Subject: Re: Copying (cloning) objects -- Testers wanted
Posted by John-David T. Smith on Fri, 09 Jun 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Martin Schultz wrote:
```
>    this is more and more fun! Please find attached a base object
>  definition which contains a few useful features that you can nicely
>  inherit in anything else:
>  The object contains a NAME and a UVALUE field which can be set with the
>  Init method (where NAME must be a string and UVALUE can be anything). It
>  contains a method for displaying an error message either as a dialog box
>  or on the log screen (controlled by the no_dialog keyword), and -- and
>  this I am most proud of -- it contains a method for duplicating itself.
>  And this works for any object that inherits from it. Thanks to James
>  Tappin who gave me the idea to create a structure with the object fields
>  ( stru = { object } ), and to JD Smith who had the Struct_Assign idea.
>  What the Copy() method does is to create a copy of self as simply as
>    ;; Initialize an empty object of the same type as self
>    clone = Obj_New( Obj_Class(self) )
>
>    ;; Do a structure assignment to copy all fields verbatim
>    Struct_Assign, self, clone, verbose=verbose
>
>  But that leaves you with the problem that pointers and object references
>  in the new object still point to the data and objects from the old
>  object. So I apply James' trick with a little variant:
>    ok = Execute('ostru = {'+Obj_Class(self)+'}')
>
>  in order to get a structure equivalent of the current object. Then I
>  loop through the structure tags and whenever there is a pointer, I call
>  a method CopyPointers which does just what it says (took me a while to
>  avoid the traps of pointer arrays and pointers pointing to object
>  references, but it is still remarkably short, I think).
<snip>
```

Some heavy hackery there. Remarkably short, eh??? I still advocate the slightly perverse but conceptually clean (and *ludicrously* short, by your standard):

```
save,obj,FILE=tmpfile & restore,tmpfile,RESTORED_OBJECTS=clone
```

with my usual caveats for later restoration, as opposed to the easy case of immediate copy going on here. This automatically takes care of all pointers and objects nested to any depth and complexity for you.

The method also permits (with additional coding) arbitrary save/restores of relevant portions of member data across IDL sessions (or even years of time) -- so you can hand pick which pointers/objects to duplicate and which to ignore.

For instance, for objects which have some widget functionality embedded within them, storing all of the usual widget info for recovery is problematic (since widget id's, etc., are state-specific). You can easily detach these prior to saving and avoid this problem to begin with. See my various long-winded posts on the intricacies of the subject if interested.

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|