Subject: enhanced container object
Posted by Martin Schultz on Fri, 21 Jul 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Hi, everybody out there,

as always, progress is much slower than anticipated, but to keep my spirits
up and to take the chance of valuable feedback from you gurus, I'd like to present
a new object of mine, MGS_Container, which enhances the IDL_Container object by
allowing child-object access via the child objects' "name" attribute (if they have
one). Thus, to retrieve all objects which are called "Hugo", you can write:

```
  theContainer->Get(name='Hugo')
```

It even goes beyond that and allows a simple wildcard search with a trailing
'*'. To find all objects whose name starts with 'A' (or 'a') type:

```
  theContainer->Get(name='A*')
```

Objects without a name attribute (or without a GetProperty method which returns
the name via a "name" keyword), are recognized as name=''.

The other small enhancement with respect to IDL_Container is that you can
remove several objects at once, i.e. it is now allowed to type e.g.

```
  theContainer->Remove,position=[0,2,4,6]
```

(and, of course, also : theContainer->Remove,name=['A*','H*'])

Two new methods allow to retrieve all objects' names (GetName()), and the
index position of objects (GetPosition(objlist), where objlist is an object
reference array or a string list with names).

Hope you find this useful,
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   [[
[[[[[[[[

```
;+
; NAME:
;   MGS_Container (Object)
;
; PURPOSE:
;      This object extends the IDL_Container object (see online help)
;   by allowing access to objects by name. If all objects stored in
;   the container possess a structure element called 'name' and allow
;   access to this element via a GetProperty method, MGS_Container
;   allows the following in excess of IDL_Container's functionality:
;   1. retrieve the container position objects via their name attributes
;   2. retrieve object references via the objects' name attributes
;   3. remove objects via their name attributes
;      Objects without a name attribute (or without a GetProperty
;   method) are internally handled as objects with an empty name string.
;      The Get, and Remove methods accept a single string or a string
;   vector with the 'name' keyword. Each name may end with a '*' which
;   is interpreted as "and any other characters including none". Note,
;   however, that '*' alone will not return empty names ('').
;      A new method GetPosition accepts an object list as
;   argument. This can either be a list of object references or a
;   string (or string vector). This method returns an index vector
;   which can be used as 'position' value in other methods.
;
; AUTHOR:
;   Dr. Martin Schultz
;   Max-Planck-Institut fuer Meteorologie
;   Bundesstr. 55, D-20146 Hamburg
;   email: martin.schultz@dkrz.de
;
; CATEGORY:
;   Objects, General Programming
;
; CALLING SEQUENCE:
;   theContainer = Obj_New( 'MGS_Container' )
;   theContainer->Add, objectref
;   objectref = theContainer->Get( [keywords] )
;   Obj_Destroy, theContainer
;
; KEYWORDS TO THE INITIALIZATION ROUTINE:
;   None.
;
; PROCEDURES AND METHODS:
;   For a more detailed explanation on individual methods or
```

```
;   procedures, see the comments in the respective code sections.
;   The following list contains only "public" procedures and methods.
;   Methods marked with '*' are changed or new with respect to the
;   parent IDL_Container object.
;
; NON-OBJECT PROCEDURES:
;   None
;
; PROCEDURE METHODS:
;   MGS_Container::Add, objref [Position=index]
;           Add an object to the container.
;   MGS_Container::Move, source, dest
;           Move the position of an object within the container.
; * MGS_Container::Remove, [ objectref | Position=index | /All |
;                       Name=string ]
;           Delete an object from the container. NEW: can specify
;           objects by name.
;
; FUNCTION METHODS:
;   MGS_Container::Count()
;           Return number of objects stored in container.
; * MGS_Container::Get( /All | IsA=classname | Position=index |
;                     Name=string  [, Count=variable] )
;           Return references to selected objects from the
;           container. If no object matches the search criteria, a
;           long value of -1 is returned (as in IDL_Container).
;           NEW: can specify objects by name.
;   MGS_Container::IsContained( objref [, Position=index] )
;           Test if object(s) is stored in container.
; * MGS_Container::GetNames( [Position=index] )
;           NEW method. Return the names of the objects stored in the
;           container. Objects without a name field yield an empty string.
; * MGS_Container::GetPosition( objlist )
;           NEW method. Returns the index values of objects specified
;           by object reference or name. Objlist can be a string vector
;           where each string can have a trailing '*' as wildcard.
;
; MODIFICATION HISTORY:
;
;   mgs, 19 Jul 2000 : VERSION 1.00
;
;-
;
 ;############################################################################
;
; LICENSE
;
; This software is OSI Certified Open Source Software.
```

```
; OSI Certified is a certification mark of the Open Source Initiative.
;
; Copyright ï¿½ 2000 Martin Schultz
;
; This software is provided "as-is", without any express or
; implied warranty. In no event will the authors be held liable
; for any damages arising from the use of this software.
;
; Permission is granted to anyone to use this software for any
; purpose, including commercial applications, and to alter it and
; redistribute it freely, subject to the following restrictions:
;
; 1. The origin of this software must not be misrepresented; you must
;    not claim you wrote the original software. If you use this software
;    in a product, an acknowledgment in the product documentation
;    would be appreciated, but is not required.
;
; 2. Altered source versions must be plainly marked as such, and must
;    not be misrepresented as being the original software.
;
; 3. This notice may not be removed or altered from any source distribution.
;
; For more information on Open Source Software, visit the Open Source
; web site: http://www.opensource.org.
;
 ;############################################################## ##################


 ;------------------------------------------------------- ----------
; GetOneName (*private*):
; This method returns the NAME field of the object stored at the
; position given as argument. If the object contains no NAME field or
; has no GetProperty method, an empty string is returned.
; Note: position must be a scalar!

FUNCTION MGS_Container::GetOneName, Position

   ;; Establish error handler in case object has no GetProperty method
   CATCH, theError
   IF theError NE 0 THEN BEGIN
      CATCH, /Cancel
      RETURN, ''
   ENDIF

   ;; Get the object reference of the specified object
   theObject = self->Get(Position=position)
   IF not Obj_Valid(theObject) THEN RETURN, ''
```

```
   ;; Atempt to call the object's GetProperty method to retrieve the
   ;; name
   theObject->GetProperty, name=name

   ;; If successful, return object name. Otherwise, error handler
   ;; should return empty string.
   RETURN, name

END


 ;------------------------------------------------------------ ----------------
; GetNames:
; This method returns the NAME fields of all objects stored in the
; container. If the optional POSITION keyword is used, only the names
; of the objects at the indices given by POSITION will be returned.
; Invalid position indices yield empty strings as names.

FUNCTION MGS_Container::GetNames, Position=position

   ;; If no position vector was passed, get number of objects stored
   ;; in container and create index list
   IF N_Elements(position) GT 0 THEN BEGIN
      index = position
   ENDIF ELSE BEGIN
      objcount = self->Count()
      IF objcount GT 0 THEN $
        index = lindgen(objcount) $
      ELSE $
        RETURN, ''
   ENDELSE

   ;; Create result array
   result = StrArr(N_Elements(index))

   ;; Get individual object names
   FOR i=0L, N_Elements(index)-1 DO $
     result[i] = self->GetOneName(index[i])

   RETURN, result

END


 ;------------------------------------------------------------ ----------------
; GetPosition:
; This method returns the position indices of the objects specified as
; object reference (in this case the IDL_Container method IsContained
```

```
; is used) or specified by names. A -1 is returned if no object in the
; container matches the search or if OBJLIST is neither of type string
; nor objref. Note that objects without name (i.e. empty strings) are
; not found with '*' as search mask but are found if an empty
; string is specified as objlist.

FUNCTION MGS_Container::GetPosition, objlist

  ;; Establish error handler to be safe
  CATCH, theError
  IF theError NE 0 THEN BEGIN
    CATCH, /Cancel
    RETURN, -1L
  ENDIF

  ;; If no objlist is given, return immediately
  IF N_Elements(objlist) EQ 0 THEN RETURN, -1L

  ;; Check if objlist is of type object reference or string
  type = Size(objlist, /TNAME)

  ;; For object references use the inherited IsContained method
  IF type EQ 'OBJREF' THEN BEGIN
    dummy = self->IsContained(objlist, Position=index)
    RETURN, index
  ENDIF

  ;; For strings, interpret them as object names. Get the names of
  ;; all objects and apply primitive pattern matching.
  ;; The search is always case-insensitive.
  IF type EQ 'STRING' THEN BEGIN
    names = self->GetNames()
    names = StrTrim( StrUpCase(names), 2)
    index = -1L
    ;; Loop through objlist and find matching indices
    FOR i=0L, N_Elements(objlist)-1 DO BEGIN
      testname = StrTrim( StrUpCase(objlist[i]), 2)

      ;; Check if wildcard is used in name
      sloppy = 0
      IF (( p = StrPos(testname,'*') )) GE 0 THEN BEGIN
        l = StrLen(testname)
        IF l-p NE 1 THEN BEGIN     ; wildcard must be last character
          Message, 'Invalid name specification '+objlist[i]+'!', /Continue
          GOTO, skip_it
        ENDIF
        testname = StrMid(testname, 0, l-1)
        sloppy = 1
```

```
        ENDIF

        ;; Look for matching object names
        IF sloppy THEN BEGIN
          w = Where(StrPos(names,testname) EQ 0, wcnt)
        ENDIF ELSE BEGIN
          w = Where(names EQ testname, wcnt)
        ENDELSE

        ;; Add indices to list
        IF wcnt GT 0 THEN index = [ index, w ]

skip_it:
      ENDFOR

      ;; Remove first dummy index if any names have been found
      IF N_Elements(index) GT 1 THEN index = index[1:*]

      RETURN, index
    ENDIF

    ;; Reaching this portion of the routine means invalid type for
    ;; objlist. Simply return -1.

    RETURN, -1L
END


 ;------------------------------------------------------------ ----------------
; Get:
; This method returns object references for objects that match
; specified criteria - or all objects if the All keyword is set.
; The IDL_Container method Get is extended to include a Name
; keyword. All and Position take precedence over name. As in the
; original, count can be used to return the number of objects
; retrieved.

FUNCTION MGS_Container::Get, All=all, Position=position, $
                  Name=name, Count=count, _Ref_Extra=e


    ;; If All or Position keywords are used, call inherited method
    IF Keyword_Set(all) THEN  $
      RETURN, self->IDL_Container::Get(/All, count=count)

    IF N_Elements(Position) GT 0 THEN $
      RETURN, self->IDL_Container::Get(Position=position, count=count)
```

```
  ;; Look for named objects ?
  IF N_Elements(name) GT 0 THEN BEGIN
    ;; Find object positions
    namepos = self->GetPosition(name)
    ;; Return objects
   RETURN, self->IDL_Container::Get(Position=namepos, count=count)
  ENDIF

  ;; Otherwise call inherited method with extra keywords
  RETURN, self->IDL_Container::Get(_extra=e, count=count)

END


 ;--------------------------------------------------------- ----------------
; Remove:
; This method removes objects from the container. Similarily to the
; Get method, this method extends the IDL_Container::Remove method to
; allow object specification by name.
; Note: Unlike the original method of IDL_Container, the child_object
; argument, or the position or name keywords can contain more than one
; element. The unique position values are sorted and reversed before
; removing objects.

PRO MGS_Container::Remove, child_object, name=name, position=position, $
              All=all, _Ref_Extra=e

  ;; If all keyword is set, call parent method accordingly
  IF Keyword_Set(all) THEN BEGIN
    self->IDL_Container::Remove, /All
    RETURN
  ENDIF

  ;; If object references are given, loop over them and call the
  ;; parent object's remove method for each of them
  IF N_Elements(child_object) GT 0 THEN BEGIN
    FOR i=0L, N_Elements(child_object)-1 DO $
      self->IDL_Container::Remove, child_object[i]
    RETURN
  ENDIF

  ;; If names are provided, convert them to position indices
  IF N_Elements(name) GT 0 THEN $
    index = self->GetPosition(name)

  ;; If position indices are provided, make local copy
  IF N_Elements(position) GT 0 THEN $
    index = long(position)
```

```
  ;; Handle name and position cases: find unique position values,
  ;; sort and reverse them and loop
  maxindex = self->Count()
  IF N_Elements(index) GT 0 THEN BEGIN
    ;; Need to sort index array and remove items from the end
    index = Reverse( index[ Uniq( index, Sort(index) ) ] )
    ok = where(index GE 0 AND index LT maxindex, cnt)
    IF cnt EQ 0 THEN RETURN   ;; no valid index values
    index = index[ok]
    FOR i=0L, N_Elements(index)-1 DO $
      self->IDL_Container::Remove, Position=index[i]
    RETURN
  ENDIF

  ;; Handle other (future extensions) cases
  self->IDL_Container::Remove, _extra=e

END


 ;--------------------------------------------------------- ----------------
; Cleanup:
; This method frees all object values and destroys all objects stored
; in the container. This is already handled in IDL_Container, so
; Cleanup only needs to call the inherited object's method.

PRO MGS_Container::Cleanup

  self->IDL_Container::Cleanup

END


 ;--------------------------------------------------------- ----------------
; Init:
; This method initializes the object values. There are no arguments,
; so the only action is to call the parent's Init method

FUNCTION MGS_Container::INIT

  RETURN, self->IDL_Container::Init()

END


 ;--------------------------------------------------------- ----------------
; MGS_Container__Define:
```

```
; This is the object definition for the container object. It inherits
; everything from IDL_Container.

PRO MGS_Container__Define

  object_class = {  MGS_Container,    $
                    INHERITS  IDL_Container  $
                 }

END
```

File Attachments

1) mgs_container__define.pro, downloaded 88 times