Subject: LIST extensions
Posted by Paul Van Delst[1] on Tue, 14 Dec 2010 21:51:11 GMT
View Forum Message <> Reply to Message

Hello,

Sorry to keep flogging this particular topic, but I'm currently up to my eyes in adapting some code from using pointers
to using lists (partly for curiosity, partly to avoid nested pointer dereferencing), and I've found the very simple
methods below to be useful in dealing with the "is it empty?", "are there nulls in the list?" etc type of questions I've
been asking here.

Comments?

cheers,

paulv

```
;+
; NAME:
;       List::Empty
;
; PURPOSE:
;       The List::Empty function method return TRUE if a list contains
;       no elements, FALSE otherwise.
;
; CALLING SEQUENCE:
;       result = Obj->[List::]Empty()
;-
FUNCTION List::Empty
  RETURN, N_ELEMENTS(self) EQ 0
END


;+
; NAME:
;       List::Length
;
; PURPOSE:
;       The List::Length function method returns the number of
;       elements in a list.
;
; CALLING SEQUENCE:
;       result = Obj->[List::]Length()
;-
```

```
FUNCTION List::Length
  RETURN, N_ELEMENTS(self)
END


;+
; NAME:
;       List::n_Items
;
; PURPOSE:
;       The List::n_Items function method returns the number of
;       non-null elements in a list.
;
; CALLING SEQUENCE:
;       result = Obj->[List::]n_Items()
;-
FUNCTION List::n_Items
  idx = WHERE(self NE !NULL, count)
  RETURN, count
END


;+
; NAME:
;       List::Compact
;
; PURPOSE:
;       The List::Compact function method returns a copy of a list
;       with all null elements removed.
;
; CALLING SEQUENCE:
;       result = Obj->[List::]Compact()
;-
FUNCTION List::Compact
  idx = WHERE(self NE !NULL, count)
  RETURN, ( count GT 0 ) ? self[idx] : LIST()
END
```