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/
