*\*&---------------------------------------------------------------------\**

*\*& Report ZTEST\_NP\_EDIT\_SALV*

*\*& Overcome the restriction of the SALV model using the power of the*

*\*& Object Oriented ABAP*

*\*&---------------------------------------------------------------------\**

REPORT ztest\_np\_edit\_salv.

*\*----------------------------------------------------------------------\**

*\* Define the Local class inheriting from the CL\_SALV\_MODEL\_LIST*

*\* to get an access of the model, controller and adapter which inturn*

*\* provides the Grid Object*

*\*----------------------------------------------------------------------\**

CLASS lcl\_salv\_model DEFINITION INHERITING FROM cl\_salv\_model\_list.

PUBLIC SECTION.

DATA: o\_control TYPE REF TO cl\_salv\_controller\_model,

o\_adapter TYPE REF TO cl\_salv\_adapter.

METHODS:

grabe\_model

IMPORTING

io\_model TYPE REF TO cl\_salv\_model,

grabe\_controller,

grabe\_adapter.

PRIVATE SECTION.

DATA: lo\_model TYPE REF TO cl\_salv\_model.

ENDCLASS. *"LCL\_SALV\_MODEL DEFINITION*

*\*----------------------------------------------------------------------\**

*\* Event handler for the added buttons*

*\*----------------------------------------------------------------------\**

CLASS lcl\_event\_handler DEFINITION.

PUBLIC SECTION.

METHODS:

on\_user\_command FOR EVENT added\_function OF cl\_salv\_events

IMPORTING e\_salv\_function.

ENDCLASS. *"lcl\_event\_handler DEFINITION*

*\*----------------------------------------------------------------------\**

*\* Local Report class - Definition*

*\*----------------------------------------------------------------------\**

CLASS lcl\_report DEFINITION.

PUBLIC SECTION.

TYPES: ty\_t\_sflights TYPE STANDARD TABLE OF sflights.

DATA: t\_data TYPE ty\_t\_sflights.

DATA: o\_salv TYPE REF TO cl\_salv\_table.

DATA: o\_salv\_model TYPE REF TO lcl\_salv\_model.

METHODS:

get\_data,

generate\_output.

ENDCLASS. *"lcl\_report DEFINITION*

*\*----------------------------------------------------------------------\**

*\* Global data*

*\*----------------------------------------------------------------------\**

DATA: lo\_report TYPE REF TO lcl\_report.

*\*----------------------------------------------------------------------\**

*\* Start of selection*

*\*----------------------------------------------------------------------\**

START-OF-SELECTION.

CREATE OBJECT lo\_report.

lo\_report->get\_data( ).

lo\_report->generate\_output( ).

*\*----------------------------------------------------------------------\**

*\* Local Report class - Implementation*

*\*----------------------------------------------------------------------\**

CLASS lcl\_report IMPLEMENTATION.

METHOD get\_data.

*\* test data*

SELECT *\* FROM sflights*

INTO TABLE me->t\_data

UP TO 30 ROWS.

ENDMETHOD. *"get\_data*

METHOD generate\_output.

*\*...New ALV Instance ...............................................*

TRY.

cl\_salv\_table=>factory(

EXPORTING

*\* r\_container = w\_alv1*

list\_display = abap\_false

IMPORTING

r\_salv\_table = o\_salv

CHANGING

t\_table = t\_data ).

CATCH cx\_salv\_msg. *"#EC NO\_HANDLER*

ENDTRY.

*\*...PF Status.......................................................*

*\* Add MYFUNCTION from the report SALV\_DEMO\_TABLE\_EVENTS*

o\_salv->set\_screen\_status(

pfstatus = 'SALV\_STANDARD'

REPORT = 'SALV\_DEMO\_TABLE\_EVENTS'

set\_functions = o\_salv->c\_functions\_all ).

*\*...Event handler for the button.....................................*

DATA: lo\_events TYPE REF TO cl\_salv\_events\_table,

lo\_event\_h TYPE REF TO lcl\_event\_handler.

*\* event object*

lo\_events = o\_salv->get\_event( ).

*\* event handler*

CREATE OBJECT lo\_event\_h.

*\* setting up the event handler*

SET HANDLER lo\_event\_h->on\_user\_command FOR lo\_events.

*\*...Get Model Object ...............................................*

DATA: lo\_alv\_mod TYPE REF TO cl\_salv\_model.

*\* Narrow casting*

lo\_alv\_mod ?= o\_salv.

*\* object for the local inherited class from the CL\_SALV\_MODEL\_LIST*

CREATE OBJECT o\_salv\_model.

*\* grabe model to use it later*

CALL METHOD o\_salv\_model->grabe\_model

EXPORTING

io\_model = lo\_alv\_mod.

*\*...Generate ALV output ...............................................*

o\_salv->display( ).

ENDMETHOD. *"generate\_output*

ENDCLASS. *"lcl\_report IMPLEMENTATION*

*\*----------------------------------------------------------------------\**

*\* LCL\_SALV\_MODEL implementation*

*\*----------------------------------------------------------------------\**

CLASS lcl\_salv\_model IMPLEMENTATION.

METHOD grabe\_model.

*\* save the model*

lo\_model = io\_model.

ENDMETHOD. *"grabe\_model*

METHOD grabe\_controller.

*\* save the controller*

o\_control = lo\_model->r\_controller.

ENDMETHOD. *"grabe\_controller*

METHOD grabe\_adapter.

*\* save the adapter from controller*

o\_adapter ?= lo\_model->r\_controller->r\_adapter.

ENDMETHOD. *"grabe\_adapter*

ENDCLASS. *"LCL\_SALV\_MODEL IMPLEMENTATION*

*\*----------------------------------------------------------------------\**

*\* Event Handler for the SALV*

*\*----------------------------------------------------------------------\**

CLASS lcl\_event\_handler IMPLEMENTATION.

METHOD on\_user\_command.

DATA: lo\_grid TYPE REF TO cl\_gui\_alv\_grid,

lo\_full\_adap TYPE REF TO cl\_salv\_fullscreen\_adapter.

DATA: ls\_layout TYPE lvc\_s\_layo.

CASE e\_salv\_function.

*\* Make ALV as Editable ALV*

WHEN 'MYFUNCTION'.

*\* Contorller*

CALL METHOD lo\_report->o\_salv\_model->grabe\_controller.

*\* Adapter*

CALL METHOD lo\_report->o\_salv\_model->grabe\_adapter.

*\* Fullscreen Adapter (Down Casting)*

lo\_full\_adap ?= lo\_report->o\_salv\_model->o\_adapter.

*\* Get the Grid*

lo\_grid = lo\_full\_adap->get\_grid( ).

*\* Got the Grid .. ?*

IF lo\_grid IS BOUND.

*\* Editable ALV*

ls\_layout-edit = 'X'.

*\* Set the front layout of ALV*

CALL METHOD lo\_grid->set\_frontend\_layout

EXPORTING

is\_layout = ls\_layout.

*\* refresh the table*

CALL METHOD lo\_grid->refresh\_table\_display.

ENDIF.

ENDCASE.

ENDMETHOD. *"on\_user\_command*

ENDCLASS. *"lcl\_event\_handler IMPLEMENTATION*

*\* Code comes before the calling the Display method of SALV*

*\* O\_SALV->DISPLAY( )*

*\*...Header.......... ...............................................*

DATA: lo\_header TYPE REF TO cl\_salv\_form\_layout\_grid,

lo\_h\_label TYPE REF TO cl\_salv\_form\_label,

lo\_h\_flow TYPE REF TO cl\_salv\_form\_layout\_flow.

*\* header object CREATE OBJECT lo\_header.*

*\* information in Bold*

lo\_h\_label = lo\_header->create\_label( row = 1 COLUMN = 1 ).

lo\_h\_label->set\_text( 'Header in Bold' ).

*\* information in tabular format*

lo\_h\_flow = lo\_header->create\_flow( row = 2 COLUMN = 1 ).

lo\_h\_flow->create\_text( text = 'This is text of flow' ).

*\* set the top of list using the header for Online.*

o\_salv->set\_top\_of\_list( lo\_header ).

Set the background wallpaper by accessing the GRID and TOP-of-LIST object

*\* Code comes in the event handler of the added button.*

*\* Get the Grid*

lo\_grid = lo\_full\_adap->get\_grid( ).

*\* Got the Grid.. ?*

IF lo\_grid IS BOUND.

DATA: lo\_form\_tol TYPE REF TO cl\_salv\_form,

lo\_dydos TYPE REF TO cl\_salv\_form\_dydos.

*\* Get the Top of list object using the adapter*

CALL FUNCTION 'GET\_GLOBALS\_FROM\_SLVC\_FULLSCR'

EXPORTING

ir\_salv\_fullscreen\_adapter = lo\_full\_adap

IMPORTING

er\_form\_tol = lo\_form\_tol.

*\* Got the Top Of List object.. ?*

IF lo\_form\_tol IS BOUND.

*\* get the object using the Wide Casting*

lo\_dydos ?= lo\_form\_tol.

*\* set the wallpaper*

lo\_dydos->set\_wallpaper( 'ALV\_BACKGROUND' ).

*\* refresh the table*

CALL METHOD lo\_grid->refresh\_table\_display.

ENDIF.

ENDIF.