Subject: Re: IDLgrLegend Property Sheets, array properties
Posted by sdettrick on Tue, 20 Apr 2004 18:18:18 GMT
View Forum Message <> Reply to Message

I hate to reply to my own message but it has occurred to me that I can just register each element of the ITEM_NAME and each triplet of the ITEM_COLOR as a seperate property with a seperate name (e.g. ITEM_NAME0, ITEM_NAME1, etc). In principle I guess the following would work ( I say "in principle", because so far I can't get it to work)

```
; Register ARRAY properties element-by-element
for i=0,n_elements( item_names )-1 do begin
  self -> IDLitComponent::RegisterProperty, $
              'ITEM_NAME'+strtrim(i,2),/STRING
  self -> IDLitComponent::RegisterProperty, $
              'ITEM_COLOR'+strtrim(i,2), /COLOR

  self -> IDLitComponent::SetPropertyByIdentifier, $
              'ITEM_NAME'+strtrim(i,2), item_names[i]
  self -> IDLitComponent::SetPropertyByIdentifier, $
              'ITEM_COLOR'+strtrim(i,2), reform( item_colors[*,i]
)
endfor
```

Then the event handler could use SetProperty instead of SetPropertyByIdentifier to make changes to the actual ITEM_NAME array property:

```
new_value = WIDGET_INFO(event.id, $
                    COMPONENT = event.component, $
                    PROPERTY_VALUE = event.identifier)
if strmatch( event.identifier, 'ITEM_NAME*' ) eq 1 then begin
  self -> IDLgrLegend::GetProperty, item_name=item_names
  modify = fix( strmid( event.identifier, $
             strpos( event.identifer,'ITEM_NAME') $
                  +strlen('ITEM_NAME')))
  item_names[modify] = new_value
  self -> IDLgrLegend::SetProperty, item_name=item_names
endif
```

By the way this is all inside a class which inherits from the IDLgrLegend class.

Subject: Re: IDLgrLegend Property Sheets, array properties
Posted by Chris[2] on Tue, 20 Apr 2004 19:14:58 GMT

Hi Sean,

You basically have two options for handling non-scalar properties:

1. You can make it a type USERDEF, create an ::EditUserdefProperty method, and then handle the property on your own, using some sort of custom widget. This is good for really complicated properties.

2. However, as you suggested, you can also just split the property up into multiple properties.

If you do #2, you *must* explicitely handle these properties in the Get/SetProperty of your class, just like all other properties. You can't just do the replacement of the property value within the event handler, because this will only work for SetProperty. When the property sheet is displayed, it automatically calls ::GetProperty to retrieve the values. So you need to handle splitting the property up, and repackaging it, within your Get/SetProperty of your subclass.

This would work fine for a property that has a known # of values. Unfortunately, in your case, you have a problem because your legend could have an unlimited # of values, which makes it difficult to handle via keywords to Get/SetProperty.

We actually had the same problem in the iTools, with iContour. In this case, you can have multiple contour levels, each with its own set of properties. To solve this, we used a USERDEF property, that fires up a multi-column property sheet, with each column corresponding to a contour level. This *might* be a good solution. If you do this, you will need to create a lightweight "legend item" class, which contains the properties for a single item. You then hand an array of these legend item objects to the property sheet.

Or, a hacky solution would be to limit the # of legend items to say 10, and then just hardcode a bunch of keywords to your subclass Get/Setproperty.

Hope this helps.

-Chris
Research Systems, Inc.

"Sean Dettrick" <sdettrick@hotmail.com> wrote in message news:c233c3ea.0404201018.5c9690d9@posting.google.com...
> I hate to reply to my own message but it has occurred to me that I can
> just register each element of the ITEM_NAME and each triplet of the
> ITEM_COLOR as a seperate property with a seperate name (e.g.

```
>  ITEM_NAME0, ITEM_NAME1, etc).  In principle I guess the following
>  would work ( I say "in principle", because so far I can't get it to
>  work)
>
>  ; Register ARRAY properties element-by-element
>  for i=0,n_elements( item_names )-1 do begin
>     self -> IDLitComponent::RegisterProperty, $
>                  'ITEM_NAME'+strtrim(i,2),/STRING
>     self -> IDLitComponent::RegisterProperty, $
>                  'ITEM_COLOR'+strtrim(i,2), /COLOR
>
>     self -> IDLitComponent::SetPropertyByIdentifier, $
>                  'ITEM_NAME'+strtrim(i,2), item_names[i]
>     self -> IDLitComponent::SetPropertyByIdentifier, $
>                  'ITEM_COLOR'+strtrim(i,2), reform( item_colors[*,i]
>  )
>  endfor
>
>  Then the event handler could use SetProperty instead of
>  SetPropertyByIdentifier to make changes to the actual ITEM_NAME array
>  property:
>
>  new_value = WIDGET_INFO(event.id, $
>                       COMPONENT = event.component, $
>                       PROPERTY_VALUE = event.identifier)
>  if strmatch( event.identifier, 'ITEM_NAME*' ) eq 1 then begin
>     self -> IDLgrLegend::GetProperty, item_name=item_names
>     modify = fix( strmid( event.identifier, $
>                  strpos( event.identifer,'ITEM_NAME') $
>                       +strlen('ITEM_NAME')))
>     item_names[modify] = new_value
>     self -> IDLgrLegend::SetProperty, item_name=item_names
>  endif
>
>  By the way this is all inside a class which inherits from the
>  IDLgrLegend class.
```

---

## Subject: Re: IDLgrLegend Property Sheets, array properties
## Posted by Chris[2] on Wed, 21 Apr 2004 03:52:07 GMT

View Forum Message <> Reply to Message

Replying to my own post...

I forgot to mention that the "EditUserdefProperty" method is actually part of the iTools framework.

What you really get is a standard widget event from the property sheet,

where the property is type userdef. You are then free to do whatever you
want with that event, presumably firing up another widget... So you don't
need to implement an EditUserdefProperty method. Sorry about the confusion.

-Chris

---

Subject: Re: IDLgrLegend Property Sheets, array properties
Posted by sdettrick on Wed, 21 Apr 2004 04:20:20 GMT
View Forum Message <> Reply to Message

"Chris Torrence" <chris_torrence@NO-SPAM.yahoo.com> wrote in message
news:<108atgenh7rqdb3@corp.supernews.com>...
> Hi Sean,

Hi Chris,

thanks for the response. It is Very nice to hear from you folks at
RSI because one can feel comfortable that it is a definitive answer!

> You basically have two options for handling non-scalar properties:
>
> 1. You can make it a type USERDEF, create an ::EditUserdefProperty method,
> and then handle the property on your own, using some sort of custom widget.
> This is good for really complicated properties.
>
> 2. However, as you suggested, you can also just split the property up into
> multiple properties.

After tooling around for a while with _REF_EXTRA in GetProperty I
finally capitulated and went the way of your method 2. I like your
suggestion 1 though, it sounds much prettier. At present I have 20
hard-coded MY_ITEM_NAME_? and MY_ITEM_COLOR_? keywords (?=0-9) in
GetProperty. Nasty looking, but it works.

This is another case where the completely opaque nature of _REF_EXTRA
is a killer. Do you think there will ever be a function to fix that,
or do you know of a workaround?

In contrast the _EXTRA keyword is very friendly, and no hard-coded
list of keywords was needed in SetProperty. Instead it was
straightforward to strip out all of the tags, query ITEM_NAME with
GetProperty, get the index of the item_name passed as a keyword,
update the ITEM_NAME array with SetProperty, and there you go,
Robert's your father's brother(!):

; This SetProperty method accepts from

```
; MY_ITEM_NAME_0 to MY_ITEM_NAME_32767,
; via the _EXTRA keyword:

PRO myLegend::SetProperty, _EXTRA=EXTRA
self->IDLgrLegend::SetProperty, _EXTRA=extra
IF n_elements( extra ) ne 0 then begin
  TAGS = tag_names( EXTRA )
  name_locs = where( strmatch( TAGS, 'MY_ITEM_NAME_*' ) eq 1 )
  IF name_locs[0] ne -1 then begin
    self -> IDLgrLegend::GetProperty, ITEM_NAME=item_names
    FOR n=0,n_elements(name_locs)-1 do begin
      index = fix( strmid( tags[name_locs[n]], $
                strlen('MY_ITEM_NAME_')))
      item_names[index] = extra.(name_locs[n])
    ENDFOR
    self -> IDLgrLegend::SetProperty, ITEM_NAME=item_names
  ENDIF
ENDIF
END
```

> Or, a hacky solution would be to limit the # of legend items to say 10, and
> then just hardcode a bunch of keywords to your subclass Get/Setproperty.

Yes indeed. Anyway now I have a Legend class that manages its own property sheet widget so I can't complain too much.

> Hope this helps.

It helps a lot. Thanks a bundle.

Sean

---

Subject: Re: IDLgrLegend Property Sheets, array properties
Posted by sdettrick on Wed, 21 Apr 2004 18:58:23 GMT
View Forum Message <> Reply to Message

"Chris Torrence" <chris_torrence@NO-SPAM.yahoo.com> wrote in message news:<o6CdnRQ-QZF7chjdRVn-uQ@comcast.com>...
> Replying to my own post...
>
> I forgot to mention that the "EditUserdefProperty" method is actually part
> of the iTools framework.
>
> What you really get is a standard widget event from the property sheet,
> where the property is type userdef. You are then free to do whatever you
> want with that event, presumably firing up another widget... So you don't
> need to implement an EditUserdefProperty method. Sorry about the confusion.

>
> -Chris

Thanks, I was just about to ask about that. My hair stood on end when I looked up EditUserdefProperty and found that the first argument was a reference to an iTool!

Cheers,
Sean

---