Subject: Widget_event and a TLB object !!!
Posted by Antonio Santiago on Mon, 13 Jun 2005 14:55:33 GMT
View Forum Message <> Reply to Message

Hi I am very desperated with a widget object based I'm writting.
This is the situation:

I want to create an object that in its INIT method creates a WIDGET_BASE
with some buttons a text widgets. I wnat to collect some values, then
the user close (unmap) the widget_base, recollect the values with some
object method (GetProperty) and destroy the object.

Well, the problems comes with event handler. I can use XMANAGER because
it cant bring the use of methods, then I tried to use WIDGET_EVENT, but
I had a lot of problems to resolve the situation. Now I know where it
fails but no why it fails :(

Initialy, I assoaciated to the WIDGET_BASE an EVENT_PRO procedure
(extracted from this http://www.rsinc.com/codebank/search.asp?FID=209
from JP). This procedure is responsible to send ocurred events into the
widget_Base to a method of my object (MyObject::Event_Handler). Well
this is part of the code (I attach all it):

FUNCTION NewProcess::Init

```
 base = WIDGET_BASE(/COLUMN, $
                    TITLE='New Process Wizard', UVALUE=self, $
                    EVENT_PRO='GenericClassEventHandler')

;;Create widgets
content = WIDGET_BASE(base, XSIZE=cxsize, YSIZE=cysize, $
                     X_SCROLL_SIZE=cxsize, Y_SCROLL_SIZE=cysize)
text = WIDGET_TEXT(content, XSIZE=20, YSIZE=30)

line = WIDGET_BASE(base, XSIZE=500, YSIZE=0, /FRAME)

buttons = WIDGET_BASE(base, /ROW)
previous = WIDGET_BUTTON(buttons, VALUE='< Previous', $
                       UVALUE='PREVIOUS', SENSITIVE=0)
next = WIDGET_BUTTON(buttons, VALUE='Next >', UVALUE='NEXT', $
                    SENSITIVE=1)
finish = WIDGET_BUTTON(buttons, VALUE='Finish', UVALUE='FINISH', $
                     SENSITIVE=0)
cancel = WIDGET_BUTTON(buttons, VALUE='Cancel', UVALUE='CANCEL')

;;Store references
self.base = base
self.content = content
```

```
    self.previous = previous
    self.next = next
    self.finish = finish
    self.cancel = cancel

    WIDGET_CONTROL, self.base, /REALIZE

    event = WIDGET_EVENT(self.base)
    print, 'Never shows this line'

    ;;Event Loop
;    WHILE NOT self.destroy DO BEGIN
;        event = WIDGET_EVENT(base)
;        self->EventHandler, event
;    ENDWHILE

    RETURN, 1
END
```

Why WIDGET_EVENT call never returns ??? I dont know if I missunderstand the doc, but I though WIDGET_EVENT gets the event, call the EVENT_PRO associated with my WIDGET_BASE and returns.

To test it execute:

```
p = OBJ_NEW('NewProcess')
```

and press NEXT button and later CANCEL button.

Finally, I had removed the EVENT_PRO from the WIDGET_BASE, uncomment the above comment code and comment the line "event=WIDGET_EVENT...".

Thanks in advice,
Antonio (a little upset :( )

--

Antonio Santiago Pï¿½rez
(email: santiago<<at>>grahi.upc.edu   )
(www: http://www.grahi.upc.edu/santiago)
(www: http://asantiago.blogsite.org   )

GRAHI - Grup de Recerca Aplicada en Hidrometeorologia
Universitat Politï¿½cnica de Catalunya

```
------------------------------------------------

;+
; @FILE_COMMENTS Answer the information needed to create a new
;               process.
;
; @AUTHOR
;   Antonio Santiago
;   http://www.grahi.upc.edu/santiago - <santiago\@grahi.upc.edu>
;                                       <asantiagop\@gmail.com>
;
; @HISTORY
;      Fri Jun 10 13:31:13 2005, Antonio Santiago
;      (santiago\@grahi.upc.edu - asantiagop\@gmail.com)
;-



;+
; Event handler
;
; @PRIVATE
;
; @PARAM event {in}{required}{type=event struct} Event stucture with
;              produced information.
;-
PRO NewProcess::EventHandler, event

   WIDGET_CONTROL, event.id, GET_UVALUE=uval

   IF ~N_ELEMENTS(uval) THEN RETURN

   CASE uval OF
      'NEXT': BEGIN
         print, 'next'
      END

      'CANCEL': BEGIN
         ;self.destroy = 1
         WIDGET_CONTROL, self.base, /DESTROY
      END

      ELSE:                ;Nothing
   ENDCASE
END



;+
; @FILE_COMMENTS This routine is called when an event occurs in any
```

```
;             object-based TLB (Top-Level Bases). The 'EventHandler'
;             method of the
;             class whose object reference is stored in the
;             'event.handler's UVALUE is called.</br></br>
;             <i>This is a personal modification of the
;             'generic_class_event.pro' procedure created by Jim
;             Pendleton (jimp\@rsinc.com). You can find the original
;             code in the RSI User Contribution page.</i>
;
; @PARAM event {in}{required}{type=structure} The widget event produced.
;
; @EXAMPLES
;   Inside your object-based TLB use WIDGET_CONTROL to associate this
;   procedure as the events ocurred in the widget:
; <pre>
;       ...
;       WIDGET_CONTROL, the_widget, SET_UVALUE = self,
;       EVENT_PRO='Generic_Class_Event'<br>
; ...
; </pre>
;-
PRO GenericClassEventHandler, Event
    WIDGET_CONTROL, event.handler, GET_UVALUE=oSelf
    IF (N_ELEMENTS(oSelf) EQ 1) THEN BEGIN
        IF (OBJ_VALID(oSelf)) THEN BEGIN
            ;; A class that uses this routine must have a method
            ;; named "EventHandler".
            oSelf->EventHandler, event
        ENDIF
    ENDIF
END



;+
; Show plugin informacion stored in a PluginInfo object.
;
; @PARAM wbase {in}{required}{type=long} Widget base where to put the
;             information.
;-
FUNCTION NewProcess::Init

     base = WIDGET_BASE(/COLUMN, $
                        TITLE='New Process Wizard', UVALUE=self, $
                        EVENT_PRO='GenericClassEventHandler')

    ;;Create widgets
    content = WIDGET_BASE(base, XSIZE=cxsize, YSIZE=cysize, $
                          X_SCROLL_SIZE=cxsize, Y_SCROLL_SIZE=cysize)
```

```
   text = WIDGET_TEXT(content, XSIZE=20, YSIZE=30)

   line = WIDGET_BASE(base, XSIZE=500, YSIZE=0, /FRAME)

   buttons = WIDGET_BASE(base, /ROW)
   previous = WIDGET_BUTTON(buttons, VALUE='< Previous', $
                        UVALUE='PREVIOUS', SENSITIVE=0)
   next = WIDGET_BUTTON(buttons, VALUE='Next >', UVALUE='NEXT', $
                     SENSITIVE=1)
   finish = WIDGET_BUTTON(buttons, VALUE='Finish', UVALUE='FINISH', $
                      SENSITIVE=0)
   cancel = WIDGET_BUTTON(buttons, VALUE='Cancel', UVALUE='CANCEL')

   ;;Store references
   self.base = base
   self.content = content
   self.previous = previous
   self.next = next
   self.finish = finish
   self.cancel = cancel

   WIDGET_CONTROL, self.base, /REALIZE

 event = WIDGET_EVENT(self.base)

 print, 'Never shows this line'

   ;;Event Loop
;    WHILE NOT self.destroy DO BEGIN
;       event = WIDGET_EVENT(base)
;       self->EventHandler, event
;    ENDWHILE

   RETURN, 1
END


;+
; Frees the memory used by the object.
;-
PRO NewProcess::Cleanup
   WIDGET_CONTROL, self.base, /DESTROY
END


;+
; Newprocess class definition.
;
```

```
; @PRIVATE
;-
PRO NewProcess__define

   struct = {NewProcess, $
            base: 0L, $
            content: 0L, $
            previous: 0L, $
            next: 0L, $
            finish: 0L, $
            cancel: 0L, $
            destroy: 0B $
           }
END
```

File Attachments

1) newprocess__define.pro, downloaded 127 times