Subject: _ref_extra
Posted by Pavel A. Romashkin on Wed, 10 Jan 2001 22:54:42 GMT
View Forum Message <> Reply to Message

I have a question.
When a number of keywords is passed through _ref_extra into called pro's
_extra structure, is there a way to break directly into the keyword
storage locations and update the values of keywords without specifying
each keyword in the called procedure explicitly?
i. e., in the called procedure, do the assignment in the form of
extra.(0) = '10', extra.(1) = 2.25 etc, so that the newly assigned
values get passed back to the caller procedure?
Thanks,
Pavel

Subject: Re: _ref_extra
Posted by Pavel A. Romashkin on Fri, 10 Aug 2001 18:31:02 GMT
View Forum Message <> Reply to Message

I can't get to that URL, David.
Sorry for stirring up this discussion again. I mean no harm. I did
Google search, too, but found nothing.
Pavel

David Fanning wrote:
>
> Oh, no. Not this discussion again! :-)
>
> Here are my raw notes for an article on the
> topic I've been meaning to write for several
> years now:
>
> ftp://www.dfanning.com/pub/dfanning/outgoing/misc/ref_extra. txt

Subject: Re: _ref_extra
Posted by david[2] on Fri, 10 Aug 2001 19:39:16 GMT
View Forum Message <> Reply to Message

Pavel A. Romashkin writes:
>
> I can't get to that URL, David.

Whoops. Sorry. This works:

ftp://ftp.dfanning.com/pub/dfanning/outgoing/misc/ref_extra. txt

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: _ref_extra
Posted by Pavel A. Romashkin on Fri, 10 Aug 2001 19:44:15 GMT
View Forum Message <> Reply to Message

Oh well. Just thought I'd ask. I got around it after all, although it
took me extra 20 lines of code. Basically, I did not want to write
get_property methods the old way anymore, inspired by David's complaint
that then you have to revisit them every time you need to alter the
object. I have one get_property for everything now, and it works :-)
Thank you,
Pavel

Subject: Re: _ref_extra
Posted by John-David T. Smith on Fri, 10 Aug 2001 20:07:32 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" wrote:
>
> Oh well. Just thought I'd ask. I got around it after all, although it
> took me extra 20 lines of code. Basically, I did not want to write
> get_property methods the old way anymore, inspired by David's complaint
> that then you have to revisit them every time you need to alter the
> object. I have one get_property for everything now, and it works :-)
> Thank you,
> Pavel

And do you mind sharing it? I once did this (in the pre-_REF_EXTRA
days) by changing GetProperty to a function, returning a structure with
keyword:value pairs, and chaining down the inheritance tree by
concatenating the structures returned. I abandoned it when _REF_EXTRA
arrived.

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.
My recommendation: only add GetProperty keywords when you run into the
first time you actually *need* that value, and even then spend some time
convincing yourself that you're happy to fix that field in stone, no
matter what other changes the remainder of the class undergoes.

JD

Subject: Re: _ref_extra
Posted by Pavel A. Romashkin on Mon, 13 Aug 2001 16:19:52 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.

Cheers,
Pavel

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

Subject: Re: _ref_extra
Posted by Pavel A. Romashkin on Mon, 13 Aug 2001 17:37:43 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
>
> 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.

All I have so far utilizes no public access to S_P and G_P at all - user code only deals with interface, or "object I/O", methods. And if a user is curious enough and willing to get to the very source code, he can change anything anyway and get to object fields. And if that happens, unified S_P and G_P will still work.
For me, the universal methods are mainly for writing other interface and processing methods, and not for direct public use. But even in that case, you only have to document those keywords you want :-)

Cheers,
Pavel

Subject: Re: _ref_extra
Posted by Martin Schultz on Thu, 23 Aug 2001 06:58:06 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@astro.cornell.edu> writes:

>>
>> JD Smith wrote:
>>>
> ... 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

Wouldn't it be lovely, had the folks at RSI thought about a "public" and "private" attribute for object fields? This is another occasion where the quick-and-dirty IDL clashes with the programming language IDL.

BTW: Here's the 7 liner that I use in my Base object (and hence all objects) to do what Pavel advertises. I haven't tried if Struct_Assign also works the other way round ...

Cheers,

Martin

```
; ------------------------------------------------------------ -------------------------
; GetState:
; This method returns the object "data" as a structure. Pointers and
; object references are not copied, i.e. they are returned as
; unitialized variables (NullPointer)!

FUNCTION MGS_BaseObject::GetState

  ok = Execute('retval = {'+Obj_Class(self)+'}')  ;; Same trick as in Copy
  IF ok THEN BEGIN
    Struct_Assign, self, retval
  ENDIF ELSE BEGIN
    retval = { nothing: 0L }
  ENDELSE

  RETURN, retval

END
```

--
[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[[