Subject: Re: _ref_extra
Posted by John-David T. Smith on Mon, 13 Aug 2001 16:47:26 GMT
View Forum Message <> Reply to Message

>
> JD Smith wrote:
>>
>> Despite the inconvenience, GetProperty as it is does have one thing in
>> its favor: if you just allow those fields to be "gotten" that you won't
>> mind keeping the same, you can isolate yourself from your own (OK, my
>> own) tendency to perform quick-fixes by digging deeper than you should.
>
> The whole reason I tried to make a uniform Get_property (G_P) method is
> because I decided that the authou of the code is allowed access to every
> single field of the object, and can decide how he uses those fields. G_P
> is solely for returning *contents* of several fields in one pass. In my
> opinion, if you want G_P to return a calculated value, it needs to
> become a separate method, or else it will become a nightmare after
> several calculations are added to G_P.
> I also have a function called Return_property (R_P) that returns just
> one field of the object. This is convenient when one field is all you
> need. lets say for passing that value as an argument.
> BTW, both G_P and R_P are unaware and don't care about what they will be
> called upon. All they need is to be recompiled with a correct class
> name. Unfortunately, I have not come up with an elegant way for G_P, so
> I will not post it here for now. I can't come up with a hack to break
> into _Ref_extra or get variable names passed via _extra, either.
> Oh, forgot to say that Set_property (S_P) works the same exact way.
>
>> My recommendation: only add GetProperty keywords when you run into the
>> first time you actually *need* that value
>
> This is the whole idea: I am not adding *any* explicit keywords to G_P,
> R_P or S_P, because it is too much hassle especially when your object is
> immature and gets a field added every now and again. My way, I don't
> care if I add a field: I reset IDL and G_P works on new fields as well
> as on the old ones.

Ahh yes. The danger is of course, if other programmers (our yourself),
come to rely on explicit details of your class structure. Like
obj.image being an array, and not a pointer to an array.

What if later you decide to rework the entire class? If they (and you)
had stuck to a known interface, you could recode without breaking
programs which use your class. The temptation of fully open data
members access is high, and I for one have pined for a native method
within IDL to allow this. This does not mean, however, that allowing
such carte blanche access is always good idea. Typically, a *small*

subset of a class' data fields are useful and stable for public consumption.

JD