Creating Dynamic Internal Table

REPORT ZR_DYNTBL_155.

INCLUDE : zr_dyntbl_155_top,
          zr_dyntbl_155_frm,
          zr_dyntbl_155_cls.

INITIALIZATION.
  CREATE OBJECT o_obj.

START-OF-SELECTION.
  DATA col TYPE i.
  IF p_l1 = abap_true.
    o_obj->set_col( 1 ).
  ELSEIF p_l2 = abap_true.
    o_obj->set_col( 2 ).
  ELSEIF p_l3 = abap_true.
    o_obj->set_col( 3 ).
  ENDIF.

END-OF-SELECTION.
  o_obj->create_dyn_tbl( ).
  o_obj->populate_fcat( ).
  o_obj->populate_dyn_tbl( ).
  o_obj->call_screen( ).

Include: ZR_DYNTBL_155_TOP
  "&---------------------------------------------------------------------*  
  *& Include ZR_DYNTBL_155_TOP                                        
  "&---------------------------------------------------------------------*  

CLASS lcl_dyntbl DEFINITION DEFERRED.

  * static fields of dynamic table
  TYPES : BEGIN OF ty_a,
            f1 TYPE string,
            f2 TYPE string,
  END OF ty_a.

  * dynamic fields of dynamic table
  TYPES: BEGIN OF ty_b,
c1 TYPE string,
c2 TYPE string,
END OF ty_b.

* structure for 2 dynamic column table
TYPES: BEGIN OF ty_c.
   INCLUDE TYPE ty_a.
   INCLUDE TYPE ty_b AS lvl1 RENAMING WITH SUFFIX l1.
TYPES : END OF ty_c.

* structure for 4 dynamic column table
TYPES: BEGIN OF ty_d.
   INCLUDE TYPE ty_c.
   INCLUDE TYPE ty_b AS lvl2 RENAMING WITH SUFFIX l2.
TYPES : END OF ty_d.

* structure for 6 dynamic column table
TYPES: BEGIN OF ty_e.
   INCLUDE TYPE ty_d.
   INCLUDE TYPE ty_b AS lvl3 RENAMING WITH SUFFIX l3.
TYPES : END OF ty_e.

DATA:
  v_i TYPE i,
  wa_a TYPE ty_a,
  wa_b TYPE ty_b,
  wa_c TYPE ty_c,
  wa_d TYPE ty_d,
  wa_e TYPE ty_e,
  it_a TYPE TABLE OF ty_a,
  it_b TYPE TABLE OF ty_b,
  it_c TYPE TABLE OF ty_c,
  it_d TYPE TABLE OF ty_d,
  it_e TYPE TABLE OF ty_e,
  o_obj TYPE REF TO lcl_dyn_tbl.

CONSTANTS : c_err TYPE c VALUE 'E',
            c_suc TYPE c VALUE 'S'.

FIELD-SYMBOLS: 

<it_dyn> TYPE STANDARD TABLE,"fs to hold the dynamic tbl
<fx_fcat> TYPE lvc_s_fcat,
<fs_a> TYPE ty_a,
<fs_b1> TYPE ty_b,
<fs_b2> TYPE ty_b,
<fs_b3> TYPE ty_b,
<fs_c> TYPE ty_c,
<fs_d> TYPE ty_d,
<fs_e> TYPE ty_e,
<fs_dy>, " fs to hold wa of dynamic tbl
<fs_v>, " fs to hold field value of dynamic wa

DEFINE fcat_mrg.
APPEND INITIAL LINE TO it_fcat ASSIGNING <fx_fcat>.
<fx_fcat>-col_pos = &1.
<fx_fcat>-fieldname = &2.
<fx_fcat>-scrtext_l = &3.
END-OF-DEFINITION.

* populating sample data for dynamic table

CLEAR wa_a.
wa_a-f1 = 'a'.
wa_a-f2 = 'b'.
APPEND wa_a TO it_a.

CLEAR wa_a.
wa_a-f1 = 'aa'.
wa_a-f2 = 'bb'.
APPEND wa_a TO it_a.

CLEAR wa_b.
wa_b-c1 = 'a'.
wa_b-c2 = 'z'.
APPEND wa_b TO it_b.

CLEAR wa_b.
wa_b-c1 = 'aa'.


wa_b-c2 = 'zz'.
APPEND wa_b TO it_b.

Include: ZR_DYNTBL_155_FRM

SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME TITLE text-001.
PARAMETERS: P_11 RADIOBUTTON GROUP R1 DEFAULT 'X',
            P_12 RADIOBUTTON GROUP R1,
            P_13 RADIOBUTTON GROUP R1.
SELECTION-SCREEN END OF BLOCK b1.

Include: ZR_DYNTBL_155_CLS

CLASS lcl_dyntbl DEFINITION.  " begin of lcl_dyntbl class definition
PUBLIC SECTION.
METHODS:
  set_col IMPORTING c TYPE i,
  create_dyn_tbl,
  populate_fcat,
  populate_dyn_tbl,
  display_alv,
  call_screen.

PRIVATE SECTION.
DATA :
  it_fcat TYPE lvc_t_fcat,
  o_cont TYPE REF TO cl_gui_custom_container,
  o_alv TYPE REF TO cl_gui_alv_grid,
  v_col TYPE i,  " variable to store no. of dynamic columns
  wa_tbl_ref TYPE REF TO data, " dynamic variable to create table structure
  wa_st_ref TYPE REF TO data.  " dynamic

ENDCLASS.  " end of lcl_dyntbl class definition

CLASS lcl_dyntbl IMPLEMENTATION.
  " begin of lcl_dyntbl class implementation

* setting no. of columns of table
METHOD set_col.
  v_col = c.
METHOD create_dyn_tbl.

CASE v_col.
  WHEN 1.  * case for 2 dynamic columns
    CREATE DATA wa_tbl_ref TYPE TABLE OF ty_c.
    CREATE DATA wa_st_ref TYPE ty_c.

  WHEN 2.  * case for 4 dynamic columns
    CREATE DATA wa_tbl_ref TYPE TABLE OF ty_d.
    CREATE DATA wa_st_ref TYPE ty_d.

  WHEN 3.  * case for 6 dynamic columns
    CREATE DATA wa_tbl_ref TYPE TABLE OF ty_e.
    CREATE DATA wa_st_ref TYPE ty_e.
ENDCASE.

ASSIGN wa_tbl_ref->* TO <it_dyn>.
ASSIGN wa_st_ref->* TO <fs_dy>.

ENDMETHOD.

METHOD populate_fcat.

  fcat_mrg : 1 'F1' 'Column 1',
             2 'F2' 'Column 2',
  IF v_col >= 1.

  fcat_mrg : 3 'C1L1' 'Column 3',
             4 'C2L1' 'Column 4',
  IF v_col >= 2.

  fcat_mrg : 5 'C1L2' 'Column 5',
             6 'C2L2' 'Column 6',
  IF v_col = 3.

  fcat_mrg : 7 'C1L3' 'Column 7',
METHOD populate_dyn_tbl.

* populating the dynamic internal table

LOOP AT it_a ASSIGNING <fs_a>.
* reading other table dynamically acc. to the no of col. to disp.
  READ TABLE it_b ASSIGNING <fs_b1> WITH KEY c1 = <fs_a>-f1.
  IF v_col >= 2.
    READ TABLE it_b ASSIGNING <fs_b2> WITH KEY c1 = <fs_a>-f1.
    IF v_col = 3.
      READ TABLE it_b ASSIGNING <fs_b3> WITH KEY c1 = <fs_a>-f1.
      ENDIF.
  ENDIF.
ENDIF.

* assigning values to dynamic working area using fcat

LOOP AT it_fcat ASSIGNING <fx_fcat>.
  ASSIGN COMPONENT <fx_fcat>-fieldname OF STRUCTURE <fs_dy> TO <fs_v>.
  CASE <fx_fcat>-fieldname.
    WHEN 'F1'.
      IF <fs_a> IS ASSIGNED.
        <fs_v> = <fs_a>-f1.
      ENDIF.
    WHEN 'F2'.
      IF <fs_a> IS ASSIGNED.
        <fs_v> = <fs_a>-f2.
      ENDIF.
    WHEN 'C1L1'.
      IF <fs_b1> IS ASSIGNED.
        <fs_v> = <fs_b1>-c1.
      ENDIF.
    WHEN 'C2L1'.
      IF <fs_b1> IS ASSIGNED.
        <fs_v> = <fs_b1>-c2.
ENDIF.
WHEN 'C1L2'.
IF <fs_b2> IS ASSIGNED.
  <fs_v> = <fs_b2>-c1.
ENDIF.
WHEN 'C2L2'.
IF <fs_b2> IS ASSIGNED.
  <fs_v> = <fs_b2>-c2.
ENDIF.
WHEN 'C1L3'.
IF <fs_b3> IS ASSIGNED.
  <fs_v> = <fs_b3>-c1.
ENDIF.
WHEN 'C2L3'.
IF <fs_b3> IS ASSIGNED.
  <fs_v> = <fs_b3>-c2.
ENDIF.
ENDCASE.
ENDLOOP.
APPEND <fs_dy> TO <it_dyn>.
ENDLOOP.
ENDMETHOD.

* displaying internal table

METHOD display_alv.
IF o_cont IS NOT BOUND.
CREATE OBJECT o_cont  * creating container for alv
  EXPORTING
    container_name = 'CONT'
EXCEPTIONS
  cntl_error = 1
  cntl_system_error = 2
  create_error = 3
  lifetime_error = 4
  lifetime_dynpro_dynpro_link = 5
  OTHERS = 6.
CASE sy-subrc.
WHEN 1.
MESSAGE 'Control Error'(002) TYPE c_suc
  DISPLAY LIKE c_err.
WHEN 2.
  MESSAGE 'Control System Error'(003) TYPE c_suc
  DISPLAY LIKE c_err.
WHEN 3.
  MESSAGE 'Create Error'(004) TYPE c_suc
  DISPLAY LIKE c_err.
WHEN 4.
  MESSAGE 'Lifetime Error'(005) TYPE c_suc
  DISPLAY LIKE c_err.
WHEN 5.
  MESSAGE 'Lifetime Dynpro Dynpor Link'(006) TYPE c_suc
  DISPLAY LIKE c_err.
WHEN 6.
  MESSAGE 'Other Error'(007) TYPE c_suc
  DISPLAY LIKE c_err.
ENDCASE.

CREATE OBJECT o_alv
  EXPORTING
    i_parent = o_cont
EXCEPTIONS
  error_cntl_create = 1
  error_cntl_init = 2
  error_cntl_link = 3
  error_dp_create = 4
  OTHERS = 5.
CASE sy-subrc.
WHEN 1.
  MESSAGE 'Error Control Create'(008) TYPE c_suc
  DISPLAY LIKE c_err.
WHEN 2.
  MESSAGE 'Error Control Initialization'(009) TYPE c_suc
  DISPLAY LIKE c_err.
WHEN 3.
  MESSAGE 'Error Control Link'(010) TYPE c_suc
  DISPLAY LIKE c_err.
WHEN 4.
MESSAGE 'Error DP Create'(011) TYPE c_suc
DISPLAY LIKE c_err.
WHEN 5.
MESSAGE 'Other Error'(007) TYPE c_suc
DISPLAY LIKE c_err.
ENDCASE.

CALL METHOD o_alv->set_table_for_first_display
CHANGING
  it_outtab = <it_dyn>
  it_fieldcatalog = it_fcat
EXCEPTIONS
  invalid_parameter_combination = 1
  program_error = 2
  too_many_lines = 3
  OTHERS = 4.
CASE sy-subrc.
  WHEN 1.
    MESSAGE 'Invalid Parameter Combination'(012) TYPE c_suc
    DISPLAY LIKE c_err.
  WHEN 2.
    MESSAGE 'Program Error'(013) TYPE c_suc
    DISPLAY LIKE c_err.
  WHEN 3.
    MESSAGE 'Too Many Lines'(014) TYPE c_suc
    DISPLAY LIKE c_err.
  WHEN 4.
    MESSAGE 'Other Error'(007) TYPE c_suc
    DISPLAY LIKE c_err.
ENDCASE.
ELSE.
CALL METHOD o_alv->refresh_table_display
EXCEPTIONS
  finished = 1
  OTHERS = 2.
CASE sy-subrc.
  WHEN 1.
MESSAGE 'Finished'(015) TYPE c_suc
DISPLAY LIKE c_err.
WHEN 2.
MESSAGE 'Other Error'(007) TYPE c_suc
DISPLAY LIKE c_err.
ENDCASE.
ENDIF.
ENDMETHOD.

METHOD call_screen.
CALL SCREEN 9000.
ENDMETHOD.

ENDCLASS." end of lcl_dyntbl class implementation

*---------------------------------------------------------------------*
*&         Module STATUS_9000 OUTPUT
*---------------------------------------------------------------------*
*       PBO of Screen 9000
*---------------------------------------------------------------------*
MODULE status_9000 OUTPUT.
SET PF-STATUS 'ST_9000'.
SET TITLEBAR 'TTL_9000'.

o_obj->display_alv( ).

ENDMODULE.

*---------------------------------------------------------------------*
*&         Module USER_COMMAND_9000 INPUT
*---------------------------------------------------------------------*
*       PAI of Screen 9000
*---------------------------------------------------------------------*
MODULE user_command_9000 INPUT.
DATA ok_code TYPE sy-ucomm.
CASE ok_code.
WHEN 'CD_BCK'.
LEAVE TO SCREEN 0.
WHEN 'CD_EXT'.
  LEAVE PROGRAM.
WHEN 'CD_CNCL'.
  LEAVE TO SCREEN 0.
WHEN OTHERS.
  ENDCASE.
ENDMODULE.