Subject: Re: Objects with Widgets, Save/Restore
Posted by John-David T. Smith on Fri, 03 Aug 2001 16:55:47 GMT
View Forum Message <> Reply to Message

Brent Griffith wrote:
> I am using a handful of home-made objects (thanks to RCK's book) in a
> application with lots of widgets. I am trying to use SAVE and RESTORE
> to get my objects back to where they were in a previous run of the
> application. I have it sort of working but would appreciate seeing
> some examples of an "object's restore method".
>
> For example, is there an elegant way to deal with the widget IDs and
> values? I am storing widgetIDs in the object for widget_control
> statements, but the newly created widgets generally have different IDs
> than the old Saved-object. I find myself repeatedly writing the same
> statements for each and every widget that might get its value changed
> -- see example statements below (* and 1 indicate multiplicity).

You might check older postings of mine which comment on this very topic ad naseum. The basic idea is to detach those portions of the object data structure which aren't relevant to a saved version, before saving. This requires a bit of organization, but is pretty easy. Here are a few messages I found regarding this method:

http://groups.google.com/groups?selm=361AC6B3.436DB6E%40astr osun.tn.cornell.eduhttp://groups.google.com/groups?selm=3847 128F.B532A27F%40astro.cornell.edu
http://groups.google.com/groups?selm=37B4F36B.E1D4BCEC%40ast rosun.tn.cornell.edu

Here's an example, excerpted from a complicated class of mine:

```
***************
  ;; Detach the stuff we don't want to save!
  detInfo=self.wInfo & self.wInfo=ptr_new() ;don't save the info
  detMsgList=self.MsgList & self.MsgList=ptr_new() ;or the message list
  ;; Here we detach the data to avoid duplication, if they want it.
  if self.SpaceSaver eq 1b then begin
    detDatas=(*self.DR).DATA  ;or the data
    (*self.DR).DATA=ptrarr(self->N_Records())
    detTotals=(*self.DR).TOTAL ;or the totals
    (*self.DR).TOTAL=ptrarr(self->N_Records())
  endif else self->RestoreAll

  oldchange=self.Changed       ;we want the file written to have
changed=0!
  self.Changed=0b              ;but save the old preference incase we
```

```
fail
  catch, serr
  if serr ne 0 then begin      ;it failed!
    catch,/CANCEL
    self.wInfo=detInfo        ;reattach them
    self.MsgList=detMsgList
    if self.SpaceSaver eq 1b then begin ;reconnect the data
      (*self.DR).DATA=detDatas
      (*self.DR).TOTAL=detTotals
    endif
    self.Changed=oldchange    ;reassign our old changed status
    wmessage,PARENT_GROUP=pg, $
     'Error Saving to File: '+pname
    return
  endif
  save,self,FILENAME=pname
  catch,/CANCEL
  ;; Reattach
  self.wInfo=detInfo
  self.MsgList=detMsgList
  if self.SpaceSaver eq 1b then begin
    (*self.DR).DATA=detDatas
    (*self.DR).TOTAL=detTotals
  endif
***************
```

Notice the error handling, especially. You don't want to lose your running widget hierarchy if the save fails. I basically remove the entire widget info structure (which is nicely organized under a single pointer) among a few other things, and then on restoration, I can rebuild the widgets there again... something like:

```
***************
  ;; make the info structure if we need it.
  if NOT ptr_valid(self.wInfo) then begin
    self.wInfo=ptr_new({scoreProj_wInfo})
  endif
  self.wInfo.base=widget_base(...
***************
```

You probably get the point.

Good luck,

JD

Subject: Re: Objects with Widgets, Save/Restore
Posted by John-David T. Smith on Fri, 03 Aug 2001 16:56:49 GMT
View Forum Message <> Reply to Message

Sorry, formatting hiccup.

http://groups.google.com/groups?selm=361AC6B3.436DB6E%40astr osun.tn.cornell.edu

http://groups.google.com/groups?selm=3847128F.B532A27F%40ast ro.cornell.edu

http://groups.google.com/groups?selm=37B4F36B.E1D4BCEC%40ast rosun.tn.cornell.edu

JD

Subject: Re: Objects with Widgets, Save/Restore
Posted by Pavel A. Romashkin on Fri, 03 Aug 2001 16:57:45 GMT
View Forum Message <> Reply to Message

I usually write a method for the object that will then create
appropriate gui and populate object fields with the right widget IDs. I
also usually place the object reference to restored object into the
Uvalue of one of the newly created widgets. Then, you simply do

```
restore, my_saved_obj
self -> restore_gui
```

and you are done. You can chuck Self after this because it will be
accessible via the widget anyway.
Cheers,
Pavel

Subject: Re: Objects with Widgets, Save/Restore
Posted by david[2] on Fri, 03 Aug 2001 17:09:05 GMT
View Forum Message <> Reply to Message

Brent Griffith writes:

> I am using a handful of home-made objects (thanks to RCK's book) in a
> application with lots of widgets. I am trying to use SAVE and RESTORE
> to get my objects back to where they were in a previous run of the
> application. I have it sort of working but would appreciate seeing
> some examples of an "object's restore method".
>
> For example, is there an elegant way to deal with the widget IDs and
> values? I am storing widgetIDs in the object for widget_control
> statements, but the newly created widgets generally have different IDs

> than the old Saved-object. I find myself repeatedly writing the same
> statements for each and every widget that might get its value changed

Yes, this can be a problem. Widgets, of course, receive a unique ID each time the creation routine is run.

I'm not sure I have a sure-fire solution for you, although I also have a large widget program with lots of objects running, and I can save and restore the current session. At least I can after a fashion.

I guess the secret is to have some kind of GUI method for the object widget that builds the interface. I find this has to be separate from the INIT method. This allows you to build a new interface from the restored object, thus getting all your widget identifiers sorted out again.

I played with it for days trying to keep the current GUI (which, of course, has the RESTORE button on it), but no matter what I tried (and I thought I tried some pretty sophisticated things) I couldn't get it to work. So now when you hit the RESTORE button, the GUI disappears and comes back a second later with the old session up and ready to go.

Cheers,

David

P.S. Let's just say I've had to learn to appreciate momentary window flashing. But it is only me. My customer has never cared two figs one way or the other. He's just thrilled he can pick up where he left off for lunch yesterday! :-)

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Objects with Widgets, Save/Restore
Posted by Pavel A. Romashkin on Fri, 03 Aug 2001 17:14:44 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> I played with it for days trying to keep the
> current GUI (which, of course, has the RESTORE
> button on it)

Now *this* got me puzzled. Why would you want a Restore button on an existing GUI?! I thought a Restore is for when you save the work and exit IDL, then come back ant start everything from where you left it!

Cheers,
Pavel

Subject: Re: Objects with Widgets, Save/Restore
Posted by david[2] on Fri, 03 Aug 2001 17:20:18 GMT
View Forum Message <> Reply to Message

I wrote a minute ago:

> I played with it for days trying to keep the
> current GUI (which, of course, has the RESTORE
> button on it), but no matter what I tried (and
> I thought I tried some pretty sophisticated
> things) I couldn't get it to work. So now when
> you hit the RESTORE button, the GUI disappears
> and comes back a second later with the old
> session up and ready to go.

After reading JD's post, I'm pretty sure there is a way to get this to work. In fact, I believed that for the 3-4 frustrating days I spent working on the problem. I *still* believe it. I just can't get it to work. :-(

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Objects with Widgets, Save/Restore
Posted by david[2] on Fri, 03 Aug 2001 17:29:03 GMT
View Forum Message <> Reply to Message

Pavel A. Romashkin writes:
>
> Now *this* got me puzzled. Why would you want a Restore button on an
> existing GUI?! I thought a Restore is for when you save the work and
> exit IDL, then come back ant start everything from where you left it!

I think you are thinking of the SAVE button, Pavel. :-)

Cheers,

David

P.S. But I do have a neat little thing that
asks you if you want to save the current session
before you restore another one, but only if you
have actually done some work. Drinking coffee and
staring at the screen doesn't count.

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Objects with Widgets, Save/Restore
Posted by John-David T. Smith on Fri, 03 Aug 2001 20:30:35 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> I wrote a minute ago:
>
>> I played with it for days trying to keep the
>> current GUI (which, of course, has the RESTORE
>> button on it), but no matter what I tried (and
>> I thought I tried some pretty sophisticated
>> things) I couldn't get it to work. So now when
>> you hit the RESTORE button, the GUI disappears
>> and comes back a second later with the old
>> session up and ready to go.
>
> After reading JD's post, I'm pretty sure

> there is a way to get this to work. In fact,
> I believed that for the 3-4 frustrating days
> I spent working on the problem. I *still*
> believe it. I just can't get it to work. :-(
>

The secret is this same notion of pruning out the bits you don't need.
Here's a sketch of how I'd approach your problem.

```
pro DavidsPlayhouse::Restore
  oldself=self              ;for killing later
  wInfo=self.wInfo          ;save the GUI, eat soy products
  self.wInfo=ptr_new()      ;detach, avoiding the carnage
  restore_obj, self.SaveFile ;travel back in time
  obj_destroy,oldself       ;kill our old self, except wInfo
  self.wInfo=wInfo          ;reattach our saved widget info
  self->UpdateGUI           ;I'm not who I think I am
end
```

where "restore_obj" restores an object on top of itself. At least I think this is what you're trying to do. I once called this "transmogrification" ... to the delight and ridicule of various newsgroup regulars. It relies on the fact that the variable "self" is actually passed by reference to all methods, so you can overwrite it simply by restoring from file an object previously saved under a variable name "self" (try "self=1" in a method sometime and see what trouble you'll get yourself into). Hopefully, this is an object of the same type, or you'll be in for some surprises. Notice how I killed my old self (after carefully detaching the GUI components I wanted to escape the Cleanup carnage), to avoid having a split personality (and memory leaks).

The secret to keeping the same GUI running across this restore process is simple: isolate all widget state info under a pointer, detach it beforehand, and retain it across the restore, for grafting onto the newly reanimated object. As long as all widget ID's and state information survive the (wait for it) transmogrification, you won't have any GUI flashing, since it'll be the same damn widgets running all throughout (yes, including the button which invoked the command). I'd also isolate the restore code in some error checking. I can send you an entire example if you're interested. And by the way, to avoid the confusion Pavel identified, I call this feature "Revert from Disk...", as in "Go back to that version I saved before I did that stupid thing". It's very handy. Do also consult the restore_obj writeup on your own website for some of the potential snafu's associated with object restoration.

Good luck,

JD

Subject: Re: Objects with Widgets, Save/Restore
Posted by Pavel A. Romashkin on Fri, 03 Aug 2001 20:55:33 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
>
> Do also consult the restore_obj writeup on your own
> website

Now, this was cruel! On the other hand, David's website is sooo big, no
wonder he can forget about all the pieces :-)
Cheers,
Pavel

Subject: Re: Objects with Widgets, Save/Restore
Posted by david[2] on Fri, 03 Aug 2001 21:07:25 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> The secret is this same notion of pruning out the bits you don't need.
> Here's a sketch of how I'd approach your problem.
>
> pro DavidsPlayhouse::Restore
> oldself=self ;for killing later
> wInfo=self.wInfo ;save the GUI, eat soy products
> self.wInfo=ptr_new() ;detach, avoiding the carnage
> restore_obj, self.SaveFile ;travel back in time
> obj_destroy,oldself ;kill our old self, except wInfo
> self.wInfo=wInfo ;reattach our saved widget info
> self->UpdateGUI ;I'm not who I think I am
> end

Humm. Since I'm within hours of delivering
the first truly production version of this
code, I'm loath to open up this topic again.

But I think now what got me into trouble
is that inside my larger application, which
is written as an object, there was an
ROI Window Object. This is a compound widget
(window) that doesn't have any notion of what's

inside it, but it can draw various types of
ROI's on itself. This too is written as an object,
and it reports its results (perimeter points,
interior indices, etc.) to the event method handler
of the parent widget object when the ROI
is completely drawn. Of course, it relies on
widget identifiers to communicate with its
"parent" properly. And it has it's own widget
identifiers to deal with.

It seems to me that I could not get this
interior widget object fired up correctly.
Although it seems to me now (as then) that
the approach you outline above should work in
both cases. And, of course, that is what I
thought I was doing.

But when the customer is paying you big bucks
and is already happy with the design, it seems
the height of arrogance to spend yet more time
on niggling cosmetic problems. Especially when
the other solution was so easy (it took me about
10 minutes to implement). Even my overly
obsessive compulsion for neatness can be
bought at the right price. :-)

But I'll have another go next time. I'm convinced
it has to work. Maybe when I figure it out I'll
have enough material for that object book.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Objects with Widgets, Save/Restore
Posted by John-David T. Smith on Fri, 03 Aug 2001 21:38:38 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" wrote:
>

> JD Smith wrote:
>>
>> Do also consult the restore_obj writeup on your own
>> website
>
> Now, this was cruel! On the other hand, David's website is sooo big, no
> wonder he can forget about all the pieces :-)
> Cheers,
> Pavel

I felt somewhat justified since restore_obj was my routine anyway. I
know I have to look back at the site about every 4 times I try object
restoration.

JD

Subject: Re: Objects with Widgets, Save/Restore
Posted by david[2] on Fri, 03 Aug 2001 22:27:02 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> I felt somewhat justified since restore_obj was my routine anyway. I
> know I have to look back at the site about every 4 times I try object
> restoration.

Yeah, people sometimes make the mistake of thinking
I maintain that web page for them. If I didn't have
a place I could go and read about something 10-15 times
I would probably never learn anything new about IDL. :-(

Cheers,

David

P.S. Let's just say I put all that information on the
web so I could look half-way intelligent as I talk
to customers in my travels around the world. You didn't
think I read novels at night, did you?

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Objects with Widgets, Save/Restore
Posted by John-David T. Smith on Fri, 03 Aug 2001 22:27:33 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> JD Smith writes:
>
>> The secret is this same notion of pruning out the bits you don't need.
>> Here's a sketch of how I'd approach your problem.
>>
>> pro DavidsPlayhouse::Restore
>> oldself=self ;for killing later
>> wInfo=self.wInfo ;save the GUI, eat soy products
>> self.wInfo=ptr_new() ;detach, avoiding the carnage
>> restore_obj, self.SaveFile ;travel back in time
>> obj_destroy,oldself ;kill our old self, except wInfo
>> self.wInfo=wInfo ;reattach our saved widget info
>> self->UpdateGUI ;I'm not who I think I am
>> end
>
> Humm. Since I'm within hours of delivering
> the first truly production version of this
> code, I'm loath to open up this topic again.
>
> But I think now what got me into trouble
> is that inside my larger application, which
> is written as an object, there was an
> ROI Window Object. This is a compound widget
> (window) that doesn't have any notion of what's
> inside it, but it can draw various types of
> ROI's on itself. This too is written as an object,
> and it reports its results (perimeter points,
> interior indices, etc.) to the event method handler
> of the parent widget object when the ROI
> is completely drawn. Of course, it relies on
> widget identifiers to communicate with its
> "parent" properly. And it has it's own widget
> identifiers to deal with.
>

Probably you'd just have to use this method to do some more aggressive pruning of your (obviously more complex) class structure. If your objects contains objects which themselves are quite tricky, you might consider a "Prune" and "Reattach" method in each, which readies an object for saving by clipping out the sensitive widget portions, and puts them back in place upon restoration. The question of course becomes, where do you put that sensitive information (or, more probably, the pointer to it), in the meantime. You can't put it any place in the

object itself, since it is about to be overwritten from disk!

In my simple solution, all pruning was controlled from a single calling level, and it was enough for me to save everything in a method local pointer variable (wInfo) to survive the restore. In other situations, in which objects contain objects which contain objects, all of which will potentially be saved to disk in a big fat file, you have to get more clever.

I would seriously consider something like a system variable or a common block for this rare operation, but if the real paranoia even those words instill in us cannot be overcome, you could always collect up the pointers and propogate them upwards to the top-level Prune method, which does the restore, and then pushes them back down through the stack of Reattach methods. If you keep everything in order between Prune and Reattach, you should be able to pair up all the correct wInfo's (etc.) with all the correct objects. The best structure for this would of course be a pointer tree, the creation of which is a rewarding exercise in its own right. The main object would create the head of the tree -- a node of form, say

```
 superinfo={INFO_TREE, sibling:ptr_new(), child:ptr_new(),$
    info:ptr_new()}
```

This would create and hand down superinfo.child (with its sibling pointer suitably set), to the first child, and etc., all the way down. The giant info tree could be lifted up over the restoration of the nested objects, and then pushed back down through the tree in the same fashion for reattachment. Voila.

In fact, now that I think of it, there's no reason there couldn't be a generic superclass SaveRestore which hides this tree logic, and makes it simple for objects to perform this dance. Probably not in time for you deadline though ;)

JD

Subject: Re: Objects with Widgets, Save/Restore
Posted by Pavel A. Romashkin on Fri, 03 Aug 2001 22:59:13 GMT
View Forum Message <> Reply to Message

By now I'd bet all David's object widgets have the same superclass, so all you basically need is a method for the superclass to prune and reattach, and the superclass probably could have the same basic widget id storage facility. If not, a simple search method can be implemented in the superclass' methods. And then all that seems necessary is a pointer, which will be local to every instance of call to that method,

even if called in a nested order on subclasses. This way it certainly
would be possible to avoid the curse of globalizing anything in an application.
Cheers,
Pavel
P.S. Oh god. I read this after writing. God forbid a non-IDL person to
reading this stuff. They will report us to a psychiatric hospital :-(

Subject: Re: Objects with Widgets, Save/Restore
Posted by david[2] on Fri, 03 Aug 2001 23:02:06 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> The main object would create the head of the tree --
> a node of form, say
>
> superinfo={INFO_TREE, sibling:ptr_new(), child:ptr_new(),$
>   info:ptr_new()}
>
> This would create and hand down superinfo.child (with its sibling
> pointer suitably set), to the first child, and etc., all the way down.
> The giant info tree could be lifted up over the restoration of the
> nested objects, and then pushed back down through the tree in the same
> fashion for reattachment. Voila.

Geez, I was just browsing my web page and
I found a LINKEDLIST object! Just the thing. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Objects with Widgets, Save/Restore
Posted by david[2] on Fri, 03 Aug 2001 23:38:08 GMT
View Forum Message <> Reply to Message

Pavel A. Romashkin writes:

> By now I'd bet all David's object widgets have the same superclass, so
> all you basically need is a method for the superclass to prune and
> reattach, and the superclass probably could have the same basic widget
> id storage facility.

Actually, this is the problem. My objects are not superclassed. My object application contains other objects which (occasionally) contain other objects. As you know (and anyone who reads this newsgroup knows) one of the great attractions of objects is that the data can be encapsulated inside them, making them exceedingly self-contained. Or, as I like to call them, smart.

This is good ... most of the time.

But, occasionally, and always at the worst possible moment, you wish they were a little more leaky (I'm making up these technical terms as I go). That is to say, you find yourself sitting in some object method scratching your head and thinking to yourself that you sure wish you could have a peek at that object over *there*, because then life would be a whole lot easier.

But, alas, that peek requires a new method and after you have added three or four of these you begin to suspect there is a better way here, if only you had been smart enough to see it three weeks ago. :-(

But, too late, and on you go. And before you know if you find yourself in the mornings drinking an extra cup of coffee and lingering over The Onion on the web, avoiding the inevitable trip into that morass that used to be a pretty neat program.

It all turns out in the end, of course, because you are a decent programmer and you know (sorta) what you are doing. But for the life of you, you can't tell whether object programming is more efficient, or just more fun. All you really know is that there are a few more gray hairs every morning when you look in the mirror. :-(

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Objects with Widgets, Save/Restore
Posted by Pavel A. Romashkin on Mon, 06 Aug 2001 16:29:17 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> That is to say, you find yourself sitting
> in some object method scratching your head and
> thinking to yourself that you sure wish you could
> have a peek at that object over *there*, because
> then life would be a whole lot easier.

This is where you create a superclass. It doesn't need to even have much in the way of data structure, maybe one string field "Goal: 'Made to fix my older objects'". All it will take is inserting "inherits" into your existing multiple classes, and restarting IDL. And that superclass will have "spying" methods sharable by all your existing objects, with unique names of course. Also, *only* this superclass will have "prune-reattach" methods used by the older objects.

> All you really know
> is that there are a few more gray hairs every
> morning when you look in the mirror. :-(

Now *that* is not IDL-related. I just start cutting hair shorter :-)

Cheers,
Pavel
