Subject: Re: It seems that there is *a bit* polymorphism in IDL.
Posted by JD Smith on Thu, 12 Aug 2004 17:28:57 GMT
View Forum Message <> Reply to Message

On Wed, 11 Aug 2004 21:05:49 -0600, David Fanning wrote:

> David Fanning writes:
>
>> Amen to this! In fact, generalized GetProperty functions are so easy to
>> write and so useful for obtaining any *single* object property, that
>> *all* your objects should have one, along with the usual GetProperty
>> procedure. :-)
>>
>> http://www.dfanning.com/tips/getproperty.html
>
> OK, IDL 6.1 arrived this afternoon, so here is a programming quiz for all
> of you who are not currently Members in Good Standing with IEPA
> (http://www.dfanning.com/misc_tips/iepa.html). (This includes all of you
> who have forgotten to pay your dues this month.)
>
> Use the SCOPE_VARNAME function with the REF_EXTRA keyword to write a short
> general purpose GETPROPERTY *procedure* method that'll return any member
> variables referenced via keyword names

At long last... legitimized access to the oft-defamed (and poorly named) ROUTINE_NAMES() variable functionality! I can now feel quite justified in my various out-of-scope variable slinging (which is used heavily in IDLWAVE debugging). Very cool new _REF_EXTRA keyword in SCORE_VARFETCH should enable lots of neat functionality, including the one you mention (I presume you meant that one instead of SCOPE_VARNAME).

Also welcome are the left-aligned and zero-padded updates for the FORMAT codes.

On the original topic, I was going to suggest a general purpose execute-based GetProperty function like:

```
function MyClass::GetProperty,_EXTRA=e
  if n_elements(e) eq 0 then return,-1
  prop=(tag_names(e))[0]
  void=execute('self->GetProperty,'+prop+'=ans')
  return,ans
end
```

which uses the GetProperty procedure method (which, in my case, often does more than simply return a field of the class structure). This uses EXECUTE as well, so is to be avoided for VM code. Your requested

solution is also trivial:

```
pro MyClass::GetProperty,_REF_EXTRA=e
  tags=tag_names(create_struct(NAME=obj_class(self)))
  for i=0,n_elements(e)-1 do begin
     wh=where(tags eq e[i],cnt)
     if cnt eq 0 then continue
     (scope_varfetch(e[i],/REF_EXTRA))=self.(wh[0])
  endfor
end
```

Note this uses the new (I think, don't have IDL 6.1 yet) functionality for 'CREATE_STRUCT,NAME=' I had just pined for: the ability to create a named structure programmatically without resorting to EXECUTE.

This still doesn't solve the problem of enabling rapid access to object data members: this function will take many hundreds or thousands of times longer to execute than a similar structure field dereference would take. I find that for speed reasons I'm often caching copies of some object's internal fields inside of other objects, which can lead to problems if the cache is not kept up to date. Encapsulation is great, but the penalty for routine calls is too high for some event-driven situations to make good use of it.

I suspect you should also be able to do something clever with SCOPE_VARFETCH and a GetProperty function, like:

```
function MyClass::GetProperty,_REF_EXTRA=e
  self->GetProperty,_EXTRA=e
  return,scope_varfetch(e[0],/REF_EXTRA)
end
```

Obviously untested, since I don't have 6.1.

Fun stuff.

JD
