Event NODE_KEYPRESS

Mit dem Ereignis NODE_KEYPRESS kann man im Programm auf verschiedene Tastendrücke reagieren: F1, F4, Enter, Einfügen, Löschen, STRG+X, STRG+C, STRG+V.

Beschreibung

Das folgende Programm generiert einen kleinen Baum mithilfe des Control CL_GUI_SIMPLE_TREE. Bei Programmstart wird der Focus sofort auf das TreeControl gesetzt, so dass der Anwender mithilfe der Cursortasten in dem Baum navigieren kann.

Steht der Cursor auf einem Eintrag, so können die Tasten „F1“, „Enter“ oder „Löschen“ gedrückt werden. Das Ereignis NODE_KEYPRESS muss dafür natürlich registriert werden.

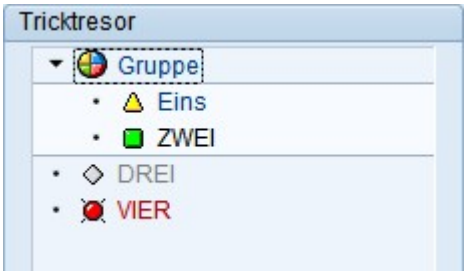
Zusätzlich müssen aber die Tasten, auf die der Tree reagieren soll auch noch bekannt gegeben werden. Das passiert mit der Methode ADD_KEY_STROKE der Klasse CL_GUI_SIMPLE_TREE.

Tasten

Folgende Tasten können registriert werden:

- CL_TREE_CONTROL_BASE=>KEY_**F1**: Funktionstaste F1
- CL_TREE_CONTROL_BASE=>KEY_**F4**: Funktionstaste F4
- CL_TREE_CONTROL_BASE=>KEY_**INSERT**: Taste Einfügen
- CL_TREE_CONTROL_BASE=>KEY_**DELETE**: Taste Entfernen
- CL_TREE_CONTROL_BASE=>KEY_**ENTER**: Enter-Taste
- CL_TREE_CONTROL_BASE=>KEY_**CUT**: Strg+X
- CL_TREE_CONTORL_BASE=>KEY_**COPY**: Strg+C
- CL_TREE_CONTROL_BASE=>KEY_**PASTE**: Strg+V

Screenshot



Der kleine Demo-Baum

Code

```
REPORT  zz_tree_keypress_demo.
*** we use icons...
TYPE-POOLS icon.

*** global data:
*== Custom Container
DATA gr_cuco TYPE REF TO cl_gui_custom_container.
*== Simple Tree
DATA gr_tree TYPE REF TO cl_gui_simple_tree.

*---------------------------------------------------------------------*
*       CLASS lcl_event_handler_class DEFINITION
*---------------------------------------------------------------------*
CLASS lcl_event_handler_class DEFINITION.
PUBLIC SECTION.
CLASS-METHODS handle_keypress
FOR EVENT node_keypress   OF cl_gui_simple_tree
IMPORTING node_key key.
ENDCLASS.                    „lcl_event_handler_class DEFINITION

*---------------------------------------------------------------------*
*       CLASS lcl_event_handler_class IMPLEMENTATION
*---------------------------------------------------------------------*
CLASS lcl_event_handler_class IMPLEMENTATION.
METHOD handle_keypress.
IF node_key IS NOT INITIAL.
*** we only get a node id for active nodes (not disabled):
MESSAGE i000(oo) WITH ‚Knoten:‘ node_key ‚- Taste:‘ key.
ENDIF.
ENDMETHOD.                    „message
ENDCLASS.                    „lcl_event_handler_class IMPLEMENTATION

START-OF-SELECTION.

*** Call screen
CALL SCREEN 1.

*---------------------------------------------------------------------*
*  MODULE status_0001 OUTPUT
*---------------------------------------------------------------------*
MODULE status_0001 OUTPUT.

*** Set status
SET PF-STATUS ‚STLI‘ OF PROGRAM ‚SAPMSSY0‘.
*** Init Controls
PERFORM init_controls.

ENDMODULE.                 “ STATUS_0100  OUTPUT

*---------------------------------------------------------------------*
*  MODULE user_command_0001 INPUT
*---------------------------------------------------------------------*
```

```
MODULE user_command_0001 INPUT.

  CASE sy-ucomm.
WHEN ‚BACK‘ OR ‚%EX‘ OR ‚RW‘.
SET SCREEN 0.
LEAVE SCREEN.
ENDCASE.

ENDMODULE.                 “ USER_COMMAND_0100  INPUT

*&———————————————————————*
*&      Form  init_controls
*&———————————————————————*
* – create custom container
* – create tree control
* – register events
* – build tree structure
* – set focus on tree control
*———————————————————————*
FORM init_controls.

*** initial build up
  CHECK gr_cuco IS INITIAL.

*** create custom container
CREATE OBJECT gr_cuco
EXPORTING
container_name = ‚TREE‘.

*** Create simple tree; single node selection
CREATE OBJECT gr_tree
EXPORTING
parent              = gr_cuco
node_selection_mode = cl_gui_simple_tree=>node_sel_mode_single.

*** register events
PERFORM register_events.

*** build tree structure
PERFORM build_tree.

*** Set focus on tree control so that user instantly can
*** navigate with cursor keys
cl_gui_control=>set_focus( gr_tree ).

ENDFORM.                    „init_controls

*&———————————————————————*
*&      Form  register_events
*&———————————————————————*
FORM register_events.

*** Data
```

```
DATA: lt_events TYPE cntl_simple_events,
ls_event  TYPE cntl_simple_event.

*** NODE_KEYPRESS-Event
ls_event-eventid = cl_gui_simple_tree=>eventid_node_keypress.
APPEND ls_event TO lt_events.

*** Register Event
CALL METHOD gr_tree->set_registered_events
EXPORTING
events = lt_events.

*** tell tree which keys to mention for NODE_KEPRESS:
*==> ENTER
gr_tree->add_key_stroke( cl_gui_simple_tree=>key_enter )  .
*==> F1
gr_tree->add_key_stroke( cl_gui_simple_tree=>key_f1 )  .
*==> DELETE
gr_tree->add_key_stroke( cl_gui_simple_tree=>key_delete )  .

*** Set handler for registered events
SET HANDLER lcl_event_handler_class=>handle_keypress FOR gr_tree.

ENDFORM.                    „register_events

*&---------------------------------------------------------------------*
*&      Form  build_tree
*&---------------------------------------------------------------------*
* Build static tree structure:
* Group
*  +- ONE
*  +- TWO
* THREE
* FOUR
*----------------------------------------------------------------------*
FORM build_tree.

*** Data
DATA ls_node  TYPE mtreesnode.
DATA lt_nodes TYPE STANDARD TABLE OF mtreesnode.

*** Node „Group“
CLEAR ls_node.
ls_node-node_key  = ‚GROUP1‘.
ls_node-relatship = cl_gui_simple_tree=>relat_last_child.
ls_node-disabled  = ‚X‘.
ls_node-isfolder  = ‚X‘.
ls_node-n_image   = icon_activity_group.
ls_node-exp_image = icon_activity_group.
ls_node-style     = cl_gui_simple_tree=>style_intensified.
ls_node-text      = ‚Gruppe‘.
APPEND ls_node TO lt_nodes.
```

```
*** Node „ONE“
CLEAR ls_node.
ls_node-node_key  = ‚ONE‘.
ls_node-relatship = cl_gui_simple_tree=>relat_last_child.
ls_node-relatkey  = ‚GROUP1‘.
ls_node-n_image   = icon_led_yellow.
ls_node-style     = cl_gui_simple_tree=>style_intensified.
ls_node-text      = ‚Eins‘.
APPEND ls_node TO lt_nodes.

*** Node „TWO“
CLEAR ls_node.
ls_node-node_key  = ‚TWO‘.
ls_node-relatship = cl_gui_simple_tree=>relat_last_child.
ls_node-relatkey  = ‚GROUP1‘.
ls_node-n_image   = icon_led_green.
ls_node-style     = cl_gui_simple_tree=>style_default.
ls_node-text      = ‚ZWEI‘.
APPEND ls_node TO lt_nodes.

*** Node „THREE“
CLEAR ls_node.
ls_node-node_key  = ‚THREE‘.
ls_node-relatship = cl_gui_simple_tree=>relat_last_child.
ls_node-disabled  = ‚X‘.
ls_node-n_image   = icon_led_inactive.
ls_node-style     = cl_gui_simple_tree=>style_inactive.
ls_node-text      = ‚DREI‘.
APPEND ls_node TO lt_nodes.

*** Node „FOUR“
CLEAR ls_node.
ls_node-node_key  = ‚FOUR‘.
ls_node-relatship = cl_gui_simple_tree=>relat_last_child.
ls_node-disabled  = space.
ls_node-n_image   = icon_led_red.
ls_node-style     = cl_gui_simple_tree=>style_intensifd_critical.
ls_node-text      = ‚VIER‘.
APPEND ls_node TO lt_nodes.

*** Add nodes
CALL METHOD gr_tree->add_nodes
EXPORTING
table_structure_name = ‚MTREESNODE‘
node_table           = lt_nodes.

*** Expand all root nodes
gr_tree->expand_root_nodes( ).

ENDFORM.                    „init_controls
```