has a "parent" field, which is an object reference
to another object. I think this is why IDL has to get everything.
And, of course, in the other direction, if an object is holding
other objects, you have to get them as well.

> I gather that each object needs to point to the class definition of the

> top level container since that's also the class definition of
> the root of the inheritance tree, but I wouldn't have thought
> that saving the definition of the class means that you
> have to save every instance of the class. That would certainly
> seem like a broken implementation for the SAVE functionality.

Each object doesn't have to know about the top-object.
Each object just knows about the object *above* it. But
all paths lead (eventually) to the top-object.

I finally solved my problem (this morning) by performing
an object "copy", and saving the copies, not the original
objects. A bit of a pain, but I think I do understand more
about the issues, which means I'll have a better idea of
how objects need to be designed from now on. Little
solace, I know, but it's *something*! :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Object Madness or Restoring Nightmares
Posted by James Kuyper on Thu, 04 Mar 2004 16:51:05 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
...
> So every object has a "parent" field, which is an object reference
> to another object. I think this is why IDL has to get everything.
> And, of course, in the other direction, if an object is holding
> other objects, you have to get them as well.

This points to a possible approach: can you set it so a given object has
no parent? If so, it may also be possible to "orphan" an object, cutting
it off from it's parent. That should allow you to save the object,
without having to save all of it's ancestors.

Subject: Re: Object Madness or Restoring Nightmares
Posted by David Fanning on Thu, 04 Mar 2004 17:08:49 GMT
View Forum Message <> Reply to Message

James Kuyper writes:

> This points to a possible approach: can you set it so a given object has
> no parent? If so, it may also be possible to "orphan" an object, cutting
> it off from it's parent. That should allow you to save the object,
> without having to save all of it's ancestors.

Well, I thought about this, but it is slightly more complicated than I have described so far. Unlike widgets, for example, our objects can have multiple parents. (Our notion of a "parent" is "an object that cares about you".)

You can imagine, for example, that you want an image displayed in three different draw widgets. Maybe a zoomed image, a normal image, and a thumbnail image. Each of the draw widgets would be a "parent" to the image object, because they care about the image object.

In our system, we are careful not to destroy an object unless it's an orphan, and then we ruthlessly slash its head off. (I wouldn't want to live in this world we have created!) So, to get back to your question. To remove all the parents of an object destroys the object in our system. Hence, there is nothing to save. :-(

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Object Madness or Restoring Nightmares
Posted by David Fanning on Thu, 04 Mar 2004 18:38:59 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> Why not implement a set of methods in your top-level which leverages the
> inherent connectedness to detach unnecessary objects before saving?

Humm, yes. The idea has been lurking in the back of my mind (where I have tried to suppress it) that I was going to have to deal with this sooner or later. I've *known* I was going to have trouble saving and restoring widgetObjects. But

the data objects were a surprise to me.

Your suggestions are wonderful, but I wonder if a simple COPY method at the CATADATATOM level (which all data objects inherit) just to copy data fields--all object fields would be ignored-- wouldn't work as well. This is easily over-ridden in more complicated cases.

I greatly appreciate the thoughts. Do you want to come up here and take over my business while I'm rowing in the Beaufort Sea? Get it all finished for me. How's your backhand?

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/
