Subject: Re: Discussion on global variables in IDL
Posted by John-David T. Smith on Mon, 13 Aug 2001 14:27:56 GMT
View Forum Message <> Reply to Message

Ben Tupper wrote:
>
> JD Smith wrote:
>
>>
>> So yes, I feel your pain. My special difference of opinion with IDL OOP
>> is the complete lack of publicly available instance data members,
>> forcing one through awkward and slow GetProperty calls at every turn
>> (one can verify the inherent slowness by comparing structure lookup
>> speed to method invocation/return speed). I even found code broken by
>> my desire to avoid this penalty: I was caching in one object another
>> object's data members, which later came to be more dynamic. Ouch.
>
> Oh geez,
>
> Just when I think I'm finally beginning to understand objects, JD throws
> another curveball. 'Publicly available instance data'. I'm wondering
> what that is, and would like to know what you mean; I hope you can field
> these questions for me.
>
> All I can imagine is something like the following...
>
> myObj = OBJ_NEW('OBJECT_CLASS', data)
>
> myObj.Name = 'Bob' (rather than myObj->SetProperty, Name = 'Bob')
>
> Am I getting that straight? No method is needed to fiddle with an
> object's data?

Precisely. Different languages have differing approaches to data
encapsulation. Some let you articulate the divide between public and
private data. Some leave all data public and leave it up to the
programmer to behave. And some, like IDL, lock everything away like a
distrusting father with his truck keys.

>
>> My
>> secondary complaint is the lack of class variables: variables accessible
>> to every instance of a class, for inter-instance communication.
>
> Does this mean that the variable is defined in the OBJECT_CLASS__DEFINE
> procedure so it gets intialized with the right value? Instead of my
> data field NAME being assigned the value '' at initialization, it gets
> assigned the value 'BOB' always?

No, it means variables can be defined within the class itself, not
within a given instance of the class (aka an object). You can then have
"global" variables within one class. So, for all objects a b c d of the
same class, a.class_var, b.class_var, c.class_var, and d.class_var are
the same actual variable, and can be used to communicate among them.

Why would you want to do that? Take a look at a post from a few years
ago on a Singleton method. There are lots of other good uses you can
imagine, also.

But anyway, we should all be glad IDL's higher level programming tools
are somewhat crude; otherwise, it might be adopted by an open source
develpment project and a raft of 13 year old kernel programmers would
flood our happy little newsgroup with increasingly irrelevant code
snippets in a pseudo-machismo recognized only by their kind. They'd
probably even start up untold "incomprehensible discussions on Histogram
functionality." Right, David?

JD