Subject: Destroying objects
Posted by Michael Wallace on Wed, 11 Feb 2004 22:26:44 GMT
View Forum Message <> Reply to Message

I have just started working with the IDL-Java Bridge and object programming in general and have a very basic question. Is there a way to destroy objects that get created which you haven't explicitly created yourself? Here's a simple example of the problem:

```
; Method 1
; Get the number of columns of a Java ResultSet

cols = (result_set -> getMetaData()) -> getColumnCount()
obj_destroy, result_set

help, /heap  ; shows an object associated with ResultSetMetaData
             ; but how do you destroy it??


; Method 2
; Get the number of columns of a Java ResultSet

meta_data = result_set -> getMetaData()
cols = meta_data -> getColumnCount()
obj_destroy, meta_data
obj_destroy, result_set

help, /heap  ; doesn't show any extra objects
```

So, is there a way to use Method 1, but still have a way to destroy the extra object that gets created? There's no variable name associated with it, so how can I use obj_destroy, if at all? Will I be forced to go by Method 2 and use only one '->' per line? I was so hopeful I could be lazy and use multiple '->' per line.... *sigh*

Mike

Subject: Re: Destroying objects
Posted by David Fanning on Mon, 16 Feb 2004 18:18:44 GMT
View Forum Message <> Reply to Message

Andy Meigs writes:

> Starting to work on objects as well using Martin Schlutz's mgs_objects base
> class system-- mighty frustrating when 'errors' (probably not of Martin's,

> but of my ignorance in how to use his code) occur several superclasses from
> the class you have written.

Yes, well, one of the things we *have* done right in our
library is implement excellent error handling. The problem
we had originally was cascading error messages. Something
would happen way down deep, and the message would propagate
up the entire object chain. It sounded like one of those
fire alarms you can't turn off. :-(

Very disconcerting to new users of the library.

We figured out a way to indicate that the error was already
"handled" so objects further up the chain could just pass it
silently. Users get one error, and the traceback points to
the right place 99.9% of the time. Note that this is a
different technique from handing the *error* silently, which
is what the iTool library does. Using the latter method, you can
be frustrated for weeks on end, since it takes four or five
times as long to realize you *have* an error than it does
to fix it. :-)

Cheers,

David

P.S. Part of my frustration last week turned out to be--don't
you know--a...uh, programmer error. I've got to get the ol'
Sherlock Holmes manual out again, but I'm pretty sure one of
the first rules is "check your assumptions". :-(

--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Destroying objects
Posted by JD Smith on Mon, 16 Feb 2004 19:55:05 GMT
View Forum Message <> Reply to Message

On Wed, 11 Feb 2004 16:26:44 -0600, Michael Wallace wrote:

> I have just started working with the IDL-Java Bridge and object
> programming in general and have a very basic question. Is there a way
> to destroy objects that get created which you haven't explicitly created
> yourself? Here's a simple example of the problem:
>

>
> ; Method 1
> ; Get the number of columns of a Java ResultSet
>
> cols = (result_set -> getMetaData()) -> getColumnCount()
> obj_destroy, result_set
>
> help, /heap ; shows an object associated with ResultSetMetaData
>   ; but how do you destroy it??
>

> So, is there a way to use Method 1, but still have a way to destroy the
> extra object that gets created? There's no variable name associated
> with it, so how can I use obj_destroy, if at all? Will I be forced to
> go by Method 2 and use only one '->' per line? I was so hopeful I could
> be lazy and use multiple '->' per line.... *sigh*

Not pretty, but:

cols = ((meta = result_set -> getMetaData())) -> getColumnCount()
obj_destroy,meta,result_set

should work. Then again, you have to wonder about returning objects
whose only purpose is to report a value and then get destroyed. I
should think something like:

result_set->GetMetaData,COLUMN_COUNT=cols

would be a much cleaner syntax, and not employ any intermediary
objects to keep track of.

JD

Subject: Re: Destroying objects
Posted by Michael Wallace on Mon, 16 Feb 2004 20:46:47 GMT
View Forum Message <> Reply to Message

> cols = ((meta = result_set -> getMetaData())) -> getColumnCount()
> obj_destroy,meta,result_set
>
> should work. Then again, you have to wonder about returning objects
> whose only purpose is to report a value and then get destroyed. I
> should think something like:
>
> result_set->GetMetaData,COLUMN_COUNT=cols
>
> would be a much cleaner syntax, and not employ any intermediary

> objects to keep track of.

Yes, I agree that the above is much cleaner. In my example, I was using the Java database connectivity classes directly. Since I only have to use a subset of the available Java SQL classes and methods, I wrote a simple Faï¿½ade for the classes and methods I needed. This ensures that only the most necessary objects are created in IDL and simplifies the access overall. I can definitely see myself using the Faï¿½ade pattern quite a bit in the coming days....

Mike

Subject: Re: Destroying objects
Posted by JD Smith on Mon, 16 Feb 2004 21:11:56 GMT
View Forum Message <> Reply to Message

On Mon, 16 Feb 2004 11:18:44 -0700, David Fanning wrote:

> Andy Meigs writes:
>
>> Starting to work on objects as well using Martin Schlutz's mgs_objects base
>> class system-- mighty frustrating when 'errors' (probably not of Martin's,
>> but of my ignorance in how to use his code) occur several superclasses from
>> the class you have written.
>
> Yes, well, one of the things we *have* done right in our
> library is implement excellent error handling. The problem
> we had originally was cascading error messages. Something
> would happen way down deep, and the message would propagate
> up the entire object chain. It sounded like one of those
> fire alarms you can't turn off. :-(
>
> Very disconcerting to new users of the library.
>
> We figured out a way to indicate that the error was already
> "handled" so objects further up the chain could just pass it
> silently. Users get one error, and the traceback points to
> the right place 99.9% of the time. Note that this is a
> different technique from handing the *error* silently, which
> is what the iTool library does. Using the latter method, you can
> be frustrated for weeks on end, since it takes four or five
> times as long to realize you *have* an error than it does
> to fix it. :-)

I wonder if you can expand on this error handling technique a bit.
One feature I like to have is the ability for errors to "just work",
regardless of if the object is being used on the command-line, or via

its GUI. In the latter case, errors should be trapped and displayed in a pop-up. In the former case, they should call MESSAGE to print to the command line and exit. I've managed it so far by having a super-class which implements Error, Warning, and other methods, which checks to see if you are in a command-line invocation or not, and uses message or dialog_message as appropriate. Suitable use of CATCH allows deep errors in other routines further down the stack (ones you didn't write, for instance) to be handled in the same way, but that requires some care to "set the trap" when necessary. This is somewhat similar to the way XManager can catch and disregard errors (as it does by default), but with reporting, and valid even when XManager isn't running.

JD

Subject: Re: Destroying objects
Posted by David Fanning on Mon, 16 Feb 2004 21:22:02 GMT
View Forum Message <> Reply to Message

M. Katz writes:

> One main source of heartburn in objects comes from passing parameters
> through _Extra and _Ref_Extra. I use them all the time and am still
> confused by them.

I'm not too unhappy with keyword inheritance. In general, I think it works quite well. I've finally figured out _EXTRA and _REF_EXTRA, I think. When I am expecting results back from keywords (for example, all GetProperty methods), I *always* use _REF_EXTRA on the method definition line, but I *always* use _EXTRA when passing the extra keywords along to the superclass method. Somehow, someway, this *always* works! I've got object templates now that help me remember these simple things, so when I'm building a new object, I'm more likely to get it right. (Amazing how much templates helped with memory leakage problems, too.)

A bigger problem initially was passing keywords along that weren't defined anywhere and not knowing about it. So we use _STRICT_EXTRA on the "atom" class object, which is inherited by everyone, to trap unhandled keywords. In other words, if a keyword gets to this level and it hasn't been defined yet, someone used the wrong keyword (or misspelled it, more likely).

Occasionally, I have more trouble with this, but not too often, and most of the time it is easily solved by just

defining a few more keywords.

> All this talk of objects and pointers is making me think of the IDL
> vs. Matlab threads that were going on 2 months back. For all the pros
> and cons on the syntax and speed, how great is it that IDL allows us
> to use object and pointers! to dream these great abstractions (and
> spend our weekends debugging.)

Well, Amen to this, I guess... :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Destroying objects
Posted by David Fanning on Mon, 16 Feb 2004 22:06:35 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> I wonder if you can expand on this error handling technique a bit.
> One feature I like to have is the ability for errors to "just work",
> regardless of if the object is being used on the command-line, or via
> its GUI. In the latter case, errors should be trapped and displayed
> in a pop-up. In the former case, they should call MESSAGE to print to
> the command line and exit. I've managed it so far by having a
> super-class which implements Error, Warning, and other methods, which
> checks to see if you are in a command-line invocation or not, and uses
> message or dialog_message as appropriate. Suitable use of CATCH
> allows deep errors in other routines further down the stack (ones you
> didn't write, for instance) to be handled in the same way, but that
> requires some care to "set the trap" when necessary. This is somewhat
> similar to the way XManager can catch and disregard errors (as it does
> by default), but with reporting, and valid even when XManager isn't
> running.

I would have thought you knew me well enough by now, JD,
to realize that if I don't immediately provide the IDL
code to solve a problem I'm either trying to get rich (ha!)
or the solution is so simpleminded I prefer to not risk
certain embarrassment. Alas, it is the latter in this case. :-(

We don't do anything fancy. All our methods contain standard

error handler code in them, added with a @error_handler
sort of call. (We originally had one code fragment because I figured
out a really slick way to tell if I was in a procedure or function
and I could either return something or not, but the method
used an EXECUTE function. When we wanted to run on the VM,
we had to go to two error handlers: one for procedures
and one for functions. What a shame!) The error handler
simply sets up a CATCH for the method. If an error is caught,
it is dispatched to our ERROR method, which is attached to
the "atom" object all objects inherit.

When we get an error we can handle in our error handler,
we add a "handled" to the front of the error message.
(I told you this was simpleminded.) As the error propagates
back up the object chain, if the error handler sees a "handled"
in front of the message it returns immediately.

I can't think of a way to tell if the object is running
at the IDL command line or not (I've shown you mine, you
should show me yours), but our error handler can write to
a log file (if there is one) or to the display. Typically,
we issue a pop-up dialog (DIALOG_MESSAGE) with a short
error message, and write a better formatted traceback of
the error in the command output log of the display.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Destroying objects
Posted by JD Smith on Tue, 17 Feb 2004 20:30:20 GMT
View Forum Message <> Reply to Message

On Mon, 16 Feb 2004 15:06:35 -0700, David Fanning wrote:

> JD Smith writes:
>
>> I wonder if you can expand on this error handling technique a bit.
>> One feature I like to have is the ability for errors to "just work",
>> regardless of if the object is being used on the command-line, or via
>> its GUI. In the latter case, errors should be trapped and displayed
>> in a pop-up. In the former case, they should call MESSAGE to print to
>> the command line and exit. I've managed it so far by having a

>> super-class which implements Error, Warning, and other methods, which
>> checks to see if you are in a command-line invocation or not, and uses
>> message or dialog_message as appropriate. Suitable use of CATCH
>> allows deep errors in other routines further down the stack (ones you
>> didn't write, for instance) to be handled in the same way, but that
>> requires some care to "set the trap" when necessary. This is somewhat
>> similar to the way XManager can catch and disregard errors (as it does
>> by default), but with reporting, and valid even when XManager isn't
>> running.
>
> I would have thought you knew me well enough by now, JD,
> to realize that if I don't immediately provide the IDL
> code to solve a problem I'm either trying to get rich (ha!)
> or the solution is so simpleminded I prefer to not risk
> certain embarrassment. Alas, it is the latter in this case. :-(
>
> We don't do anything fancy. All our methods contain standard
> error handler code in them, added with a @error_handler
> sort of call. (We originally had one code fragment because I figured
> out a really slick way to tell if I was in a procedure or function
> and I could either return something or not, but the method
> used an EXECUTE function. When we wanted to run on the VM,
> we had to go to two error handlers: one for procedures
> and one for functions. What a shame!) The error handler
> simply sets up a CATCH for the method. If an error is caught,
> it is dispatched to our ERROR method, which is attached to
> the "atom" object all objects inherit.
>
> When we get an error we can handle in our error handler,
> we add a "handled" to the front of the error message.
> (I told you this was simpleminded.) As the error propagates
> back up the object chain, if the error handler sees a "handled"
> in front of the message it returns immediately.
>
> I can't think of a way to tell if the object is running
> at the IDL command line or not (I've shown you mine, you
> should show me yours), but our error handler can write to
> a log file (if there is one) or to the display. Typically,
> we issue a pop-up dialog (DIALOG_MESSAGE) with a short
> error message, and write a better formatted traceback of
> the error in the command output log of the display.

Interesting method. I would worry that putting an explicit CATCH in each method might bog things down, if your events typically make deeply nested excursions through dozens of routines (as mine often do). On the other hand IDL's native routine-based error-propagation is fast, but inflexible. I have a few methods for getting the best of both worlds by putting a top-level CATCH-based handler at some

high-level, but this is complicated by non-blocking apps.

Since I'm not planning to get rich off of it, I don't mind showing you my simple method for discriminating between command-line and GUI. I require inheriting classes to override a "ReportWidget" method, like this:

```
 ;=============================================================
=================
;  ReportWidget - Return the widget to test the validity of to decide
;               between graphical and text-based errors/warnings.
;               Should be overridden.  By default, looks for a class
;               tag "wInfo" which is a ptr to an info structure with
;               a "Base" tag.
 ;=============================================================
=================
function ObjReport::ReportWidget
  catch, err
  if err ne 0 then begin
     catch,/cancel
     message, "No class tag pointer `wInfo' with tag `Base' found.  " + $
            "Override the ReportWidget Method?"
  endif
  if ptr_valid(self.wInfo) then return,((*self.wInfo).Base)
  return,-1L
end
```

As it turns out, I often use a detachable self.wInfo field for all widget info, and the top level base is often attached to a "Base" tag inside of that, so I don't typically need to override this, and the default works fine. Then testing for the presence of the GUI is a simple matter of:

```
 ;=============================================================
=================
;  isWidget - Do we use a widget for popup messages?
 ;=============================================================
=================
function ObjReport::IsWidget
  self.or_widget=self->ReportWidget() ;re-get the widget, it could be changing
  return,widget_info(self.or_widget,/VALID_ID)
end
```

JD

Subject: Re: Destroying objects
Posted by David Fanning on Tue, 17 Feb 2004 21:19:33 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> Interesting method. I would worry that putting an explicit CATCH in
> each method might bog things down, if your events typically make
> deeply nested excursions through dozens of routines (as mine often
> do).

I don't know about speed. Those errors seem to happen instantaneously
to me. :-(

Lord knows I'm not going through "dozens" of routines, typically.
I have enough trouble keeping everything in mind with relatively
flat object hierarchies. Six or eight levels deep is a fairly deep
traceback from most of the programs I write. But it takes next to
no time to go into a routine and back out of it again (which is what
happens, essentially, to a handled message), so I don't notice a
time penalty at all.

> Since I'm not planning to get rich off of it, I don't mind showing you
> my simple method for discriminating between command-line and GUI.

Interesting. My library is built for application building, which
pretty much means widgets from the get-go, so we just assume
a widget is around and pop-up dialogs on errors. If I had it
to do over, though, I might think about abstracting the
widget functionality (this is one thing the iTool library
does nicely, but at a huge cost in complexity) and separating
the object from the widget itself. Or, better yet, just have
RSI marry the widget and object functionality so we don't
have to do it. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Destroying objects
Posted by chris_torrence@NOSPAM on Fri, 18 Jan 2013 17:12:48 GMT
View Forum Message <> Reply to Message

On Friday, January 18, 2013 9:33:43 AM UTC-7, Helder wrote:
> Hi,
>
> as usual I'm do things wrong believing I'm not until I wake-up.
>
> So here's the thing. I'm using a structure to describe some objects. In my case these are regions of interest. Within this structure I have real objects. So that my structure looks something like this:
>
>
>
> MyStruct = {ROI_Structure, a:ID_Nr, b:SomeOtherParameter, c:Obj_new()}
>
> I then fill in MyStruct.c by doing something like this:
>
> XROI, MyImage, REGIONS_OUT = XROI_ObjRef
>
> MyStruct.c = XROI_ObjRef
>
>
>
> The thing is that I then replicate this structure when I add new ROIs.
>
> So I end up with say 10 {ROI_Structure} where for each element I did something like this:
>
> for i=0,9 do MyStruct[i].c = XROI_ObjRef[i]
>
>
>
> If and when somewhere later in the program I want to delete object number 5... do I have to take care of using
>
> Obj_destroy, MyStruct[5].c
>
>
>
> Is there no alternative?
>
> Thing is, in this structure I have many pointers and objects...
>
>
>
> I'm I wrong in saying that I have to do the good-boy bookkeeping and free/destroy for each deleted structure element all pointers and objects?
>
>
>
> Thanks,

>
> Helder

Hi Helder,

If you are using IDL 8.0 or later, then as long as you don't have any "circular" references, the automatic garbage collection should free all of the pointers & objects once the variables go out of scope.

However, if you have circular references, where one object references another object, and that other object somehow references the first one, then the garbage collector won't be able to decrement the reference counts to zero.

Another possibility is to use HEAP_GC, which is a big hammer...

Cheers,
Chris
ExelisVIS

Subject: Re: Destroying objects
Posted by David Fanning on Fri, 18 Jan 2013 17:50:05 GMT
View Forum Message <> Reply to Message

Helder writes:

> as usual I'm do things wrong believing I'm not until I wake-up.
> So here's the thing. I'm using a structure to describe some objects. In my case these are regions of interest. Within this structure I have real objects. So that my structure looks something like this:
>
> MyStruct = {ROI_Structure, a:ID_Nr, b:SomeOtherParameter, c:Obj_new()}
> I then fill in MyStruct.c by doing something like this:
> XROI, MyImage, REGIONS_OUT = XROI_ObjRef
> MyStruct.c = XROI_ObjRef
>
> The thing is that I then replicate this structure when I add new ROIs.
> So I end up with say 10 {ROI_Structure} where for each element I did something like this:
> for i=0,9 do MyStruct[i].c = XROI_ObjRef[i]
>
> If and when somewhere later in the program I want to delete object number 5... do I have to take care of using
> Obj_destroy, MyStruct[5].c
>
> Is there no alternative?
> Thing is, in this structure I have many pointers and objects...
>
> I'm I wrong in saying that I have to do the good-boy bookkeeping and free/destroy for each

deleted structure element all pointers and objects?

I'm not sure what the problem is, Helder. Doesn't this, do the job:

   Obj_Destroy, myStruct.c[*]

Cheers,

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

Subject: Re: Destroying objects
Posted by Helder Marchetto on Fri, 18 Jan 2013 18:07:55 GMT
View Forum Message <> Reply to Message

On Friday, January 18, 2013 6:50:05 PM UTC+1, David Fanning wrote:
> Helder writes:
>
>
>
>> as usual I'm do things wrong believing I'm not until I wake-up.
>
>> So here's the thing. I'm using a structure to describe some objects. In my case these are regions of interest. Within this structure I have real objects. So that my structure looks something like this:
>
>>
>
>> MyStruct = {ROI_Structure, a:ID_Nr, b:SomeOtherParameter, c:Obj_new()}
>
>> I then fill in MyStruct.c by doing something like this:
>
>> XROI, MyImage, REGIONS_OUT = XROI_ObjRef
>
>> MyStruct.c = XROI_ObjRef
>
>>
>
>> The thing is that I then replicate this structure when I add new ROIs.
>
>> So I end up with say 10 {ROI_Structure} where for each element I did something like this:
>
>> for i=0,9 do MyStruct[i].c = XROI_ObjRef[i]

>
>>
>
>> If and when somewhere later in the program I want to delete object number 5... do I have to take care of using
>
>> Obj_destroy, MyStruct[5].c
>
>>
>
>> Is there no alternative?
>
>> Thing is, in this structure I have many pointers and objects...
>
>>
>
>> I'm I wrong in saying that I have to do the good-boy bookkeeping and free/destroy for each deleted structure element all pointers and objects?
>
>
>
> I'm not sure what the problem is, Helder. Doesn't this, do the job:
>
>
>
> Obj_Destroy, myStruct.c[*]
>
>
>
> Cheers,
>
>
>
> David
>
> --
>
> David Fanning, Ph.D.
>
> Fanning Software Consulting, Inc.
>
> Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
>
> Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Thank to both.
I think this is a general problem I'm realizing about programming. If you use structures that contain pointers or objects, that might contain other pointer and objects and so on, you have to take care

of cleaning up the mess you've made.

Will have to do some housekeeping or trust 8.2.1 do to it for me.

Cheers,
Helder

Subject: Re: Destroying objects
Posted by David Fanning on Fri, 18 Jan 2013 18:23:13 GMT
View Forum Message <> Reply to Message

Helder writes:

> I think this is a general problem I'm realizing about programming. If you use structures that contain pointers or objects, that might contain other pointer and objects and so on, you have to take care of cleaning up the mess you've made.
>
> Will have to do some housekeeping or trust 8.2.1 do to it for me.

If you write CLEANUP routines for your objects, it really doesn't seem
to be much of a job to me.

Cheers,

David

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