Subject: Re: Generating keyword parameters from strings
Posted by Paul Van Delst[1] on Wed, 22 Jul 2009 18:00:21 GMT
View Forum Message <> Reply to Message

Eric Hudson wrote:
> Hi,
>
> Does anyone know of a method of taking a string and turning it into a
> keyword parameter? As an example, say I want to write a function:
>
> function GetProperty, object, propertyName
> object->GetProperty, 'propertyName'=value ; This is pseudocode --
> this line is what I need
> return, value
> end
>
> Obviously I can write this using execute, but I'd prefer to avoid that
> if at all possible.
> Also, although I use the GetProperty method as an example, this is a
> more generic question about generating keyword parameter calls (so,
> for example, David's nice discussion of a general GetProperty method
> http://www.dfanning.com/tips/getproperty.html doesn't help here).
>
> If I were just setting a value this would be easy to do with _EXTRA by
> making my own structure. But for getting values it doesn't seem like
> there is an easy equivalent (it would be nice to just make a
> _REF_EXTRA structure if it behaved equivalent to _EXTRA but it seems
> it doesn't).

?

I use _EXTRA for my Set_Property() methods, and _REF_EXTRA for my Get_Property() methods

when I have superclasses and it seems to work fine. For example:

```
PRO SUBCLASS::Set_Property, $
  ....subclass property keywords here...
  _EXTRA = Extra  ; Keywords passed onto SUPERCLASS::Set_Property

  ...set the subclass properties....

  ; Set the superclass properties
  self->SUPERCLASS::Set_Property, _EXTRA = Extra

END
```

```
PRO SUBCLASS::Get_Property, $
  ....subclass property keywords here...
  _REF_EXTRA = Extra  ; Keywords passed onto SUPERCLASS::Get_Property

  ...get the subclass properties....

  ; Get the superclass properties
  self->SUPERCLASS::Get_Property, _EXTRA = Extra

END
```

I really don't understand why, in the SUBCLASS::Get_Property method, I can get away with using the _EXTRA keyword to the SUPERCLASS::Get_Property method, but it works. Go figure.

cheers,

paulv

Subject: Re: Generating keyword parameters from strings
Posted by David Fanning on Wed, 22 Jul 2009 18:03:59 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> I really don't understand why, in the SUBCLASS::Get_Property method, I can get away with
> using the _EXTRA keyword to the SUPERCLASS::Get_Property method, but it works. Go figure.

The rule is you use _REF_EXTRA (or _REF_STRICT_EXTRA, etc) on
the procedure or function *definition* line, but *all* extra
parameters are to be *passed* with _EXTRA.

Cheers,

David

--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Generating keyword parameters from strings
Posted by Eric Hudson on Wed, 22 Jul 2009 20:46:59 GMT

Hi,

I guess I was unclear, sorry.

What I mean is that I want to take a string and turn it into a
keyword. So continuing my above example:

```
function GetProperty, object, propertyName
; The below line is pseudocode -- what I want
  object->GetProperty, 'propertyName'=value
; using execute I'd say:
; ok = execute('object->GetProperty, '+propertyName+'=value')
; but I'm wondering if there is another way to do this
; that will work when execute is forbidden
  return, value
end

pro TestGet
  myObj = obj_new('IDLgrPalette')
  myvalue = GetProperty(myObj,'N_COLORS')
  ; Should be functionally equivalent to
  ; myObj->GetProperty, N_colors=myValue
end
```

That is, my 'GetProperty' function has to take the string 'N_COLORS'
and then call the GetProperty method using that STRING as a reference
keyword. It is this functionality that I'm interested in (I'm just
using GetProperty as an example).

Thanks,
Eric

Subject: Re: Generating keyword parameters from strings
Posted by Michael Galloy on Wed, 22 Jul 2009 22:55:06 GMT

Eric Hudson wrote:
> Hi,
>
> I guess I was unclear, sorry.
>
> What I mean is that I want to take a string and turn it into a
> keyword. So continuing my above example:
>

>
> function GetProperty, object, propertyName
> ; The below line is pseudocode -- what I want
> object->GetProperty, 'propertyName'=value
> ; using execute I'd say:
> ; ok = execute('object->GetProperty, '+propertyName+'=value')
> ; but I'm wondering if there is another way to do this
> ; that will work when execute is forbidden
> return, value
> end
>
> pro TestGet
> myObj = obj_new('IDLgrPalette')
> myvalue = GetProperty(myObj,'N_COLORS')
> ; Should be functionally equivalent to
> ; myObj->GetProperty, N_colors=myValue
> end
>
> That is, my 'GetProperty' function has to take the string 'N_COLORS'
> and then call the GetProperty method using that STRING as a reference
> keyword. It is this functionality that I'm interested in (I'm just
> using GetProperty as an example).

I don't think this is possible without a) using EXECUTE or b) writing a temporary IDL .pro file. I messed around with the REF_EXTRA keyword to SCOPE_VARFETCH, but I couldn't get what you wanted.

Mike
--
www.michaelgalloy.com
Associate Research Scientist
Tech-X Corporation

Subject: Re: Generating keyword parameters from strings
Posted by Giorgio on Thu, 23 Jul 2009 00:01:32 GMT
View Forum Message <> Reply to Message

I can image something with create_struct and call_method. But I do not know if this will work, like call_method, 'GetProperty', object, _Extra = create_struct('keyword', value)

Giorgio

Subject: Re: Generating keyword parameters from strings
Posted by Jeremy Bailin on Thu, 23 Jul 2009 12:14:17 GMT

View Forum Message <> Reply to Message

On Jul 22, 8:01 pm, Giorgio <giorgiol...@gmail.com> wrote:
> I can image something with create_struct and call_method. But I do not
> know if this will work, like call_method, 'GetProperty', object,
> _Extra = create_struct('keyword', value)
>
> Giorgio

Yeah, that's what AUGMENT_INHERITED_KEYWORD does (note that you can
give it an undefined variable to start off with, if you're not
actually augmenting an existing structure):

http://web.astroconst.org/jbiu/jbiu-doc/misc/augment_inherit ed_keyword.html

-Jeremy.

Subject: Re: Generating keyword parameters from strings
Posted by Paul Van Delst[1] on Thu, 23 Jul 2009 19:48:43 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Paul van Delst writes:
>
>> I really don't understand why, in the SUBCLASS::Get_Property method, I can get away with
>> using the _EXTRA keyword to the SUPERCLASS::Get_Property method, but it works. Go figure.
>
> The rule is you use _REF_EXTRA (or _REF_STRICT_EXTRA, etc) on
> the procedure or function *definition* line, but *all* extra
> parameters are to be *passed* with _EXTRA.

Ah, o.k. A nice, simple rule to remember.

Of course it would be nicer if I didn't have to remember it at all. Passing arguments/keywords by reference or value is an implementation detail the user shouldn't be concerned with. Having to consider it in IDL OO programming is mildly ironic.

cheers,

paulv

Subject: Re: Generating keyword parameters from strings
Posted by David Fanning on Thu, 23 Jul 2009 20:03:59 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> Of course it would be nicer if I didn't have to remember it at all. Passing
> arguments/keywords by reference or value is an implementation detail the user shouldn't be
> concerned with. Having to consider it in IDL OO programming is mildly ironic.

I think you are forgetting the 80-year history
of IDL. (Or is it 100? I can never remember.)

Cheers,

David
--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Generating keyword parameters from strings
Posted by Paul Van Delst[1] on Mon, 27 Jul 2009 14:39:41 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Paul van Delst writes:
>
>> Of course it would be nicer if I didn't have to remember it at all. Passing
>> arguments/keywords by reference or value is an implementation detail the user shouldn't be
>> concerned with. Having to consider it in IDL OO programming is mildly ironic.
>
> I think you are forgetting the 80-year history
> of IDL. (Or is it 100? I can never remember.)

Oh no, I'm not forgetting. FORTRAN (which necessarily has a longer history than IDL) had the same issue although it was a tacit understanding about passing mechanism. I assume IDL has that restriction because FORTRAN did (IIRC, wasn't IDL orginally written in f77 [or f66?] on a VMS system? But you would know more about that than me.).

Now that we have Fortran (not all caps, f90 and later) the passing mechanism can vary based on situation - the compiler can choose the better method (for a suitable definition of "better").

If you write f90+ code that depends on a particular passing mechanism it's not illegal (in many cases at least), but you're asking for chunks to be removed from your read-end in the future. It can suck for some MPI/F90+ amalgams since the former relies on passing by reference whereas the latter does not.

So, Fortran grew out of the de-facto need to be particular about passing mechanism. I see no reason why IDL can't. It certainly would make it easier for new users who wonder what

they've done wrong until they discover you can't pass arguments *out* from procedures/functions when the actual arguments in the calling routine are array elements or structure components.

cheers,

paulv

Subject: Re: Generating keyword parameters from strings
Posted by David Fanning on Mon, 27 Jul 2009 14:54:34 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> It certainly would make it easier for new users who wonder what
> they've done wrong until they discover you can't pass arguments *out* from
> procedures...

I don't think new users are meant to use direct
graphics commands. That's why they are made as
difficult to use as possible. These problems don't
exist with iTools. :-)

Cheers,

David

P.S. *Are* there any new users? I can't think of
one person using iTools, with the possible exception
of Ken Bowman, who always seems to be complaining
about them. ;-)

--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Generating keyword parameters from strings
Posted by Kenneth P. Bowman on Mon, 27 Jul 2009 21:26:03 GMT
View Forum Message <> Reply to Message

In article <MPG.24d76d9aa33164a7989836@news.giganews.com>,
David Fanning <news@dfanning.com> wrote:

> Paul van Delst writes:
>
>> It certainly would make it easier for new users who wonder what
>> they've done wrong until they discover you can't pass arguments *out* from
>> procedures...
>
> I don't think new users are meant to use direct
> graphics commands. That's why they are made as
> difficult to use as possible. These problems don't
> exist with iTools. :-)
>
> Cheers,
>
> David
>
> P.S. *Are* there any new users? I can't think of
> one person using iTools, with the possible exception
> of Ken Bowman, who always seems to be complaining
> about them. ;-)

Well, I don't teach iTools graphics when in my beginning IDL course. The students are supposed to be learning to program, not point, click, and drag. And *programming* the iTools is not for the faint of heart.

I do use iTools when I have to do 3-D interactive graphics because I am not smart enough to write my own object graphics programs.

Cheers, Ken

Subject: Re: Generating keyword parameters from strings
Posted by JDS on Thu, 30 Jul 2009 15:21:33 GMT
View Forum Message <> Reply to Message

On Jul 23, 3:48 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:
> David Fanning wrote:
>> Paul van Delst writes:
>
>>> I really don't understand why, in the SUBCLASS::Get_Property method, I can get away with
>>> using the _EXTRA keyword to the SUPERCLASS::Get_Property method, but it works. Go figure.
>
>> The rule is you use _REF_EXTRA (or _REF_STRICT_EXTRA, etc) on
>> the procedure or function *definition* line, but *all* extra
>> parameters are to be *passed* with _EXTRA.
>

> Ah, o.k. A nice, simple rule to remember.
>
> Of course it would be nicer if I didn't have to remember it at all. Passing
> arguments/keywords by reference or value is an implementation detail the user shouldn't be
> concerned with. Having to consider it in IDL OO programming is mildly ironic.

Nicety is in the eye of the beholder. At the time I proposed _REF_EXTRA and it was added to IDL 5.1, the IDL programmer who implemented it was very proud that it required only a single change to the routine definition line, not all the _EXTRA calls themselves. At the time, I objected to the duplicate interface. But I came to understand that it was an absolute requirement due to a simple but frustrating issue: various people (including those posting to this thread!) routinely build or augment their own _EXTRA structures.

If you think about it, there is no way (without inventing a new type of IDL variable) to simultaneously change the semantics of _EXTRA to allow passing by reference, and *not* break all of those sneaky codes which are poking around in _EXTRA structure themselves. So we have only ourselves to blame.

JD