ALV Object Model – Simple 2D Table – Event Handling

Applies to:
Netweaver 2004 and Netweaver 2004s

Summary
This tutorial will show how to implement event handling when using the new ALV object model. For more examples, see any program which begins with SALV* in your Netweaver ABAP System.

Author(s): Rich Heilman

Company: Yorktowne Cabinetry

Created on: 28 September 2006

Rich Heilman is an ABAP/J2EE Software Engineer/Analyst for Yorktowne Cabinetry, Inc. based in Red Lion, Pennsylvania, USA. He has a total of nine years experience in the IT industry. He has spent the past five years studying ABAP and Java.
# Table of Contents

Applies to: ........................................................................................................................................ 1  
Summary.......................................................................................................................................... 1  
The Basic Program.......................................................................................................................... 3  
Set the Gui Status............................................................................................................................ 4  
Event Handlers - Event ADDED_FUNCTION .................................................................................. 5  
Event Handlers - Event DOUBLE_CLICK ......................................................................................... 8  
Related Content............................................................................................................................. 11  
Disclaimer and Liability Notice........................................................................................................ 12
The Basic Program
Starting with the program below, we will add coding to handle some events for the ALV Grid. In this example the events DOUBLE_CLICK and ADDED_FUNCTION will be handled.

```abap
report zalvom_demo3.

data: ispfli type table of spfli.
data: xspfli type spfli.
data: gr_table type ref to cl_salv_table.
data: gr_selections type ref to cl_salv_selections.

start-of-selection.

select * into corresponding fields of table ispfli from spfli up to 100 rows.

call method cl_salv_table=>factory
importing
  r_salv_table = gr_table
changing
  t_table = ispfli.

* Set up selections.
gr_selections = gr_table->get_selections( ).
gr_selections->set_selection_mode( 1 ). "Single

* Display
gr_table->display( ).
```
Set the Gui Status

Next, go to function group SALV_METADATA_STATUS and copy the gui status SALV_TABLE_STANDARD into the ZALVOM_DEMO3 program. This is the standard gui status for the 2 Dimensional Table ALV grid. Once you have copied the status, set the screen status using the appropriate method of the object GR_TABLE. Go to the gui status and add a new button on the application toolbar and name it as "MYFUNCTION".

```plaintext
report zalvom_demo3.

data: ispfli type table of spfli.
data: xspfli type spfli.
data: gr_table type ref to cl_salv_table.
data: gr_selections type ref to cl_salv_selections.

start-of-selection.

    select * into corresponding fields of table ispfli from spfli 
        up to 100 rows.

    call method cl_salv_table=>factory 
       importing 
         r_salv_table = gr_table 
       changing 
         t_table      = ispfli.

    gr_table->set_screen_status( 
        pfstatus      = 'SALV_TABLE_STANDARD' 
        report        = sy-repid 
        set_functions = gr_table->c_functions_all ).

* Set up selections.
    gr_selections = gr_table->get_selections( ).
    gr_selections->set_selection_mode(1). "Single"

* Display
    gr_table->display( ).
```
Event Handlers - Event ADDED_FUNCTION

Next, create a local class which will act as the event handler, define the event handler method for the ADDED_FUNCTION event. Define an object reference variable for the local class. Retrieve the events object from the GR_TABLE, create the event handler object and set the handler method for the event. Finally, add the implementation for the ON_USER_COMMAND event handler method.

```
REPORT zalvom_demo3.

DATA:  ISPFLI TYPE TABLE OF SPFLI.
DATA:  XSPFLI TYPE SPFLI.

DATA:  GR_TABLE TYPE REF TO CL_SALV_TABLE.
DATA:  GR_EVENTS TYPE REF TO CL_SALV_EVENTS_TABLE.
DATA:  GR_SELECTIONS TYPE REF TO CL_SALV_SELECTIONS.

CLASS lcl_handle_events DEFINITION.
  PUBLIC SECTION.
  METHODS:
    ON_USER_COMMAND FOR EVENT ADDED_FUNCTION OF CL_SALV_EVENTS
    IMPORTING e_salv_function,
  ENDCLASS.  "lcl_handle_events DEFINITION

DATA:  EVENT_HANDLER TYPE REF TO LCL_HANDLE_EVENTS.

START-OF-SELECTION.

SELECT * INTO CORRESPONDING FIELDS OF TABLE ISPFLI FROM SPFLI
  UP TO 100 ROWS.

CALL METHOD CL_SALV_TABLE=>FACTORY
  IMPORTING
    r_salv_table = GR_TABLE
  CHANGING
    t_table = ISPFLI.

GR_TABLE->SET_SCREEN_STATUS((pfstatus = 'SALV_TABLE_STANDARD'
  REPORT = SY-REPID
  SET_FUNCTIONS = GR_TABLE->C_FUNCTIONS_ALL).

GR_EVENTS = GR_TABLE->GET_EVENT( ).

CREATE OBJECT EVENT_HANDLER.

SET HANDLER EVENT_HANDLER->ON_USER_COMMAND FOR GR_EVENTS.

* Set up selections.
```
gr_selections = gr_table->get_selections( ).
gr_selections->set_selection_mode( 1 ).   "Single

* Display
gr_table->display( ).

*----------------------------------------------------------------------*
*       CLASS lcl_handle_events IMPLEMENTATION
*----------------------------------------------------------------------*
*
*----------------------------------------------------------------------*
class lcl_handle_events implementation.
method on_user_command.
* Get the selection rows
  data: lr_selections type ref to cl_salv_selections.
  data: lt_rows  type salv_t_row.
  data: ls_rows  type i.
  data: message type string.

  case e_salv_function.
    when 'MYFUNCTION'.

      lr_selections = gr_table->get_selections( ).
      lt_rows = lr_selections->get_selected_rows( ).

      read table lt_rows into ls_rows index 1.
      read table ispfli into xspfli index ls_rows.

      concatenate xspfli-carrid   xspfli-connid
                   xspfli-cityfrom xspfli-cityto
            into message separated by space.

      message i001(00) with 'You pushed the button!' message.

    endcase.

  endmethod.                    "on_user_command

endclass.                    "lcl_handle_events IMPLEMENTATION
Run the program, select a row by single clicking on it and click the icon for the new function that you added. Notice that some of the data in the row that was clicked is now showing in the message.
**Event Handlers - Event DOUBLE_CLICK**

Define the event handler method for DOUBLE_CLICK event and add the implementation for the ON_DOUBLE_CLICK event handler method. Remember to set the handler for the event.

```abap
report zalvom_demo3.

data: ispfli type table of spfli.
data: xspfli type spfli.
data: gr_table type ref to cl_salv_table.
data: gr_functions type ref to cl_salv_functions_list.
data: gr_events type ref to cl_salv_events_table.
data: gr_selections type ref to cl_salv_selections.

*----------------------------------------------------------------------*
*       CLASS lcl_handle_events DEFINITION
*----------------------------------------------------------------------*

class lcl_handle_events definition.
  public section.
  methods:
    on_user_command for event added_function of cl_salv_events
      importing e_salv_function,
      on_double_click for event double_click of cl_salv_events_table
      importing row column.
  endclass.

data: event_handler type ref to lcl_handle_events.

start-of-selection.

  select * into corresponding fields of table ispfli from spfli
    up to 100 rows.

  call method cl_salv_table=>factory
    importing
      r_salv_table = gr_table
    changing
      t_table      = ispfli.

  gr_table->set_screen_status( 
    pfstatus       = 'SALV_TABLE_STANDARD'
    report         = sy-repid
    set_functions  = gr_table->c_functions_all ).

  gr_events = gr_table->get_event( ).

  create object event_handler.

  set handler event_handler->on_user_command for gr_events.
```
set handler event_handler->on_double_click for gr_events.

* Set up selections.
  gr_selections = gr_table->get_selections( ).
  gr_selections->set_selection_mode(1). "Single

* Display
  gr_table->display( ).

*----------------------------------------------------------------------*
*       CLASS lcl_handle_events IMPLEMENTATION
*----------------------------------------------------------------------*

*----------------------------------------------------------------------*
class lcl_handle_events implementation.

method on_user_command.

* Get the selection rows
  data: lr_selections type ref to cl_salv_selections.
  data: lt_rows type salv_t_row.
  data: ls_rows type i.
  data: message type string.

  case e_salv_function.
    when 'MYFUNCTION'.
      lr_selections = gr_table->get_selections( ).
      lt_rows = lr_selections->get_selected_rows( ).
      read table lt_rows into ls_rows index 1.
      read table ispfli into xspfli index ls_rows.
      concatenate xspfli-carrid xspfli-connid xspfli-cityfrom xspfli-cityto into message separated by space.
      message i001(00) with 'You pushed the button!' message.
  endcase.
endmethod. "on_user_command

method on_double_click.

  data: message type string.
  data: row_c(4) type c.

  row_c = row.
  concatenate 'Row' row_c 'Column' column into message separated by space.
  message i001(00) with 'You double-clicked on ' message.
endmethod. "on_double_click
endclass. "lcl_handle_events IMPLEMENTATION
Run the program, double click on the fifth row in the Depart. City column, notice the information message contains the row number and column name of the cell which you double clicked.

### ALV OM Demo 3

<table>
<thead>
<tr>
<th>Cli...</th>
<th>Air...</th>
<th>Flig...</th>
<th>Co...</th>
<th>Depart. city</th>
<th>De...</th>
<th>Co...</th>
<th>Arrival city</th>
<th>Ta...</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>AA</td>
<td>17</td>
<td>US</td>
<td>NEW YORK</td>
<td>JFK</td>
<td>US</td>
<td>SAN FRANCISCO</td>
<td>SFO</td>
</tr>
<tr>
<td>000</td>
<td>AA</td>
<td>64</td>
<td>US</td>
<td>SAN FRANCISCO</td>
<td>SFO</td>
<td>US</td>
<td>NEW YORK</td>
<td>JFK</td>
</tr>
<tr>
<td>000</td>
<td>AZ</td>
<td>555</td>
<td>IT</td>
<td>ROME</td>
<td>FCO</td>
<td>DE</td>
<td>FRANKFURT</td>
<td>FRA</td>
</tr>
<tr>
<td>000</td>
<td>AZ</td>
<td>788</td>
<td>IT</td>
<td>ROME</td>
<td>FCO</td>
<td>JP</td>
<td>TOKYO</td>
<td>TYO</td>
</tr>
<tr>
<td>000</td>
<td>AZ</td>
<td>789</td>
<td>JP</td>
<td>TOKYO</td>
<td>TYO</td>
<td>IT</td>
<td>ROME</td>
<td>FCO</td>
</tr>
<tr>
<td>000</td>
<td>AZ</td>
<td>790</td>
<td>IT</td>
<td>ROME</td>
<td>FCO</td>
<td>JP</td>
<td>OSAKA</td>
<td>KIX</td>
</tr>
<tr>
<td>000</td>
<td>D</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You double-clicked on Row 5 Column CITYFROM.
Related Content

Help - ALV Object Model

Utilizing the New ALV Object Model

SDN ABAP Forum
Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. 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 the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

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