ABAP Code Sample for Dynamic Table for ALV with Cell Coloring

Applies To:

ABAP / ALV Grid

Article Summary

ABAP Code Sample that uses dynamic programming techniques to build a dynamic internal table for display in an ALV Grid with Cell Coloring.

By: Charles Folwell, Consultant
Date: 04 Feb 2005

Code Sample

```
REPORT  zcdf_dynamic_table.

* Dynamic ALV Grid with Cell Coloring.

*  Build a field catalog dynamically and provide the ability to color
*   the cells.

*  To test, copy this code to any program name and create screen 100
*   as described in the comments.  After the screen is displayed, hit
*   enter to exit the screen.

* Tested in 4.6C and 6.20

* Charles Folwell - cfolwell@csc.com - Feb 2, 2005

DATA:
  r_dyn_table      TYPE REF TO data,
  r_wa_dyn_table   TYPE REF TO data,
  r_dock_ctnr      TYPE REF TO cl_gui_docking_container,
  r_alv_grid       TYPE REF TO cl_gui_alv_grid,

  t_fieldcat1      TYPE lvc_t_fcat,           "with cell color
  t_fieldcat2      TYPE lvc_t_fcat,           "without cell color

  wa_fieldcat      LIKE LINE OF t_fieldcat1,
  wa_cellcolors    TYPE LINE OF lvc_t_scol,
  wa_is_layout     TYPE lvc_s_layo.

FIELD-SYMBOLS:
  <t_dyn_table>    TYPE STANDARD TABLE,
  <wa_dyn_table>   TYPE ANY,
  <t_cellcolors>   TYPE lvc_t_scol,
  <w_field>        TYPE ANY.
START-OF-SELECTION.
```

```
* Build field catalog based on your criteria.

  wa_fieldcat-fieldname = 'FIELD1'.
  wa_fieldcat-inttype   = 'C'.
  wa_fieldcat-outputlen = '10'.
  wa_fieldcat-coltext   = 'My Field 1'.
  wa_fieldcat-seltext   = wa_fieldcat-coltext.

  APPEND wa_fieldcat TO t_fieldcat1.

  wa_fieldcat-fieldname = 'FIELD2'.
  wa_fieldcat-inttype   = 'C'.
  wa_fieldcat-outputlen = '10'.
  wa_fieldcat-coltext   = 'My Field 2'.
  wa_fieldcat-seltext   = wa_fieldcat-coltext.

  APPEND wa_fieldcat TO t_fieldcat1.

* Before adding cell color table, save fieldcatalog to pass
*  to ALV call.  The ALV call needs a fieldcatalog without
*  the internal table for cell coloring.

  t_fieldcat2[] = t_fieldcat1[].

* Add cell color table.
*  CALENDAR_TYPE is a structure in the dictionary with a
*   field called COLTAB of type LVC_T_SCOL.  You can use
*   any structure and field that has the type LVC_T_SCOL.

  wa_fieldcat-fieldname = 'T_CELLCOLORS'.
  wa_fieldcat-ref_field = 'COLTAB'.
  wa_fieldcat-ref_table = 'CALENDAR_TYPE'.

  APPEND wa_fieldcat TO t_fieldcat1.

* Create dynamic table including the internal table
*  for cell coloring.

  CALL METHOD cl_alv_table_create=>create_dynamic_table
    EXPORTING
      it_fieldcatalog           = t_fieldcat1
    IMPORTING
      ep_table                  = r_dyn_table
    EXCEPTIONS
      generate_subpool_dir_full = 1
      OTHERS                    = 2.

  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
               WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.

* Get access to new table using field symbol.

  ASSIGN r_dyn_table->* TO <t_dyn_table>.

* Create work area for new table.

  CREATE DATA r_wa_dyn_table LIKE LINE OF <t_dyn_table>.

* Get access to new work area using field symbol.
```

```
  ASSIGN r_wa_dyn_table->* TO <wa_dyn_table>.

* Get data into table from somewhere.  Field names are
*  known at this point because field catalog is already
*  built.  Read field names from the field catalog or use
*  COMPONENT <number> in a DO loop to access the fields.  A
*  simpler hard coded approach is used here.

  ASSIGN COMPONENT 'FIELD1' OF STRUCTURE <wa_dyn_table> TO <w_field>.

  <w_field> = 'ABC'.

  ASSIGN COMPONENT 'FIELD2' OF STRUCTURE <wa_dyn_table> TO <w_field>.

  <w_field> = 'XYZ'.

  APPEND <wa_dyn_table> TO <t_dyn_table>.

  ASSIGN COMPONENT 'FIELD1' OF STRUCTURE <wa_dyn_table> TO <w_field>.

  <w_field> = 'TUV'.

  ASSIGN COMPONENT 'FIELD2' OF STRUCTURE <wa_dyn_table> TO <w_field>.

  <w_field> = 'DEF'.

  APPEND <wa_dyn_table> TO <t_dyn_table>.

* Color cells based on your criteria.  In this example, a test on
*  FIELD2 is used to decide on color.

  LOOP AT <t_dyn_table> INTO <wa_dyn_table>.

    ASSIGN COMPONENT 'FIELD2' OF STRUCTURE <wa_dyn_table> TO <w_field>.

*   Get access to internal table used to color cells.

    ASSIGN COMPONENT 'T_CELLCOLORS'
      OF STRUCTURE <wa_dyn_table> TO <t_cellcolors>.

    CLEAR wa_cellcolors.

    wa_cellcolors-fname = 'FIELD2'.

    IF <w_field> = 'DEF'.
      wa_cellcolors-color-col = '7'.
    ELSE.
      wa_cellcolors-color-col = '5'.
    ENDIF.

    APPEND wa_cellcolors TO <t_cellcolors>.

    MODIFY <t_dyn_table> FROM <wa_dyn_table>.

  ENDLOOP.
```

```
* Display screen.  Define screen 100 as empty, with next screen
*  set to 0 and flow logic of:
*
*    PROCESS BEFORE OUTPUT.
*      MODULE initialization.
*
*    PROCESS AFTER INPUT.

  CALL SCREEN 100.

*---------------------------------------------------------------------*
*  MODULE initialization OUTPUT
*---------------------------------------------------------------------*

MODULE initialization OUTPUT.

* Set up for ALV display.

  IF r_dock_ctnr IS INITIAL.

    CREATE OBJECT r_dock_ctnr
           EXPORTING
              side  = cl_gui_docking_container=>dock_at_left
              ratio = '90'.

    CREATE OBJECT r_alv_grid
           EXPORTING i_parent = r_dock_ctnr.

*   Set ALV controls for cell coloring table.

    wa_is_layout-ctab_fname = 'T_CELLCOLORS'.

*   Display.

    CALL METHOD r_alv_grid->set_table_for_first_display
      EXPORTING
        is_layout       = wa_is_layout
      CHANGING
        it_outtab       = <t_dyn_table>
        it_fieldcatalog = t_fieldcat2.

  ELSE.     "grid already prepared

*   Refresh display.

    CALL METHOD r_alv_grid->refresh_table_display
      EXPORTING
        i_soft_refresh = ' '
      EXCEPTIONS
        finished       = 1
        OTHERS         = 2.

  ENDIF.

ENDMODULE.                 " initialization  OUTPUT
```

Disclaimer & Liability Notice

This document may discuss sample coding, which does not include official interfaces and therefore is not supported. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing of the code and methods suggested here, and anyone using these methods, is doing it under his/her own responsibility.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of the technical article, including any liability resulting from incompatibility between the content of the technical article and the materials and services offered by SAP. You agree that you will not hold SAP responsible or liable with respect to the content of the Technical Article or seek to do so.