Subject: Inheriting Properties (or something similar) in IDLDOC
Posted by Matt Francis on Tue, 10 Jul 2012 00:05:03 GMT
View Forum Message <> Reply to Message

I've just started a project of re-commenting a large IDL code into IDLDOC format and moving some external text file documentation into those comments.

One issue I have is that the code uses IDL custom object, and in particular inheritance, extensively. I would like the documentation for derived classes to inherit appropriate info from the parent class documentation.

In particular, the functionality of a lot of the object functions, such as threshold levels and other parameters and switches are controlled by config files. A base class may have some behaviour controlled by some 'tag' in a config file and a derived class may be controlled by additional tags. I would like a way to summarise all of the config file tags that control an object (i.e. are looked at by any of the objects functions) and then have derived classes documentation inherit the summary from the superclass.

Looking over the IDLDOC reference guide, I though perhaps the :Properties: tag in the file comments might allow this. So I have comments in the base class <obj_name>__define.pro file that look something like this:

```
;+
; Abstract Base class for handling input data for mapping.
; Provides a common interface for different data sources.
;
; :Properties:
;   data_type : property name
;     The IDL data type to use for the data storage array pointed to
by self.data
;   somthing_else :
;     Some other property
;-
```

This produces a nice summary of 'properties' in the IDLDOC produced docs. However, this does not show up the doc page for derived classes (not that I was expected to, just hoping....).

Is there any way to achieve what I'm trying to do? For comparison the way IDLDOC handles the actual class members is exactly the behaviour I want, but for a 'user defined' set of properties of objects. Is this possible?

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by Michael Galloy on Wed, 29 Aug 2012 18:14:47 GMT
View Forum Message <> Reply to Message

On 8/29/12 1:10 AM, Yngvar Larsen wrote:
> On Tuesday, 10 July 2012 05:34:20 UTC+2, Mike Galloy wrote:
>> On 7/9/12 6:05 PM, Bogdanovist wrote:
>>
>>> I've just started a project of re-commenting a large IDL code into
>>
>>> IDLDOC format and moving some external text file documentation into
>>
>>> those comments.
>>
>>>
>>
>>> One issue I have is that the code uses IDL custom object, and in
>>
>>> particular inheritance, extensively. I would like the documentation
>>
>>> for derived classes to inherit appropriate info from the parent class
>>
>>> documentation.
>>
>>>
>>
>>> In particular, the functionality of a lot of the object functions,
>>
>>> such as threshold levels and other parameters and switches are
>>
>>> controlled by config files. A base class may have some behaviour
>>
>>> controlled by some 'tag' in a config file and a derived class may be
>>
>>> controlled by additional tags. I would like a way to summarise all of
>>
>>> the config file tags that control an object (i.e. are looked at by any
>>
>>> of the objects functions) and then have derived classes documentation
>>
>>> inherit the summary from the superclass.
>>
>>>
>>
>>> Looking over the IDLDOC reference guide, I though perhaps
>>
>>> the :Properties: tag in the file comments might allow this. So I have
>>
>>> comments in the base class <obj_name>__define.pro file that look

>>
>>> something like this:
>>
>>>
>>
>>> ;+
>>
>>> ; Abstract Base class for handling input data for mapping.
>>
>>> ; Provides a common interface for different data sources.
>>
>>> ;
>>
>>> ; :Properties:
>>
>>> ;   data_type : property name
>>
>>> ;   The IDL data type to use for the data storage array pointed to
>>
>>> by self.data
>>
>>> ;   somthing_else :
>>
>>> ;   Some other property
>>
>>> ;-
>>
>>>
>>
>>> This produces a nice summary of 'properties' in the IDLDOC produced
>>
>>> docs. However, this does not show up the doc page for derived classes
>>
>>> (not that I was expected to, just hoping....).
>>
>>>
>>
>>> Is there any way to achieve what I'm trying to do? For comparison the
>>
>>> way IDLDOC handles the actual class members is exactly the behaviour I
>>
>>> want, but for a 'user defined' set of properties of objects. Is this
>>
>>> possible?
>>
>>>
>>
>>

>>
>> Ah, no, not right now. That is an interesting feature though, that would
>>
>> be quite useful. I will make a feature ticket for that.
>>
>>
>>
>> One workaround for it right now would be to create a
>>
>> my_class_properties.idldoc page that lists all the properties for
>>
>> my_class and then (manually) link to that page from all the derived classes.
>>
>>
>>
>> Mike
>
> [Sorry for the late posting. Slowly catching up on the back log from this summer.]
>
> For what it's worth: I would also like this feature. In our software, we almost in every class use inheritance from a couple of base classes, and it would be nice if the methods from these would show up in the "Routine summary" in the derived classes.
>

The trunk now shows a listing of the inherited properties in the same
manner that the inherited fields are shown. Features/fixes are
collecting, so a release is probably going to happen soon.

But, you also want a listing of inherited routines?

Mike
--
Michael Galloy
www.michaelgalloy.com
Modern IDL: A Guide to IDL Programming (http://modernidl.idldev.com)
Research Mathematician
Tech-X Corporation

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by Fabzi on Thu, 30 Aug 2012 02:33:11 GMT
View Forum Message <> Reply to Message

> The trunk now shows a listing of the inherited properties in the same
> manner that the inherited fields are shown. Features/fixes are
> collecting, so a release is probably going to happen soon.
>
> But, you also want a listing of inherited routines?

>
> Mike

It's a long time since I've not been programing Java but I remember
JavaDoc to be structured like this:

Method Summary
  Class Method1
  Class Method2
 Methods inherited from class SuperClass1
  List of methods with links to the SuperClass1 Page
 Methods inherited from class SuperClass2
  List of methods with links to the SuperClass2 Page

Method Detail
  Class Method1 in detail
  Class Method2 in detail

And more or less the same for class properties.

No need to redo Javadoc (IdlDoc is already a very great tool). But
listing properties and methods from superclasses is a cool feature,
especially for library documentation. My experience is that users that
are not familiar with objects but still need to use objects are really
unwilling to look into the whole tree to understand what's behind the
object. They just want to know what the object can do...

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by David Fanning on Thu, 30 Aug 2012 03:22:22 GMT
View Forum Message <> Reply to Message

Fab writes:

> My experience is that users that
> are not familiar with objects but still need to use objects are really
> unwilling to look into the whole tree to understand what's behind the
> object. They just want to know what the object can do...

I learned tonight from Nicholas Carr, author of "Is Google
Making Us Stupid?", that the average time people look at
a web page--ANY web page--is about 10 seconds. I wish I had
known that 15 years ago. I would have saved my self a LOT
of effort. :-(

 http://www.theatlantic.com/magazine/archive/2008/07/is-googl e-making-us-
stupid/306868/

Cheers,

David

P.S. Let's just say this lecture tonight explained a LOT!

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by Yngvar Larsen on Thu, 30 Aug 2012 06:09:50 GMT
View Forum Message <> Reply to Message

On Thursday, 30 August 2012 04:33:15 UTC+2, Fab wrote:
>> The trunk now shows a listing of the inherited properties in the same
>
>> manner that the inherited fields are shown. Features/fixes are
>
>> collecting, so a release is probably going to happen soon.
>
>>
>
>> But, you also want a listing of inherited routines?
>
>>
>
>> Mike
>
>
>
> It's a long time since I've not been programing Java but I remember
>
> JavaDoc to be structured like this:
>
>
>
> Method Summary
>
> Class Method1
>
> Class Method2
>
> Methods inherited from class SuperClass1
>

> List of methods with links to the SuperClass1 Page
>
> Methods inherited from class SuperClass2
>
> List of methods with links to the SuperClass2 Page
>
>
>
> Method Detail
>
> Class Method1 in detail
>
> Class Method2 in detail
>
>
>
> And more or less the same for class properties.

This is _exactly_ what I want.

>
> No need to redo Javadoc (IdlDoc is already a very great tool). But
>
> listing properties and methods from superclasses is a cool feature,
>
> especially for library documentation. My experience is that users that
>
> are not familiar with objects but still need to use objects are really
>
> unwilling to look into the whole tree to understand what's behind the
>
> object. They just want to know what the object can do...

And this is exactly my motivation.

--
Yngvar

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by Matt Francis on Mon, 03 Sep 2012 00:51:24 GMT
View Forum Message <> Reply to Message

I agree with the consensus here, I need to provide documentation for people who will be simultaneously learning what an 'object' is and learning how to use my specific code. Having all the information about what a class contains and can do on a single page, without needing to investigate the docs all the way up the inheritance tree, will make this learning curve much shallower for them.

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by David Fanning on Mon, 03 Sep 2012 01:01:31 GMT
View Forum Message <> Reply to Message

Bogdanovist writes:

> I agree with the consensus here, I need to provide documentation for people who will be simultaneously learning what an 'object' is and learning how to use my specific code. Having all the information about what a class contains and can do on a single page, without needing to investigate the docs all the way up the inheritance tree, will make this learning curve much shallower for them.

When I was younger, I was also an optimist. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by Michael Galloy on Mon, 03 Sep 2012 03:22:11 GMT
View Forum Message <> Reply to Message

On 9/2/12 6:51 PM, Bogdanovist wrote:
> I agree with the consensus here, I need to provide documentation for people who will be simultaneously learning what an 'object' is and learning how to use my specific code. Having all the information about what a class contains and can do on a single page, without needing to investigate the docs all the way up the inheritance tree, will make this learning curve much shallower for them.
>

I have fields and properties attached to classes so adding parent fields/properties is no problem. But I have to think about methods a bit more. It's just a bit more complicated because currently routines are attached to a File object, not directly to the Class object. What is displayed if there is only a single method in a file? What if there are multiple classes in a single file?

Mike
--

Michael Galloy
www.michaelgalloy.com
Modern IDL, A Guide to Learning IDL: http://modernidl.idldev.com
Research Mathematician
Tech-X Corporation

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by Matt Francis on Mon, 03 Sep 2012 04:01:20 GMT
View Forum Message <> Reply to Message

To me it makes sense for everything to be attached to Classes rather than source files. That's just my view though, which I think comes about because I'm looking at documentation as a way of showing other coders what tool are available in the whole package that they can use, rather than documenting the existing code for someone who wants to change it. That means I care about making it very clear how the tools currently coded work (i.e. all the info about a class in one place) rather than caring about being able to quickly and easily see exactly what source code file provides each method.

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by tom.grydeland on Mon, 10 Sep 2012 09:14:49 GMT
View Forum Message <> Reply to Message

On Monday, September 3, 2012 5:22:10 AM UTC+2, Mike Galloy wrote:
> I have fields and properties attached to classes so adding parent
> fields/properties is no problem. But I have to think about methods a bit
> more. It's just a bit more complicated because currently routines are
> attached to a File object, not directly to the Class object. What is
> displayed if there is only a single method in a file? What if there are
> multiple classes in a single file?

I think this points to a direction where the file is less important, and Classes and Routines matter more. (I suppose this echoes what Bogdanovist has said already).

You are already compiling the files and using introspection, right?

Can introspection tell you directly which methods belong to a class? If not, can you use the introspection equivalent of HELP, /ROUTINES to search for everything pertaining to a given CLASS (and its inheritance tree)? Can you use the equivalent of HELP, /SOURCE to list the file where a method is defined as part of the method description?

> Michael Galloy

--T (tom grydeland @ norut)

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by tom.grydeland on Wed, 30 Jan 2013 12:04:08 GMT
View Forum Message <> Reply to Message

On Wednesday, August 29, 2012 8:14:48 PM UTC+2, Mike Galloy wrote:

> But, you also want a listing of inherited routines?

This was exactly what we thought we would like for IDLDoc.

Given that the nicest possible way of asking for a new feature is to implement it, I present a pair of functions called SUPERCLASSES and METHOD_NAMES, and somehow I attach to them a hope for the requested feature to materialize in a future version of IDLdoc. METHOD_NAMES makes no attempt at distinguishing between function and procedure methods, but the extension to handle that should be straightforward.

> Michael Galloy

--Tom Grydeland

```
; docformat = 'rst'

;+
; :Copyright:
;   (c) 2013, Northern Research Institute Tromsø AS (Norut),
;                 All rights reserved.
;           This code can be redistributed in source and binary form
;           under the same terms as the rest of the IDLdoc system.
;
; :Requires:
;    IDL 8.0
;
; :Author: Tom Grydeland <tom.grydeland@norut.no>
;-

;+
;  List all superclasses of an object or class name, in method search order
;  (depth-first, left-to-right)
;
;  Will attempt to compile class definitions in order to find their
;  superclasses.
;
; :Params:
;   obj : required, in
;     an object, or the name of a class (a string)
;
; :Returns:
;   A string array (may be empty)
;
```

```
; :Example:
;   List all superclasses of 'gsar_reader'::
;     IDL> print, transpose(superclasses('gsar_reader'))
;-
function superclasses, obj
  compile_opt idl2, logical_predicate

  case size(obj, /type) of
    7 : classname=obj          ;string
    11 : classname=obj_class(obj)
    else: message, 'Not an object' + string(obj)
  endcase

  catch, error_status
  if error_status ne 0 then begin
    message, 'unable to resolve ' + classname, /info
    return, []
  endif

  resolve_routine, classname + '__define', /no_recompile
  super = obj_class(classname, count=nsuper, /super)

  if nsuper eq 0 then return, []

  supersuper = []
  for ii=0, nsuper-1 do begin
    supersuper = [supersuper, super[ii], superclasses(super[ii])]
  endfor

  return, supersuper
end



; docformat = 'rst'

;+
; :Copyright:
;   (c) 2013, Northern Research Institute Tromsø AS (Norut),
;                   All rights reserved.
;             This code can be redistributed in source and binary form
;             under the same terms as the rest of the IDLdoc system.
;
; :Requires:
;     IDL 8.0
;
; :Author: Tom Grydeland <tom.grydeland@norut.no>
;-
```

```
;+
;  List all methods of an object or class name, including inherited methods
;
; :Params:
;   obj: required, in
;     an object, or the name of a class (a string)
;   whence: optional, out
;     A hash that maps method name to superclass name
;
; :Returns:
;   a string array of method names
;-
function method_names, obj, whence
  compile_opt idl2, logical_predicate

  case size(obj, /type) of
    7 : classname=obj          ;string
    11 : classname=obj_class(obj)
    else: message, 'Not an object' + string(obj)
  endcase

  routines = [routine_names(/proc), routine_names(/func)]
  whence = hash()

  ;; find inherited methods and own methods
  foreach class, [reverse(superclasses(classname)), classname] do begin
    resolve_routine, class + '__define', /compile_full, /no_recompile
    ;; ii = where(strcmp(routines, super + '::', strlen(super)+2))
    foreach method, routines[where(strcmp(routines, class + '::', strlen(class)+2, /fold_case))] do
begin
      parts = strsplit(method, '::', /extract)
      if n_elements(parts) eq 2 then begin
        whence[parts[1]] = parts[0]     ; = class
      endif
    endforeach
  endforeach

  return, (whence.keys()).toArray()
end
```

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by Michael Galloy on Sun, 03 Feb 2013 22:30:00 GMT
View Forum Message <> Reply to Message

On 1/30/13 5:04 am, Tom Grydeland wrote:
> On Wednesday, August 29, 2012 8:14:48 PM UTC+2, Mike Galloy wrote:

>
>> But, you also want a listing of inherited routines?
>
> This was exactly what we thought we would like for IDLDoc.
>
> Given that the nicest possible way of asking for a new feature is to
> implement it, I present a pair of functions called SUPERCLASSES and
> METHOD_NAMES, and somehow I attach to them a hope for the requested
> feature to materialize in a future version of IDLdoc. METHOD_NAMES
> makes no attempt at distinguishing between function and procedure
> methods, but the extension to handle that should be straightforward.

OK, showing inherited methods is in the current master branch of IDLdoc.
See the output at:

http://docs.idldev.com/mglib/

In particular, see a subclass:

http://docs.idldev.com/mglib/collection/mgcoarraylist__defin e.html

This will be in the next release of IDLdoc. Let me know if you see any
problems.

Mike
--
Michael Galloy
www.michaelgalloy.com
Modern IDL: A Guide to IDL Programming (http://modernidl.idldev.com)
Research Mathematician
Tech-X Corporation

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by tom.grydeland on Wed, 06 Feb 2013 14:42:41 GMT
View Forum Message <> Reply to Message

On Sunday, February 3, 2013 10:30:00 PM UTC, Mike Galloy wrote:

> OK, showing inherited methods is in the current master branch of IDLdoc.
> See the output at:
> http://docs.idldev.com/mglib/
> In particular, see a subclass:
> http://docs.idldev.com/mglib/collection/mgcoarraylist__defin e.html
> This will be in the next release of IDLdoc. Let me know if you see any
> problems.

Basically, this looks good. I have a couple of issues:

1) Shadowed (or overridden) methods are listed. I would much prefer they weren't. Showing only the method that will be called makes browsing documentation much easier. In the currently produced documentation, you have to look through the complete method list to see which one will be called. (And are superclasses listed in inheritance order or not?) Granted, there are tasks where knowing about the shadowed methods of superclasses is useful, but I would prefer a different approach. For instance, the documentation for each method could have a small box (beneath the signature, before documentation) stating "Overrides definition from superclass idl_object", with link to the documentation for the superclass method (when applicable).

2) In the top-level listing (frames view), I followed "subclasses" links from idl_object__define to mg_graph_democlass__define. From this window, links to inherited methods did not work. (HREF=" http://docs.idldev.com/mglib/vis/objects/idl_object__define.
html#idl_object::init#idl_object::init")

> Mike

Thanks,

--T

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by Michael Galloy on Wed, 06 Feb 2013 19:10:49 GMT
View Forum Message <> Reply to Message

On 2/6/13 7:42 AM, Tom Grydeland wrote:
> Basically, this looks good. I have a couple of issues:
>
> 1) Shadowed (or overridden) methods are listed. I would much prefer
> they weren't. Showing only the method that will be called makes
> browsing documentation much easier. In the currently produced
> documentation, you have to look through the complete method list to
> see which one will be called. (And are superclasses listed in
> inheritance order or not?) Granted, there are tasks where knowing
> about the shadowed methods of superclasses is useful, but I would
> prefer a different approach. For instance, the documentation for
> each method could have a small box (beneath the signature, before
> documentation) stating "Overrides definition from superclass
> idl_object", with link to the documentation for the superclass method
> (when applicable).
>
> 2) In the top-level listing (frames view), I followed "subclasses"
> links from idl_object__define to mg_graph_democlass__define. From
> this window, links to inherited methods did not work.
> (HREF=" http://docs.idldev.com/mglib/vis/objects/idl_object__define.
html#idl_object::init#idl_object::init")

Ok, I have a fix for 2), its in the git repo now and the online docs
will be updated shortly.

1) will take a bit more thought.

Mike
--
Michael Galloy
www.michaelgalloy.com
Modern IDL: A Guide to IDL Programming (http://modernidl.idldev.com)
Research Mathematician
Tech-X Corporation

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by tom.grydeland on Mon, 11 Feb 2013 09:43:40 GMT
View Forum Message <> Reply to Message

> On 2/6/13 7:42 AM, Tom Grydeland wrote:
>> 1) Shadowed (or overridden) methods are listed. I would much prefer
>> they weren't.

On Wednesday, February 6, 2013 7:10:49 PM UTC, Mike Galloy wrote:
> 1) will take a bit more thought.

Okay, I'll bite (again).

Here is a method for DOCTREECLASS that lists a class's inherited methods, obeying inheritance order, and with methods shadowed by the class's own methods removed.

This method also exposes a bug in MGcoHashTable::values(). Whenever values had been removed so that the list for a particular bucket was empty, the expression "list->get(/all)" would return -1. This value is overwritten on the next iteration, but in my usage values are objects and -1 is not a valid value.

The fix is to replace the test on line 407 of mgcohashtable__define.pro with

```
   if (obj_valid(list) && list->count() gt 0) then begin
```

> Mike

--T

Subject: Re: Inheriting Properties (or something similar) in IDLDOC
Posted by tom.grydeland on Mon, 11 Feb 2013 09:48:59 GMT

On Monday, February 11, 2013 9:43:40 AM UTC, Tom Grydeland wrote:
> Here is a method for DOCTREECLASS that lists a class's inherited methods,

actually, _here_ it is (below).

My thought was that the getVariable method could grow a couple of new variables, such as 'n_inherited_methods' and 'inherited_methods':

```
    'inherited_methods': begin
       methods = self->getInheritedMethods()
       return, methods
     end
```

and the template 'profile.tt' etc could be modified to use these variables instead:

```
     [% IF has_class %][% FOREACH class IN classes %][% SCOPE class %]
     <h2>Inherited methods</h2>
        [% IF n_inherited_methods gt 0L %]
        [% FOREACH r IN inherited_methods %][% SCOPE r %]
                etc etc etc

        [% END %][% END %][% END %]
     [% END %][% END %][% END %]
```

but I have not been able to figure out how to make this work. I may be missing something obvious.

```
function doctreeclass::getInheritedMethods
  compile_opt strictarr, hidden

  ; First, find all methods of all ancestors in inheritance order
  inhm = obj_new('MGcoHashtable', key_type=7, value_type=11)
  n_anc = self.ancestors->count()
  for a = 0L, n_anc - 1L do begin
    anc = self.ancestors->get(position=n_anc-a-1L)
    methods = anc->getVariable('visible_methods')
    for m = 0L, n_elements(methods) - 1L do begin
      m_name = methods[m]->getVariable('name')
      parts = strsplit(m_name, '::', /extract)
      if n_elements(parts) ne 2 then continue
      inhm->put, parts[1], methods[m]
    endfor
  endfor

  ; Next, if there is a direct method of the same name, delete the inherited
  ; method
  n_meth = self.methods->count()
```

```
  if n_meth ne 0 then begin
    keys = self.methods->keys()
    for m = 0L, n_meth - 1L do begin
      parts = strsplit(keys[m], '::', /extract)
      inhm->remove, parts[1]
    endfor
  endif

  methods = inhm->values()
  ; cleanup
  return, methods
end
```