```
*&---------------------------------------------------------------------*
*&Report:ZALV_dragdrop                                                 *
*&Author : Swarna.S
*&---------------------------------------------------------------------*
*& AS : ALV report with drag and drop functionality on its rows
*---------------------------------------------------------------------*

REPORT  zalv_dragdrop.

*Structure declaration for T016T
TYPES : BEGIN OF ty_t016t,
         brsch TYPE brsch,
         brtxt TYPE text1_016t,
         spras TYPE spras,
         END OF ty_t016t.

*Work area and internal table for  T016T
DATA : it_t016t TYPE STANDARD TABLE OF ty_t016t,
       wa_t016t TYPE ty_t016t.

*class declaration
CLASS lcl_objdragdropapp DEFINITION DEFERRED.

*data declarations for alv
DATA:   c_dragdropapp TYPE REF TO lcl_objdragdropapp,
        c_dockingcont TYPE REF TO cl_gui_docking_container,
        c_alv TYPE REF TO cl_gui_alv_grid,
* reference variable to CL_DRAGDROP:
        c_dragdropalv TYPE REF TO cl_dragdrop,
        it_layout TYPE lvc_s_layo,
        it_fcat  TYPE lvc_t_fcat.            "Field catalogue

*declarations for handle event
DATA:   effect TYPE i,
        handle_alv TYPE i.

*initialization event

INITIALIZATION.

*start of selection event
START-OF-SELECTION.

*select data
  PERFORM fetch_data.

*ALV output
  PERFORM alv_output.


* Class definitions and method implementation for drag and drop
CLASS lcl_dragdrop DEFINITION.
  PUBLIC SECTION.
    DATA: wa TYPE ty_t016t,
          index TYPE i.   "Index of Line to be moved

ENDCLASS.                    "LCL_DRAGDROP DEFINITION


*Application class definition
CLASS lcl_objdragdropapp DEFINITION.

  PUBLIC SECTION.
```

```
    METHODS:
*Handling Event Drag
      handle_alv_drag
        FOR EVENT ondrag
        OF cl_gui_alv_grid
        IMPORTING e_row e_column e_dragdropobj,

*Handling event DROP
      handle_alv_drop
        FOR EVENT ondrop
        OF cl_gui_alv_grid
        IMPORTING e_row e_column e_dragdropobj.

ENDCLASS.                    "LCL_objdragdropapp DEFINITION

*Application class implementation
CLASS lcl_objdragdropapp IMPLEMENTATION.

*  OnDrag event is used to 'fetch' information from the drag source.
  METHOD handle_alv_drag.
    DATA: dataobj TYPE REF TO lcl_dragdrop,
          line TYPE ty_t016t.
* Read dragged row
    READ TABLE it_t016t INDEX e_row-index INTO line.

* create and fill dataobject for events ONDROP
    CREATE OBJECT dataobj.

* Remembering row index to move a line
    MOVE e_row-index TO dataobj->index.

* store the dragged line.
    READ TABLE it_t016t INTO dataobj->wa INDEX e_row-index.

* Assigning data object to the refering event parameter

    e_dragdropobj->object = dataobj.

  ENDMETHOD.                    "HANDLE_ALV_DRAG

*Event handler for event 'OnDrop'. This event is used
*to use your dragged information in combination with your drop source.

  METHOD handle_alv_drop.

    DATA: dataobj TYPE REF TO lcl_dragdrop,
          drop_index TYPE i,
          stable TYPE lvc_s_stbl.

* Refresh Alv Grid Control without scrolling
    stable-row = 'X'.
    stable-col = 'X'.


* the Catch-Statement to make sure that the drag&drop-Operation is aborted
properly.

    CATCH SYSTEM-EXCEPTIONS move_cast_error = 1.

      dataobj ?= e_dragdropobj->object.

      DELETE it_t016t INDEX dataobj->index.
      INSERT dataobj->wa INTO it_t016t INDEX e_row-index.
```

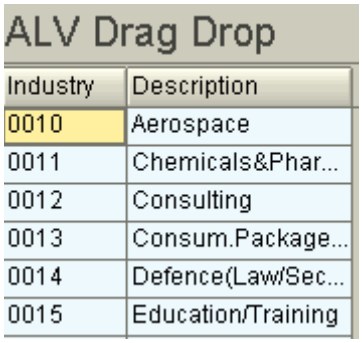
```
*Refreshing the ALV
      CALL METHOD c_alv->refresh_table_display
        EXPORTING
          i_soft_refresh = 'X'
          is_stable      = stable.

    ENDCATCH.
    IF sy-subrc <> 0.

* If anything went wrong aborting the drag and drop operation:

      CALL METHOD e_dragdropobj->abort.

    ENDIF.

  ENDMETHOD.                    "HANDLE_ALV_DROP

ENDCLASS.                    "LCL_objdragdropapp IMPLEMENTATION

*&---------------------------------------------------------------------*
*&      Form  alv_output
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
FORM alv_output .

  CALL SCREEN 600.


ENDFORM.                    " alv_output

** Calling the ALV screen with custom container
*On this statement double click  it takes you to the screen painter SE51.*Enter
the attributes
*Create a Custom container and name it CC_CONT and OK code as OK_CODE.
*Save check and Activate the screen painter.
*Now a normal screen with number 600 is created which holds the ALV grid.
* PBO of the actual screen ,Here we can give a title and customized menus

*&---------------------------------------------------------------------*
*&      Module  STATUS_0600  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE status_0600 OUTPUT.
*  SET PF-STATUS 'xxxxxxxx'.
*  SET TITLEBAR 'xxx'.
  IF c_alv IS INITIAL.
    PERFORM alv_controls.
  ENDIF.
ENDMODULE.                 " STATUS_0600  OUTPUT

*&---------------------------------------------------------------------*
*&      Form  alv_CONTROLS
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
```

```
*---------------------------------------------------------------------*
FORM alv_controls.
* create docking container for alv control
  CREATE OBJECT c_dockingcont
    EXPORTING
        dynnr = '600'
        extension = 300
        side = cl_gui_docking_container=>dock_at_top.


* create alv control
  CREATE OBJECT c_alv
     EXPORTING i_parent = c_dockingcont.

* create the application object to handle the ABAP Objects Events

  CREATE OBJECT c_dragdropapp.

* Events alv control
*For Dragging
  SET HANDLER c_dragdropapp->handle_alv_drag FOR c_alv.
*For Dropping
  SET HANDLER c_dragdropapp->handle_alv_drop FOR c_alv.

* build tree nodes for drag&drop
  PERFORM build_handle.

* Fieldcatalogue for ALV

  PERFORM alv_build_fieldcat.

* ALV attributes FOR LAYOUT

  PERFORM alv_report_layout.


* Call ALV GRID

  CALL METHOD c_alv->set_table_for_first_display
    EXPORTING
      is_layout                     = it_layout
    CHANGING
      it_outtab                     = it_t016t
      it_fieldcatalog               = it_fcat
    EXCEPTIONS
      invalid_parameter_combination = 1
      program_error                 = 2
      too_many_lines                = 3
      OTHERS                        = 4.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
               WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.



ENDFORM.                               "ALV_CONTROLS
*&---------------------------------------------------------------------*
*&      Form  build_handle
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*  -->  p1        text
```

```
*  <--  p2        text
*---------------------------------------------------------------------*

FORM build_handle.


* define a drag & Drop behaviour for the whole grid
  CREATE OBJECT c_dragdropalv.

  effect = cl_dragdrop=>move + cl_dragdrop=>copy.

  CALL METHOD c_dragdropalv->add
    EXPORTING
      flavor     = 'Line'
      dragsrc    = 'X'
      droptarget = 'X'
      effect     = effect.

*getting the handle for drag and drop
  CALL METHOD c_dragdropalv->get_handle
    IMPORTING
      handle = handle_alv.


ENDFORM.                               " build_handle
*&---------------------------------------------------------------------*
*&      Form  fetch_data
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
FORM fetch_data .

* select and display data from t016
  SELECT brtxt brsch spras FROM t016t INTO CORRESPONDING FIELDS OF TABLE
it_t016t
  WHERE spras = 'EN'.


ENDFORM.                    " fetch_data
*&---------------------------------------------------------------------*
*&      Form  alv_report_layout
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
FORM alv_report_layout .

  it_layout-grid_title = 'ALV Drag Drop'.

* provide handle to alv control to all rows for same drag & drop behaviour
  it_layout-s_dragdrop-row_ddid = handle_alv.


ENDFORM.                    " alv_report_layout
*&---------------------------------------------------------------------*
*&      Form  alv_build_fieldcat
*&---------------------------------------------------------------------*
*       text
```

```
*---------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*---------------------------------------------------------------------*
FORM alv_build_fieldcat .

  DATA lv_fldcat TYPE lvc_s_fcat.
  CLEAR lv_fldcat.

  lv_fldcat-row_pos   = '1'.
  lv_fldcat-col_pos   = '1'.
  lv_fldcat-fieldname = 'BRSCH'.
  lv_fldcat-tabname   = 'IT_T016T'.
  lv_fldcat-outputlen = 8.
  lv_fldcat-scrtext_m = 'Industry'.

  APPEND lv_fldcat TO it_fcat.
  CLEAR lv_fldcat.

  lv_fldcat-row_pos   = '1'.
  lv_fldcat-col_pos   = '2'.
  lv_fldcat-fieldname = 'BRTXT'.
  lv_fldcat-tabname   = 'IT_T016T'.
  lv_fldcat-outputlen = 15.
  lv_fldcat-scrtext_m = 'Description'.

  APPEND lv_fldcat TO it_fcat.
  CLEAR lv_fldcat.


ENDFORM.                    " alv_build_fieldcat

* PAI module of the screen created. In case we use an interactive ALV or
*for additional functionalities we can create OK codes
*and based on the user command we can do the coding.

*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0600  INPUT
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
MODULE user_command_0600 INPUT.

ENDMODULE.                 " USER_COMMAND_0600  INPUT
```

Output before drag and drop.



Output after Drag and Drop