Subject: Re: Object style guide
Posted by davidf on Fri, 12 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Martin Schultz (m218003@modell3.dkrz.de) writes:

> I'd be extremely happy if (especially) the gurus among you would find it
> equally useful to adopt a new format for the file header in object
> definition routines. The attached file should serve as a basis for
> discussion, I am open for any suggestions.

I appreciate you getting the discussion started about this, Martin. Documenting object programs *is* difficult, especially so because each method is its own separate procedure or function with its own parameters, keywords, etc. And then I know from long experience that just because you document something doesn't mean anyone is going to use it. You have to spell it out for them. Give them easy-to-follow examples of what you mean by the documentation.

That little DRAWCOLORS object I offered the other day took me an hour to write and 3 days to document, what with code comments and official documentation! Even with a hugely sympathetic boss (I.e., me), that is hard to justify. :-(

When you are writing widget objects you have a further complication in that some methods are really event handlers--not really public methods. Obviously these don't need to be documented the way direct interaction methods need to be. But should they be in a separate file? And what about other programs that are called from within objects (e.g., XCOLORS)? Sigh...

I'm going to have another look at the new projects capability in IDL 5.3. Maybe that is one way to keep all of this stuff straight.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com

Subject: Re: Object style guide
Posted by Karri Kaksonen on Fri, 12 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Nice idea. While you are making the template you could also
catch the errors in the init-block:

```
function init...
    CATCH, Error_Status
    ;Error_Status = 0
    if (Error_Status ne 0) then begin
        ;if (obj_valid(self.complexPlotFont)) then $
 ;        obj_destroy, self.complexPlotFont
        return, 0
    endif

    ;if (self->IDLgrModel::init(_REF_EXTRA=extra) ne 1) then return, 0
    ...
end
```

If you are creating objects for a display then you usually
encapsulate everything in a model:

```
pro %NAME%__define
    %NAME% = {%NAME%, $
               ;INHERITS IDLgrModel, $
    ...
end
```

--
Karri Kaksonen
Picker Nordstar