TextEdit Control

Contributed by Henrik Frank
Thursday, 01 February 2007

TIP: Use SE75 to create your own custom text ID for SAVE_TEXT object
- Example 1: Creating the TextEdit control
- Example 2: Event handling - Application event
- Example 3: Event handling - System event
- Example 4: Calling a methods of the control
- Example 5: Responding to an event
- Example 6: Protect a line in the TextEdit control and the importance of FLUSH
- Example 7: Using multiple controls

See the whole program codeExample 1: Creating the TextEdit control

This is a simple example of how to implement a text edit control.

StepsCreate a reportIn the start of selection event add: SET SCREEN '100'.Create screen 100Place a custom control on the screen by choosing the custom control icon which can be recognized by the letter 'C', and give it the name MYCONTAINER1.To be able to exit the program, add a pushbutton with the function code EXIT.In the elements list enter the name OK_CODE for the element of type OK.

The code REPORT sapmz_hf_controls1 . CONSTANTS: line_length TYPE i VALUE 254. DATA: ok_code LIKE sy-ucomm. DATA: * Create reference to the custom container custom_container TYPE REF TO cl_gui_custom_container, * Create reference to the TextEdit control editor TYPE REF TO cl_gui_textedit, repid LIKE sy-repid. START-OF-SELECTION. SET SCREEN '100'. *---*
 * MODULE USER_COMMAND_0100 INPUT *

 MODULE user_command_0100 INPUT. CASE ok_code. WHEN 'EXIT'. LEAVE TO SCREEN 0. ENDCASE. ENDMODULE. " USER_COMMAND_0100 INPUT *&---*
 *& Module STATUS_0100 OUTPUT
 &---
 MODULE status_0100 OUTPUT. * The TextEdit control should only be initialized the first time the
 * PBO module executes
  IF editor IS INITIAL. repid = sy-repid. * Create obejct for custom container CREATE OBJECT custom_container EXPORTING container_name = 'MYCONTAINER1' EXCEPTIONS cntl_error = 1 cntl_syst = 3 lifetime_error = 4 lifetime_dynpro_dynpro_link = 5 others = 6 . IF sy-subrc <> 0. NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4. ENDIF. * Create obejct for the TextEditor cor CREATE OBJECT editor EXPORTING wordwrap_mode = cl_gui_textedit=>wordwrap_at_fixed_positio wordwrap_position = line_length wordwrap_to_linebreak_mode = cl_gui_textedit=>true parent = custom EXCEPTIONS error_cntl_create = 1 error_cntl_init = 2 error_cntl_link = 3 error_dp_create = gui_type_not_supported = 5 others = 6 . IF sy-subrc <> 0. MESSAGE ID sy-msgid TYPE 'I' NUMBER WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4. ENDIF. ENDIF. ENDMODULE. " STATUS_0100 OUTPUTExamp
Event handling - Application eventThere are 2 types of events:
System events. These events are triggerede irrespective of the screen flow-logic.
Application events. The PAI module is processed after an event. The method CL_GUI_CFW=>DISPATCH must be called to initiate event handling
In this example an application event is added to the program in example 1. New code is marked with red.
Steps:
Create an input/output field on screen 100, where the event type can be output. Name it EVENT_TYPE
The code:

```
REPORT sapmz_hf_controls1 .
CONSTANTS:
  line_length TYPE i VALUE 254.
DATA: ok_code LIKE sy-ucomm.
DATA:
* Create reference to the custom container
  custom_container TYPE REF TO cl_gui_custom_container,
* Create reference to the TextEdit control
  editor TYPE REF TO cl_gui_textedit,
  repid LIKE sy-repid.
***********************************************************************
```

```
* Impmenting events
***********************************************************************
DATA:
  event_type(20) TYPE c,

* Internal table for events that should be registred
  i_events TYPE cntl_simple_events,

* Structure for oneline of the table
  wa_events TYPE cntl_simple_event.
*---------------------------------------------------------------------*
*       CLASS lcl_event_handler DEFINITION
*---------------------------------------------------------------------*
CLASS lcl_event_handler DEFINITION.

  PUBLIC SECTION.
   CLASS-METHODS:
     catch_dblclick FOR EVENT dblclick
       OF cl_gui_textedit IMPORTING sender.
ENDCLASS.
CLASS lcl_event_handler IMPLEMENTATION.

  METHOD catch_dblclick.
    event_type = 'Event DBLCLICK raised'.
  ENDMETHOD.

ENDCLASS.

START-OF-SELECTION.
  CLEAR wa_events. refresh i_events.
  SET SCREEN '100'.

*---------------------------------------------------------------------*
*       MODULE USER_COMMAND_0100 INPUT                                *
*---------------------------------------------------------------------*
MODULE user_command_0100 INPUT.
  CASE ok_code.
    WHEN 'EXIT'.
      LEAVE TO SCREEN 0.
    WHEN OTHERS.
*   Call the Dispacth method to initiate application event handling

    call method cl_gui_cfw=>Dispatch.

  ENDCASE.
ENDMODULE.                 " USER_COMMAND_0100  INPUT
*&---------------------------------------------------------------------*
*&      Module  STATUS_0100  OUTPUT
*&---------------------------------------------------------------------*
MODULE status_0100 OUTPUT.
* The TextEdit control shoul only be initialized the first time the
* PBO module executes
  IF editor IS INITIAL.
    repid = sy-repid.
*   Create obejct for custom container
    CREATE OBJECT custom_container
      EXPORTING
        container_name              = 'MYCONTAINER1'
      EXCEPTIONS
        cntl_error                  = 1
        cntl_system_error           = 2
        create_error                = 3
        lifetime_error              = 4
        lifetime_dynpro_dynpro_link = 5
        others                      = 6
```

```
      .
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE 'I' NUMBER sy-msgno
              WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.
*   Create obejct for the TextEditor control
    CREATE OBJECT editor
      EXPORTING
        wordwrap_mode          =
              cl_gui_textedit=>wordwrap_at_fixed_position
        wordwrap_position      = line_length
        wordwrap_to_linebreak_mode = cl_gui_textedit=>true
       parent                  = custom_container
      EXCEPTIONS
        error_cntl_create      = 1
        error_cntl_init        = 2
        error_cntl_link        = 3
        error_dp_create        = 4
        gui_type_not_supported = 5
        others                 = 6
      .
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE 'I' NUMBER sy-msgno
              WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.
*   Link the event handler method to the event and the
*   TextEdit control
    SET HANDLER lcl_event_handler=>catch_dblclick FOR editor.
*   Register the event in the internal table i_events
    wa_events-eventid = cl_gui_textedit=>event_double_click.
    wa_events-appl_event = 'X'. "This is an application event
    append wa_events to i_events.
*   Pass the table to the TextEdit control using method
*   set_registred_events
    call method editor->set_registered_events
       exporting events = i_events.
  ENDIF.
ENDMODULE.                 " STATUS_0100  OUTPUT
```

Result:

When you double click on the TextEdit control, the input/ouput field should show the text: Event DBLCLICK

Example 3: Event handling - System eventSystem events are passed irrespective of the flow-logic of the screen. To implement a system event change the code from example 2 as follows:

Code:

```
CLASS lcl_event_handler IMPLEMENTATION.
  METHOD catch_dblclick.
*---  event_type = 'Event DBLCLICK raised'.
* Reacting to the system event
  call method cl_gui_cfw=>set_new_ok_code
   exporting new_code = 'SHOW'.
MODULE user_command_0100 INPUT.
  CASE ok_code.
   code.........
    WHEN 'SHOW'.
      event_type = 'System dblclick'.
    WHEN OTHERS.
*----    call method cl_gui_cfw=>Dispatch.
  ENDCASE.
ENDMODULE.                 " USER_COMMAND_0100  INPUT

MODULE status_0100 OUTPUT.
 Code ................
*---    wa_events-appl_event = 'X'. "This is an application event
    wa_events-appl_event = space. "This is a system event
  ENDIF.
ENDMODULE.                 " STATUS_0100  OUTPUT
```

Result:
When you double clicks on the TextEdit control nothing happens, since the flow-logic of the screen and the fielde transport is ignore.
Example 4: Calling methods of the controlIn this exercise a function that loads the texts of an internal table into the text window, is implemented.
Steps:
Define anoterh pushbutton on the screen, that activates the method that fills the TextEdit control. Give itname PUSHBUTTON_IMPORT and function code IMP.
Define a form CREATE_TEXTS that carries out the text import.
Only changes to the code in example 2 is show.
Code:

```
MODULE user_command_0100 INPUT.
  CASE ok_code.
   code.........
    WHEN 'IMP'.
      perform load_texts.
  ENDCASE.
ENDMODULE.                " USER_COMMAND_0100  INPUT
*&---------------------------------------------------------------------*
*&      Form  load_texts
*&---------------------------------------------------------------------*
* This form creates an internal table with texts. The the contents of
* the table is instered into the TextEdit control using method
* set_text_as_r3table
*---------------------------------------------------------------------*
FORM load_texts.
  TYPES:
   BEGIN OF t_texttable,
     line(line_length) TYPE c,
   END OF t_texttable.
  DATA
    i_texttable TYPE TABLE OF t_texttable.
* Create internal table with texts
  APPEND 'This a method that fills the TextEdit control' TO i_texttable.
  APPEND 'with a text.' TO i_texttable.
  DO 10 TIMES.
    APPEND 'hallo world !' TO i_texttable.
  ENDDO.
* Load TextEdit control with texts
  CALL METHOD editor->set_text_as_r3table
    EXPORTING table = i_texttable.
  IF sy-subrc > 0.
*   Display an error message
    EXIT.
  ENDIF.
* All methods that operates on controls are transferred to the frontend
* by a RFC calls. the method FLUSH is used to determine when this is done.
  CALL METHOD cl_gui_cfw=>flush.
  IF sy-subrc > 0.
*   Display an error message
  ENDIF.
ENDFORM.                  " create_texts
```

Example 5: Responding to an eventWhen you double click on a text line in the TextEdit control, you want it to be prefixed with a '*'.
The line number of the TextEdit control that is double clicked, is retreived using method GET_SELECTION_POS. The internal text table is reloaded froim the TextEdit control with method GET_TEXT_AS_R3TABLE. The position of the double click in the TextEdit control is used to find the entry in the table, and the entry is prefixed with '*' and loaded into the TextEdit control again.
The program should be changed so that the internal table i_texttable is global, and a global flag g_loaded added. The load of the table should be moved to the PBO module. The changes in thje code are marked with red. The whole program now looks like this:
Code

```
REPORT sapmz_hf_controls1 .
CONSTANTS:
```

```
  line_length TYPE i VALUE 254.
DATA: ok_code LIKE sy-ucomm.
DATA:
* Create reference to the custom container
  custom_container TYPE REF TO cl_gui_custom_container,
* Create reference to the TextEdit control
  editor TYPE REF TO cl_gui_textedit,
  repid LIKE sy-repid.
***********************************************************************
* Utillity table to load texts
***********************************************************************
TYPES:
   BEGIN OF t_texttable,
     line(line_length) TYPE c,
   END OF t_texttable.
DATA:
  i_texttable TYPE TABLE OF t_texttable,
  g_loaded(1) TYPE c.


***********************************************************************
* Impmenting events
***********************************************************************
DATA:
  event_type(20) TYPE c,
* Internal table for events that should be registred
  i_events TYPE cntl_simple_events,
* Structure for oneline of the table
  wa_events TYPE cntl_simple_event.
*---------------------------------------------------------------------*
*       CLASS lcl_event_handler DEFINITION
*---------------------------------------------------------------------*
CLASS lcl_event_handler DEFINITION.
  PUBLIC SECTION.
    CLASS-METHODS:
      catch_dblclick FOR EVENT dblclick
         OF cl_gui_textedit IMPORTING sender.
ENDCLASS.
CLASS lcl_event_handler IMPLEMENTATION.
  METHOD catch_dblclick.
    DATA:
      from_line TYPE i,
      from_pos  TYPE i,
      to_line TYPE i,
      to_pos TYPE i,
      wa_texttable TYPE t_texttable.
* Used for the sytem event
  call method cl_gui_cfw=>set_new_ok_code
    exporting new_code = 'SHOW'.
* Read the position of the double click
    CALL METHOD sender->get_selection_pos
     IMPORTING
        from_line = from_line
        from_pos  = from_pos
        to_line   = to_line
        to_pos    = to_pos.
*   Texts in the TextEdit control can have been changed, so
*   first reload text from the control into the internal
*   table that contains text
  IF NOT g_loaded IS INITIAL.
      CALL METHOD sender->get_text_as_r3table
        IMPORTING table = i_texttable.
*   Read the line of the internal table that was clicked
      READ TABLE i_texttable INDEX from_line INTO wa_texttable.
      IF sy-subrc <> 0.
```

```
        EXIT.
      ENDIF.
      IF wa_texttable+0(1) CS '*'.
        SHIFT wa_texttable.
      ELSEIF wa_texttable+0(1) NS '*'.
        SHIFT wa_texttable RIGHT.
        wa_texttable+0(1) = '*'.
      ENDIF.

      modify i_texttable from wa_texttable index from_line.
*     Reload texts from h einternal table
      perform load_texts.
    ENDIF.
  ENDMETHOD.
ENDCLASS.


START-OF-SELECTION.
  CLEAR wa_events.
  REFRESH: i_events.
  SET SCREEN '100'.
*---------------------------------------------------------------------*
*       MODULE USER_COMMAND_0100 INPUT                                *
*---------------------------------------------------------------------*
MODULE user_command_0100 INPUT.
  CASE ok_code.
    WHEN 'EXIT'.
      LEAVE TO SCREEN 0.
    WHEN 'SHOW'.
      event_type = 'System dblclick'.
    WHEN 'IMP'.
      PERFORM Load_texts.
    WHEN OTHERS.
*    CALL METHOD cl_gui_cfw=>dispatch. "Not used for system events
  ENDCASE.
ENDMODULE.                 " USER_COMMAND_0100  INPUT
*&---------------------------------------------------------------------*
*&      Module  STATUS_0100  OUTPUT
*&---------------------------------------------------------------------*
MODULE status_0100 OUTPUT.
* The TextEdit control shoul only be initialized the first time the
* PBO module executes
  IF editor IS INITIAL.
    repid = sy-repid.
*   Create object for custom container
    CREATE OBJECT custom_container
      EXPORTING
        container_name              = 'MYCONTAINER1'
      EXCEPTIONS
        cntl_error                  = 1
        cntl_system_error           = 2
        create_error                = 3
        lifetime_error              = 4
        lifetime_dynpro_dynpro_link = 5
        others                      = 6
        .
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE 'I' NUMBER sy-msgno
                 WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.
*   Create obejct for the TextEditor control
    CREATE OBJECT editor
      EXPORTING
         wordwrap_mode         =
                cl_gui_textedit=>wordwrap_at_fixed_position
```

```
        wordwrap_position      = line_length
        wordwrap_to_linebreak_mode = cl_gui_textedit=>true
       parent                  = custom_container
     EXCEPTIONS
       error_cntl_create      = 1
       error_cntl_init        = 2
       error_cntl_link        = 3
       error_dp_create        = 4
       gui_type_not_supported = 5
       others                 = 6
       .
   IF sy-subrc <> 0.
     MESSAGE ID sy-msgid TYPE 'I' NUMBER sy-msgno
               WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
   ENDIF.
*   Link the event handler method to the event and the
*   TextEdit control
   SET HANDLER lcl_event_handler=>catch_dblclick FOR editor.
*   Register the event in the internal table i_events
   wa_events-eventid = cl_gui_textedit=>event_double_click.
*    wa_events-appl_event = 'X'. "This is an application event
   wa_events-appl_event = space. "This is a system event
   APPEND wa_events TO i_events.
*   Pass the table to the TextEdit control uding method
*   set_registred_events
   CALL METHOD editor->set_registered_events
      EXPORTING events = i_events.
* Create internal table with texts taht can be uploaded to
* the TextEdit control
  APPEND 'This a method that fills the TextEdit control' TO i_texttable.
   APPEND 'with a text.' TO i_texttable.
   DO 10 TIMES.
     APPEND 'hallo world !' TO i_texttable.
   ENDDO.

  ENDIF.
ENDMODULE.                 " STATUS_0100  OUTPUT
*&---------------------------------------------------------------------*
*&      Form Load_texts
*&---------------------------------------------------------------------*
* This form loads the lines of the internal table i_texttable into
* the TextEdit control
*----------------------------------------------------------------------*
FORM Load_texts.
* Load TextEdit control with texts
  CALL METHOD editor->set_text_as_r3table
    EXPORTING table = i_texttable.
  IF sy-subrc > 0.
*   Display an error message
    EXIT.
  ENDIF.
* All methods that operates on controls are transferred to the frontend
* by a RFC calls. the method FLUSH is used to determine when this is
* done.
  CALL METHOD cl_gui_cfw=>flush.
  IF sy-subrc > 0.
*   Display an error message
  ENDIF.
  g_loaded = 'X'.
ENDFORM.                    " create_texts
```

Example 6: Protect a line in the TextEdit control and the importance of FLUSHAll methods that operates on controls are transfered to the fronend by RFC calls. The FLUSH method is used to synchronize control execution and the frontend. This is very important when working e.g. with export parameters from a method, as the parmeters will not be correct before the FLUSH method has been called.

The example below portects selected lines in the TextEdit and uses FLUSH to ensure that the correct parameters are

returned from method GET_SELECTION_POS.
Note: Instead of using method PROTECT_LINES, the method PROTECT_SELECTION could be used. This method does not need line numbers or a FLUSH statement
Steps
Add a new pushbutton to the screen with the function code PROTECT.
Code
Add the following code to the example:

```
* Global variables

DATA:
   from_idx TYPE i,
   to_idx TYPE i,
   index TYPE i.
MODULE user_command_0100 INPUT.
  CASE ok_code.
    code.......................
    WHEN 'PROTECT'.
      PERFORM protect.
    .......................
  ENDCASE.

*&---------------------------------------------------------------------*
*&      Form  protect
*&---------------------------------------------------------------------*
* Protects marked lines in a TextEdit control
*---------------------------------------------------------------------*
FORM protect.
* Determine the area selected by the user
  CALL METHOD editor->get_selection_pos
     IMPORTING
       from_line = from_idx
       to_line   = to_idx
     EXCEPTIONS
       error_cntl_call_method = 1.
* Synchronize execution in the control with the ABAP program.
* Without this synchronization the variables from_idx and
* to_idx will have obsolutete values (The initial value for
* both, are 0)
  CALL METHOD cl_gui_cfw=>flush.
  IF sy-subrc > 0.
* Errormessage: Error in flush
  ENDIF.
* Protect the selected lines
  IF to_idx > from_idx.
    to_idx = to_idx - 1.
  ENDIF.
  CALL METHOD editor->protect_lines
    EXPORTING
      from_line = from_idx
      to_line   = to_idx.
* The PROTECT_SELECTION method could be used instead, eliminating the
* need of line numbers and the last FLUSH
* call method editor->protect_selection.
* Flush again to protect immidiately


  CALL METHOD cl_gui_cfw=>flush.
  IF sy-subrc > 0.
* Errormessage: Error in flush
  ENDIF.

ENDFORM.                    " protect
```

Example 7: Using multiple controlsIn this example a second TextEdit control will be added to the screen. The new TextEdit control will be designed to act as a clipboard for short texts.

Steps:
Add a new container to the screen and name it MYCONTAINER2.

Code:
Insert global datadeclaration:

```
**********************************************************************
* Implementing a second Scratch TextEdit control
**********************************************************************


DATA:
  scratch            TYPE REF TO cl_gui_textedit,
  custom_container2  TYPE REF TO cl_gui_custom_container.
```

Insert the following code in the PBO module:

```
*-----------------------------------------------------
* The SCRATCH TextEdit control
*-----------------------------------------------------

  IF scratch IS INITIAL.
*   Create obejct for custom container2
    CREATE OBJECT custom_container2
      EXPORTING
        container_name              = 'MYCONTAINER2'
      EXCEPTIONS
        cntl_error                  = 1
        cntl_system_error           = 2
        create_error                = 3
        lifetime_error              = 4
        lifetime_dynpro_dynpro_link = 5
        others                      = 6
        .
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE 'I' NUMBER sy-msgno
              WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.
*   Create obejct for the SCRATCH TextEditor control
    CREATE OBJECT scratch
      EXPORTING
        parent         = custom_container2
        wordwrap_mode  =
              cl_gui_textedit=>wordwrap_at_windowborder
        wordwrap_to_linebreak_mode = cl_gui_textedit=>true.
*   Remove the staus bar
    CALL METHOD scratch->set_statusbar_mode
      EXPORTING statusbar_mode = cl_gui_textedit=>false.
  ENDIF.
```