Subject: Re: Inheritance query
Posted by J.D. Smith on Mon, 15 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Martin Schultz wrote:
>
> In article <3829BEB9.A3D2D3A6@astro.cornell.edu>,
> "J.D. Smith" <jdsmith@astro.cornell.edu> writes:
>>
>> One more tip: make good use of the _REF_EXTRA mechanism for chaining up to
>> methods which should return something:
>>
>> pro SubClass::GetProperty,VALUE=val,_REF_EXTRA=e
>> val=self.value
>> SuperClass::GetProperty,_EXTRA=e
>> end
>>
>> This allows the SuperClass's GetProperty Method to put things into variables for
>> return (like properties of the SuperClass, which aren't always just data
>> member!), impossible with the _EXTRA mechanism.
>>
>> JD
>>
>
> Thanks JD for bringing this up! I am just experimenting a little bit
> with objects myself, and came across this _REF_EXTRA in -- I think it was
> Struan's -- code. What I don't understand is: why do you use
> _REF_EXTRA in the procedure header but then pass it on to SuperClass
> via _EXTRA? I tried to follow the online help on this but couldn't really
> find an answer. Is it simply syntax convention that one *always* uses
> _EXTRA when calling the routine that accepts _EXTRA or _REF_EXTRA
> keywords? Or is there more to it?

Martin,

When I began asking RSI for a by-reference keyword mechanism, I fully expected it to be invisible... i.e. to occur using the existing _EXTRA mechanism. In a series of discussions with the RSI programmer who wrote _REF_EXTRA last year, I gained an understanding of why _REF_EXTRA is the way it is. You can search on "IDL v5.1 impressions" for the full thread. The basic synopsis and a few things I've learned by experience:

* _REF_EXTRA was needed to preserve backwards compatibility with older code which often uses explicitly the fact that the "extra" variable was a structure with a certain format in the intermediate routine. People were commonly making their own "extra" structs, or modifying them in transit. It could be argued that this is outside the scope of what _EXTRA was intended to address.

* _REF_EXTRA need only be used in the definition of the routine for which by-reference inherited keywords are wanted. I.e. if a routine wants to pass a value back to its caller through an unspecified keyword whose value will be obtained from another routine called there, it must be defined with _REF_EXTRA. Example:

```
pro r3, R3_VAL=g
 g=81
end

pro r2, R2_VAL=g,_REF_EXTRA=re
 g=42
 r3,_EXTRA=re
end

pro r1, R1_VAL=g,_REF_EXTRA=re
 g=8
 r2,_EXTRA=re
end

r1, R1_VAL=v1,R2_VAL=v2,R3_VAL=v3
```

Here we wanted to put a value in v1,v2,and v3, from routines called at various depths down in the calling heirarchy. See below for an explanation of the various uses of _REF_EXTRA and _EXTRA.

*For any routine *calling* syntax, the plain old _EXTRA can and should be used, and IDL will automatically *know* whether you're using the new or the old method. This is confusing, but saves having to go through all old code and update _EXTRA->_REF_EXTRA in the calling sequences. RSI could have required all _REF_EXTRA's to be used in definitions and calling sequences symmetrically, but they spared us that agony (though not the resultant confusion).

* Having said that, if you are never "peeking behind the curtain" in your inherited keyword routines -- are never modifying or changing or creating from scratch the standard extra structure, but just simply passing them through to one or more subsidiary routines -- you can simply use _REF_EXTRA in routine definitions always. It's much faster than _EXTRA, and has the nice properties of by reference that it should have had in the first place, using exactly the same rules as arguments and normal keywords do. I think of _REF_EXTRA as the way _EXTRA should have been done in the first place.

* Inside the routine with a definition including _REF_EXTRA=re, the variable re is a string array with the names of the extra keywords passed. The *values* are nowhere to be found... they are invisible, and only accessible by those routines called from this routine with _EXTRA=re.

* Never *call* a routine with _REF_EXTRA. It will compile and run, but it

probably won't do what you want (it greedily "eats" all keywords it can and doesn't let the called routine see any). This is a shame, since we can't really tell people to abandon the _EXTRA keyword altogether. At least it's shorter to type.

Anyway, hope this was clear. It's really more complicated than it could have been, but sometimes you've just got to make do.

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 |*|