Subject: Re: Object Method validity
Posted by Mark Hadfield on Tue, 12 Aug 2003 22:15:25 GMT
View Forum Message <> Reply to Message

Robert Moss wrote:

> Mark Hadfield <m.hadfield@niwa.co.nz> wrote in message
news:<bh9457ctm1@newsreader.mailgate.org>...
>
>>Robert Moss wrote:
>>
>>>Is there a way to verify the existance of a particular object method?
>>>Something like
>>>
>>>valid = Method_Valid(theObject, "HopedForMethod")
>>>
>>>I guess. I've spent a little while rtfm-ing and googling to no avail.
>>
>>I believe that the only generally valid approach is to call the method
>>and trap any errors with a CATCH statement. This has the advantage(??)
>>of actually calling the method and checking that your parameters are
>>valid. Something like this (Warning: I haven't checked it and haven't
>>generalised it to handled function-type methods)...
>>
>> catch, err
>> if err ne 0 then begin
>>   valid = 0B
>>   goto, finished
>> endif
>> call_method, theObject, 'HopedForMethod', param0
>> valid = 1B
>> catch, /CANCEL
>> finished:
>>
>>But be aware that if HopedForMethod is not bound to the class that
>>theObject belongs to, then IDL will spend a significant amount of time
>>searching the path for a file with the name
>>
>> obj_class(theObject)+'__hopedformethod.pro'
>
> That's what I was afraid of. Oh well, time for another minor feature
> request. Thanks for taking the time to answer.

Are you really trying to answer the question "Does this method exist?"
rather than "Will this method call work?"

If the latter, then I believe the "try it and see" approach is
appealing because it answers the question in the most direct way

possible. (I guess its major drawback is that it may be difficult to clean up after the method call if it fails.) In this I am influenced by my exposure to the Python language where "try it and see" is almost always the preferred approach. ("Can I read another line from this file?" "Try it and see!" "Will my database query work?" "Try it and see!" "Does this object support the functionality I want?" "Try it & see!")

If you really do want to answer the question "Does this method exist?" then you *could* try to duplicate IDL's rules for resolving methods. These are hinted at in the documentation, but there are subtleties. I suggest you do a Google Groups search for a thread in June 1998 entitled "Important object lesson". Here is an excerpt from a posting of mine in that thread:

> This is my current working hypothesis, based on a little experimentation, a modest amoung of logic, generous conjecture & a minimal scanning of the documentation...
>
> If IDL encounters
>
> MyClass->MyMethod
>
> the three situations are:
>
> 1. IDL finds a MyClass::Method in memory and uses it. (In the normal course of events the method will have been included in the myclass__define.pro, before the myclass__define procedure, so it will have been compiled the first time an instance of the class was created.) If MyClass::MyMethod is recompiled, the modifications are recognised.
>
> 2. Not finding MyClass::Method, IDL searches up the inheritance tree, finds a ASuperClass::Method in memory and uses it for the remainder of the session. If MyClass::MyMethod is recompiled, the modifications are not recognised, because this method is never called. Which is confusing.
>
> 3. Failing 1 & 2, IDL searches the !path for myclass__mymethod.pro (and maybe then for similar files for all superclasses). This can take a while. For ordinary methods, failure to find it results in an error. Obj_new and obj_destroy look for an Init and Cleanup respectively, but if they fail to find them, they just skip that step--until the next time & the next time & ...etc.

To look for methods that have already been compiled (steps 1 & 2), you can use

```
  HELP, /ROUTINES, OUTPUT=routine_list
```

and then search routine_list for HopedForMethod, following the same inheritance rules as IDL. Step 3 requires you to search the !PATH for .pro and .sav files, but if you know that methods are always stored along with class structure definitions in *__define.pro files then you can skip this.

And then there's built-in methods and DLMs...Hmmm.

But, as I said above, if the question is really "Will this method call work?" then "Try it and see!" is appealing

BTW I think my code, quoted above, was incorrect and should be

```
  catch, err
  if err ne 0 then begin
    valid = 0B
    goto, finished
  endif
  call_method, theObject, 'HopedForMethod', param0
  valid = 1B
  finished:
  catch, /CANCEL
```

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)