Subject: Object Widgets
Posted by Bernard Puc on Fri, 05 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Hello,

What's the current status of writing object widgets? I've been in DejaNews reading Mark Rivers' method of objectifying a widget program and Reinhold Schaff's outlines of CWidget. Is anybody writing objectified simple widgets? I'd like to develop some widget programs which can be inherited and modified by subclasses.

--
Bernard Puc AETC, INC.
bpuc@va.aetc.com 1225 Jefferson Davis Highway #800
(703) 413-0500 Arlington, VA 22202

Subject: Re: Object Widgets
Posted by J.D. Smith on Mon, 15 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Struan Gray wrote:
>
> Struan Gray, struan.gray@sljus.lu.se writes:
>
> Whoops. Bad formatting in my last post.
>
> This:
>
>> objwidref = obj_new('object_widget') objwidref -> Start
>
> Should read:
>
>> objwidref = obj_new('object_widget')
>> objwidref -> Start
>
>
> Struan

I often do all sorts of things in the Start method. Usually they are along the lines of setting up variables (like window id's), that don't yet exist until a subsidiary object is Started. Also for drawing to widget_draw's, and similar things which require the widgets already to have been started up. But it's not just for widget issues.

For instance, I have a color manager object which other object widgets inquire for various colors to set themselves up. Until this object is initialized, it

makes no sense to inquire things of it. If there were only one object utilizing it... no problem -- just make sure it's initted first. But when many objects being initted in many different places might be involved, this is *much* easier.

Another nice thing about having a standard method "Start" is that a controlling class can automatically "Start" all of it's composited objects, without knowing the details of what they're doing (be it widget or otherwise).

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|

Subject: Re: Object Widgets
Posted by Pavel Romashkin on Tue, 16 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Martin Schultz wrote:

> probably the best way to initialize the object is to call its
> own SetProperty method from the Init method

It seems very logical. In fact this is what they do at RSI, take a look at their __define procedures in the object examples directory.

Cheers,
Pavel

Subject: Re: Object Widgets
Posted by Struan Gray on Tue, 16 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Martin Schultz, m218003@modell3.dkrz.de writes:

[calling SetProperty methods from Init]

> Any thoughts about this?

You are running the keyword check twice, which might slow you down if you have a lot of objects to create (I've played with making crystal structures containing tens of thousands of atom-objects), it

will also run checks on any properties that you don't want (or need) to set at startup time.

If only to avoid spaghetti-ising the code, I would prefer to have seperate copies of the simple assignments in your example. If setting a property takes more than a up few lines I put it in its own private method that is called from both Init and SetProperty. This probably has more to do with the way my mind works than any real-world efficiency.

Struan

Subject: Re: Object Widgets
Posted by Struan Gray on Tue, 16 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

J.D. Smith, jdsmith@astro.cornell.edu writes:

> I often do all sorts of things in the Start method.

Having thought about it a bit I can see several advantages to this idea. Part of me still likes the idea of invoking a working widget with a single call, but I expect I'll get over it. Incidentally, if you feel like showing the public your generalised messaging class I for one would be very interested.

Struan

Subject: Re: Object Widgets
Posted by m218003 on Tue, 16 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

In article <38306F7E.F5371DDE@astro.cornell.edu>,
"J.D. Smith" <jdsmith@astro.cornell.edu> writes:
>
> I often do all sorts of things in the Start method. Usually they are
> along the
> lines of setting up variables (like window id's), that don't yet exist
> until a
> subsidiary object is Started. Also for drawing to widget_draw's, and similar
> things which require the widgets already to have been started up. But it's
> not
> just for widget issues.

>
> For instance, I have a color manager object which other object widgets
> inquire
> for various colors to set themselves up. Until this object is initialized,
> it
> makes no sense to inquire things of it. If there were only one object
> utilizing
> it... no problem -- just make sure it's initted first. But when many objects
> being initted in many different places might be involved, this is *much*
> easier.
>
> Another nice thing about having a standard method "Start" is that a
> controlling
> class can automatically "Start" all of it's composited objects, without
> knowing
> the details of what they're doing (be it widget or otherwise).
>

The last couple of days I experimented a litle with a hierarchy of objects
defining a rectangular box (superclass), a page (one subclass branch)
and a [plotting] frame (another subclass branch). Halfway through it occured
to me that probably the best way to initialize the object is to call its
own SetProperty method from the Init method (of course you need to make
sure that the inherited methods are called as well and you must set
default values for all possible keywords). I then recalled that this is what
was proposed in onebook about OOP that I read (wait - 8 years? ago).
Upon second thought, this method (pun intended) of course only works
for the public properties of your object, i.e. those that are "visible"
to the SetProperty method. The advantage of this approach is that you
don't need to check your keywords twice for correctness.

Here's a quick example:

```
pro bogus::SetProperty, afloat=afloat, anintarr=anintarr

  ; make sure arguments are correct
  if n_elements(afloat) eq 1 then self.afloat=float(afloat)
  if n_elements(anintarr) gt 0 then self.anintarr = fix(anintarr)

end


function bogus::Init, afloat=afloat, anintarr=anintarr

  ; need only set default values here
  if n_elements(afloat) eq 0 then afloat = 0.
  if n_elements(anintarr) eq 0 then anintarr = intarr(1)
```

```
  self -> SetProperty,afloat=afloat, anintarr=anintarr

end
```

Any thoughts about this?

Regards,
Martin

--
[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[

Subject: Re: Object Widgets
Posted by J.D. Smith on Tue, 16 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Struan Gray wrote:
>
> J.D. Smith, jdsmith@astro.cornell.edu writes:
>
>> I often do all sorts of things in the Start method.
>
>   Having thought about it a bit I can see several advantages to this
> idea. Part of me still likes the idea of invoking a working widget
> with a single call, but I expect I'll get over it.   Incidentally, if
> you feel like showing the public your generalised messaging class I
> for one would be very interested.
>
> Struan

Do to demand I have put a tiny ObjMsg page up at:

  http://www.astro.cornell.edu/staff/jdsmith/objmsg/objmsg.htm l

It's just a skeleton of a page, and, as it mentions, without examples and
details, ObjMsg doesn't seem to have much utility. Basically, ObjMsg is just a
set of conventions for method extensions and some convenient housekeeping. The
real meat of this paradigm exists in inheriting classes, some of which I hope to

clean up and post at some point. For instance, the "Start" method convention doesn't even exist at this high level, but originates one level down in a so-called broker class. It's not really as complicated as it sounds.

Anyway, comments are welcome.

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|

Subject: Re: Object Widgets
Posted by J.D. Smith on Wed, 17 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Struan Gray wrote:
>
> J.D. Smith, jdsmith@astro.cornell.edu writes:
>
>> http://www.astro.cornell.edu/staff/jdsmith/objmsg/objmsg.htm l
>>
>> Anyway, comments are welcome.
>
> Nice stuff. Thanks for the sneak peek.
>
> I have something similar (though I got lazy and subclassed from
> IDL_Container rather than write my own list manager), and am still
> muddling over whether to make the actual messages objects as well.
>
> At some point one has to start deciding how the message should be
> handled by the receiver. It looks to me as if you do that by
> modifying the ObjMsg::Message method, which implies that all the
> entities participating in the message passing have to be objects and
> have to have a Messsage method.
>
> For older widgets which I can't be bothered to objectise (fewer
> and fewer with time) I created a Gossip object which exists solely to
> turn a call like
>
> sendlist[i]->Message, msg
>
> in ObjMsg::MsgSend into a suitable widget event and put it in the
> event queue. The old-style widget then processes it with a
> conventional old-style event loop.

>
>   This would obviously work in your scheme too, but I extended (or,
> more accurately, am in the process of extending) the gossip object
> idea so that even objects and object widgets generate gossip objects
> to handle messaging (they don't *have* to of course). This means an
> object can effectively have different Message methods for
> communicating with different objects. I would need to see how you
> create and hendle your msg structures, but I get the feeling I
> seperate the behaviour of the list manager and list items more
> explicitly than you do.
>
>   As an example: I have a generic IDLgrModel viewer which generates
> sub-widgets in their own TLBs to control settings for the trackball,
> the viewing position, colours, etc. Instead of a single Message
> method which looks at fields (or properties) of the message to decide
> where it's coming from and what to do, I have several gossip objects
> which already know which sub-widget they are dealing with.
>
>   As is often the case in OOP, all I'm really doing is avoiding a
> lot of IF or CASE statements in a single, big routine (in this case, a
> general message handler). It's a style issue, but I find this easier
> to get my head round, a little easier to adapt to specific cases, and
> a tiny bit faster as it cuts down on the need for Message methods to
> run detailed checks on every possible event type.

The Message methods of all object derived from ObjMsg are called *only* for those messages for which they have signed up (where the details of "signing up", including what information you sign up, are left to inheriting classes). This is handled by the sender object in the method MsgSendWhich. Objects are only sent the messages they want, which is typically quite few, so nothing like the long if-then-else-case jungles seen in most *event* handlers are required.

This is the key distinction between the object message paradigm and the standard widget event paradigm. It's sort of like David's general method of separate event handlers for different sources of events, but can be reconfigured dynamically during run-time, on both the sending an receiving ends, and associations aren't determined by widget hierarchy, but by the programmer, using whatever structuring he finds convenient. Granted the skeletal ObjMsg doesn't fully demonstrate that functionality, which is, as you guessed, developed by properly extending ObjMsg methods. The doc header suggest what to override/extend and how.

I guess the basic difference is one of organization. You seem to favor many specialty methods for handling "messages" (though you can't really call them that anymore when there's no standard for delivery). I typically make objects that service just some small part of an application, and receive only a few messages necessary for that functionality. I divide by making separate objects, as opposed to single objects with separate methods. This makes possible the use

of objects which, thanks to having a standard shared set of interface elements, can be accessed with the minimum of fuss. One big idea using this stuff which I think is within reach is an image viewing application that supports easy to write, easy to exchange, easy to use "plug-ins", so that you can custom tailor an environment that works for you. Need statistics? Plug in a stats module. Need Convolution? Convolve module? Aperture Photometry? etc... That was the original thinking, though it hasn't played out fully yet. It's still not quite easy enough to write them (though using is pretty simple). Here is an excerpt from an end-user program which puts together some of the plug-in tools for use:

```
;; A slicer object, green
slicer=obj_new('tvSlice',tvD,COLOR=stretcher->GetColor('Green'), $
            /EXCLUSIVE)

;; a zoom in tool, green
zoomer=obj_new('tvZoom',tvD,/EXCLUSIVE,  $
            COLOR=stretcher->GetColor('Green'), _EXTRA=e)

;; a histogram tool, red, with stretcher as its color object.
hist=obj_new('tvHist',tvD,COLOR=stretcher->GetColor('Red'),/EXCLUSIVE, $
          /CORNERS,COLOBJ=stretcher,_EXTRA=e)

;; a base for our selector buttons
sbase=widget_base(base,/ROW,/FRAME,SPACE=1)

;; a box statistics tool, yellow
stats=obj_new('tvStats',base,tvD,COLOR=stretcher->GetColor('Yellow'), $
            /EXCLUSIVE,/CORNERS,/HIDE,HANDLE=2,_EXTRA=e)
```

That's basically all thats required. No need to setup message flow, etc., since that's handled inately in the ObjMsg objects themselves.

An interesting idea about a reverse compatibility object. I think a Gossip object would fit nicely into this framework.

What I really think could use some work in my stuff is the way in which messages, and the functionality they represent, can be advertised and subscribed to. Currently, this is not extensible enough. I wanted to be free to do this in different ways for different projects, which is why ObjMsg doesn't contain specifics, but now I'm realizing the utility of some reuseable paradigm.

Anyway, food for thought.

JD
--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875

Ithaca, NY 14853 |*|

Subject: Re: Object Widgets
Posted by Struan Gray on Wed, 17 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

J.D. Smith, jdsmith@astro.cornell.edu writes:

> http://www.astro.cornell.edu/staff/jdsmith/objmsg/objmsg.htm l
>
> Anyway, comments are welcome.

Nice stuff. Thanks for the sneak peek.

I have something similar (though I got lazy and subclassed from IDL_Container rather than write my own list manager), and am still muddling over whether to make the actual messages objects as well.

At some point one has to start deciding how the message should be handled by the receiver. It looks to me as if you do that by modifying the ObjMsg::Message method, which implies that all the entities participating in the message passing have to be objects and have to have a Messsage method.

For older widgets which I can't be bothered to objectise (fewer and fewer with time) I created a Gossip object which exists solely to turn a call like

```
sendlist[i]->Message, msg
```

in ObjMsg::MsgSend into a suitable widget event and put it in the event queue. The old-style widget then processes it with a conventional old-style event loop.

This would obviously work in your scheme too, but I extended (or, more accurately, am in the process of extending) the gossip object idea so that even objects and object widgets generate gossip objects to handle messaging (they don't *have* to of course). This means an object can effectively have different Message methods for communicating with different objects. I would need to see how you create and hendle your msg structures, but I get the feeling I seperate the behaviour of the list manager and list items more explicitly than you do.

As an example: I have a generic IDLgrModel viewer which generates sub-widgets in their own TLBs to control settings for the trackball, the viewing position, colours, etc. Instead of a single Message

method which looks at fields (or properties) of the message to decide where it's coming from and what to do, I have several gossip objects which already know which sub-widget they are dealing with.

As is often the case in OOP, all I'm really doing is avoiding a lot of IF or CASE statements in a single, big routine (in this case, a general message handler). It's a style issue, but I find this easier to get my head round, a little easier to adapt to specific cases, and a tiny bit faster as it cuts down on the need for Message methods to run detailed checks on every possible event type.

Struan

Subject: Re: Object Widgets
Posted by Struan Gray on Thu, 18 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

J.D. Smith, jdsmith@astro.cornell.edu wrote:

> I guess the basic difference is one of organization.

Agreed. Once you go beyond sending the same message to every subscriber, a decision on which messages to send to whom has to be made somewhere. Exactly where doesn't matter much until you start trying to share code with someone who has done it differently.

> You seem to favor many specialty methods for handling "messages"
> (though you can't really call them that anymore when there's
> no standard for delivery).

My actual messages tend to have a common organisation (an extension of the must-have fields in an event structure. Recipients know that there will be a field called 'data' for example, and use the structure name or the widget/object id of the sender to decide what to do with it. Using intermediate gossip objects (which I do subclass madly) does the weeding out that you do at the list manager stage.

> One big idea using this stuff which I think is within
> reach is an image viewing application that supports easy to
> write, easy to exchange, easy to use "plug-ins", so that you
> can custom tailor an environment that works for you.

This is one of my major motivations. I don't want to have to rewrite a spectral deconvolution widget because a user wants to try a new lineshape.

> What I really think could use some work in my stuff is the
> way in which messages, and the functionality they represent,
> can be advertised and subscribed to. Currently, this is not
> extensible enough.

This is where I'm bogged down. I have some tentative preference manager objects which hand out and set preferences for all sort of things, and I've been trying to find an elegant way for running widgets to say 'tell me when I should change my background colour'. Cludges are easy, but making it general and extensible is proving harder than I thought.

> Anyway, food for thought.

Feeling slightly sick, Peter Rabbit went to look for some parsley...

Struan

Subject: Re: object widgets
Posted by Pavel A. Romashkin on Fri, 11 Jan 2002 06:23:25 GMT
View Forum Message <> Reply to Message

They got so popular, everyone calls them their own names nowdays.
Did you try www.dfanning.com?
Or has David been cut off from his Dedicated Satellite Link? :)
try
http://www.mpimet.mpg.de/~schultz.martin/idl/index.html
Cheers,
Pavel

"Marc Schellens" <m_schellens@hotmail.com> wrote in message news:3C3F13E9.EA191A30@hotmail.com...
> Does anybody know what happened to Martin Schulz
> object widgets?
> When he introduced them I was to busy to have a closer look.
> Yesterday I wanted but could not find them.
> Is there a 'standard' way for object widgets so far?
>
> thanks,
> marc

Subject: Re: object widgets
Posted by btt on Fri, 11 Jan 2002 14:05:50 GMT
View Forum Message <> Reply to Message

Marc Schellens wrote:

>
> Thanks, I looked there.
> But I could not find any object widgets.
> Which are these?
> Probably I miss the forest because of to much trees...
>
> :-) marc

Hi,

Try this one
http://www.mpimet.mpg.de/~schultz.martin/idl/html/libmgs_obj ects.html

Ben

--
Ben Tupper
Bigelow Laboratory for Ocean Science
180 McKown Point Road
West Boothbay Harbor, ME 04575
www.bigelow.org
btupper@bigelow.org

Subject: Re: object widgets
Posted by David Fanning on Fri, 11 Jan 2002 14:57:57 GMT
View Forum Message <> Reply to Message

Pavel Romashkin (pavel.romashkin@noaa.gov) writes:

> Or has David been cut off from his Dedicated Satellite Link? :)

No, no. But I did find my New Year's Resolutions list.
The first item is this:

1. Get a life! Give those poor folks on the Newsgroup a break!

I'm exercising restraint this year. :-)

There hasn't been too much written down about object widgets
yet, although I understand that may be changing soon. But

basically, these objects are written with some kind of
GUI method to create the interface and the self object is
the former info structure that holds the program information
required to run the program.

The trick, if there is one, is to get widget events
into object methods where they can access the self
information. I usually do this by placing in the UValue
of those widgets which will generate events I want to
respond to a "message" structure. This anonymous
structure contains an object reference (typically,
but not always to the "self") and the name of an
event handler method to be called on the object.

Here is part of a GUI method from a program I
wrote for a client:

```
  fileID = Widget_Button(menuID, Value='File')
  button = Widget_Button(fileID, Value='Acquire New Image...', $
    UValue={object:self, method:'AcquireNewImage'})
  button = Widget_Button(fileID, Value='Open Existing Image...', $
    UValue={object:self, method:'OpenExistingImage'})
  button = Widget_Button(fileID, Value='Restore Session', $
    UValue={object:self, method:'RestoreSession'}, /Separator)
  button = Widget_Button(fileID, Value='Save Session', $
    UValue={object:self, method:'SaveSession'})
  button = Widget_Button(fileID, Value='Quit', $
    UValue={object:self, method:'QuitProgram'}, /Separator)
```

The event handler for *all* the events generated by the program
is very simple. Here it is:

```
  PRO COMPANY_ANALYSIS_EVENT, event
  Widget_Control, event.id, Get_UValue=message
  Call_Method, message.method, message.object, event
  END
```

The event handler simply gets the "message" stored in
the UValue of the widget causing the event, and calls
the appropriate method for that object, using Call_Method.

The event handler methods, then, are written *exactly*
like your former widget event handlers. Here is an example
of one:

```
  PRO COMPANY_ANALYSIS::SaveSession, event

  ; This method saves the current session.
```

```
    ; Error handling.

  Catch, theError
  IF theError NE 0 THEN BEGIN
    Catch, /Cancel
    ok = Error_Message(Traceback = self.debug)
    RETURN
  ENDIF

  GetName:
  filename = Dialog_Pickfile(Title='Save Session As...',  $
   File='petrograhic_session.pws', $
   Filter='*.pws', Path=self.directory, /Write)
  IF filename EQ "" THEN RETURN

  checkFile = FindFile(filename, Count=count)
  IF count EQ 1 THEN BEGIN
    answer = Dialog_Message('File exists. OK to overwrite?', /Question)
    IF StrUpCase(answer) EQ 'YES' THEN Save, self, File=filename ELSE $
      GoTo, GetName
  ENDIF ELSE Save, self, File=filename
  END
```

Please ignore the GOTO statement. I think it was 3:30 in the
morning when I wrote this particular piece of lousy code. :-)

That's about it. Pretty simple. But extraordinarily powerful.

Cheers,

David

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

Subject: Re: object widgets
Posted by Pavel A. Romashkin on Fri, 11 Jan 2002 19:55:38 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> There hasn't been too much written down about object widgets

> yet, although I understand that may be changing soon. But
> basically, these objects are written with some kind of
> GUI method to create the interface and the self object is
> the former info structure that holds the program information
> required to run the program.

I have been thinking for a while (after seeing some still present
hesitation to use those widgects) that one link is missing there. I
think I'l try to fix that. It will be an object class called OW_Ai,
standing for Object widget artificial intelligence, to be inhereted by widgects.
What it will do is track down the coordinates and context of those
multiple random mouse click that a user is making while trying in
frustration to debug his code. Then, it will attempt to deduce what did
the user want (because most of the time when one is simply asked, he
can't give a concise reply). Then, it will "transmogrify" the widgect in
question accordingly, reset session and return to the application with
everything now working, much to developers' surprise.
I will post the code as soon as it is less than 100 lines of code. I
don't want Marc to blame me for too long piece of code again :)
Cheers,
Pavel

Subject: Re: object widgets
Posted by btt on Fri, 11 Jan 2002 20:06:29 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" wrote:

> I have been thinking for a while (after seeing some still present
> hesitation to use those widgects) that one link is missing there. I
> think I'l try to fix that. It will be an object class called OW_Ai,
> standing for Object widget artificial intelligence, to be inhereted by widgects.
> What it will do is track down the coordinates and context of those
> multiple random mouse click that a user is making while trying in
> frustration to debug his code. Then, it will attempt to deduce what did
> the user want (because most of the time when one is simply asked, he
> can't give a concise reply). Then, it will "transmogrify" the widgect in
> question accordingly, reset session and return to the application with
> everything now working, much to developers' surprise.
>

Pavel,

I don't suppose you would consider adding a spell-chucker, would you?

Ben
--

Ben Tupper
Bigelow Laboratory for Ocean Science
180 McKown Point Road
West Boothbay Harbor, ME 04575
www.bigelow.org
btupper@bigelow.org

Subject: Re: object widgets
Posted by Pavel A. Romashkin on Fri, 11 Jan 2002 20:15:05 GMT
View Forum Message <> Reply to Message

Ben Tupper wrote:
>
> Pavel,
>
> I don't suppose you would consider adding a spell-chucker, would you?
>
> Ben

He-ey, what are you implying? Gim'me a break. IDL is my third language,
English being the second. And for us critters from the outer space your
human languages are all pretty hard.
So I might add a spill chunker for those too observant :)

Chairs,
Pavel

Subject: Re: object widgets
Posted by btt on Fri, 11 Jan 2002 20:23:37 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" wrote:

>
>
> He-ey, what are you implying? Gim'me a break. IDL is my third language,
> English being the second. And for us critters from the outer space your
> human languages are all pretty hard.
> So I might add a spill chunker for those too observant :)
>
>

I knew you could take it!

It's really remarkable, English is my second language and I don't seem to

know any others!

Seriously, how about adding an auto-documentation method?

Ben

--
Ben Tupper
Bigelow Laboratory for Ocean Science
180 McKown Point Road
West Boothbay Harbor, ME 04575
www.bigelow.org
btupper@bigelow.org

Subject: Re: object widgets
Posted by Pavel A. Romashkin on Fri, 11 Jan 2002 21:57:52 GMT
View Forum Message <> Reply to Message

Ben Tupper wrote:
>
> Seriously, how about adding an auto-documentation method?

No need for stinkin' documentation. Those smart ones who visit this NG will be able to read IDL just as they would plain English, and the dumb ones don't need the functionality to begin with :)

Cheers,
Pavel

Subject: Re: object widgets
Posted by marc schellens[1] on Fri, 11 Jan 2002 22:20:24 GMT
View Forum Message <> Reply to Message

Pavel Romashkin wrote:
>
> They got so popular, everyone calls them their own names nowdays.
> Did you try www.dfanning.com?
> Or has David been cut off from his Dedicated Satellite Link? :)
> try
> http://www.mpimet.mpg.de/~schultz.martin/idl/index.html
> Cheers,
> Pavel

Thanks, I looked there.

But I could not find any object widgets.
Which are these?
Probably I miss the forest because of to much trees...

:-) marc

Subject: Re: object widgets
Posted by David Fanning on Thu, 03 Mar 2005 05:55:31 GMT
View Forum Message <> Reply to Message

Marc Schellens writes:

> I remeber there was a discussion here about widgets wrapped
> in objects some year ago, but didn't follow it till the end.
> Has there now a de-facto standard library evolved for this?

If you mean by "de-facto standard" no documentation, then
I suppose so:

http://www.dfanning.com/tips/catlib.html

Cheers,

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

Subject: Re: object widgets
Posted by marc schellens[1] on Thu, 03 Mar 2005 07:00:07 GMT
View Forum Message <> Reply to Message

I had already a look at it, and it seems to be exactly what I want but
is at least some raw documentation available? Or the sourcecode?

De-facto standard meant something almost everbody here uses.

Cheers,
marc

Subject: Re: object widgets
Posted by David Fanning on Thu, 03 Mar 2005 07:47:12 GMT

Mark Schellens writes:

> I had already a look at it, and it seems to be exactly what I want but
> is at least some raw documentation available? Or the sourcecode?

The code itself is extensively documented and mostly accurate.
I take great pains to keep it up-to-date as much as I can.
In fact, objects can document themselves. What is not available
is a User's Guide, with the big picture of how these 80 plus
objects can be used with one another. Programmers who use
the library currently don't seem to need much more than what
is already there, but I am about to teach the library to
two people who have never programmed in IDL before. We will
see how much hand-holding they are going to need. :-)

I think if you are comfortable with objects generally, I can
explain in a couple of hours how the whole thing works. There
are some tricks, ways of doing things, that are important, but
these are almost all illustrated in the example program that
comes with the library. Our mantra while building the library
was "simple and easy". "Even scientists should be able to
use it," we kept telling ourselves. You don't have to know
anything at all about object graphics to use the library
unless you really, really want to. But 95% of what we build,
we build with good ol' direct graphics.

Like anything else, you have to build about three programs
with the library and then pretty much everything is trivial.
It helps to have some knowledge of widget programming, generally,
but working with widgets is greatly simplified. Pretty much
everything you want to do with a widget occurs with GetProperty
and SetProperty methods. Communication between objects is
simple and acts very much like widget event handling.

We are currently having some issues getting the code
to work perfectly on all platforms. We push pretty
hard on some things, and we have the usual cross-platform
headaches that come with any but the simplest programs.
(If you have ever looked at any of the code supplied
with IDL, you know we are not the only ones running into
these kinds of problems.)

Around here the source code is known as Fanning's Folly
because it was suppose to be a Gold Mine and it has
turned into something of an albatross. The bottom line
is this: we took two years to build the library, we
know we can build applications with it, so we presume

it might have some commercial value for someone other
than us. It's a quixotic point of view, but we are
sticking with it, at least for now. So, yes, you can
see the source code, but it will probably cost you. :-)

> De-facto standard meant something almost everbody here uses.

Around here is means whatever happens to be at hand and
almost works. :-)

Cheers,

David

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

Subject: Re: object widgets
Posted by marc schellens[1] on Thu, 03 Mar 2005 09:12:02 GMT
View Forum Message <> Reply to Message

> Around here the source code is known as Fanning's Folly
> because it was suppose to be a Gold Mine and it has
> turned into something of an albatross. The bottom line
> is this: we took two years to build the library, we
> *know* we can build applications with it, so we presume
> it might have some commercial value for someone other
> than us. It's a quixotic point of view, but we are
> sticking with it, at least for now. So, yes, you can
> see the source code, but it will probably cost you. :-)

What means probably around your place? :-)

Important is that it is a de-facto standard ie. the code is "at hand"
(now).
For an evaluation, the source code of the examples without
the library itself would be fine I guess. Maybe packed with one or
two of the classes.

marc
