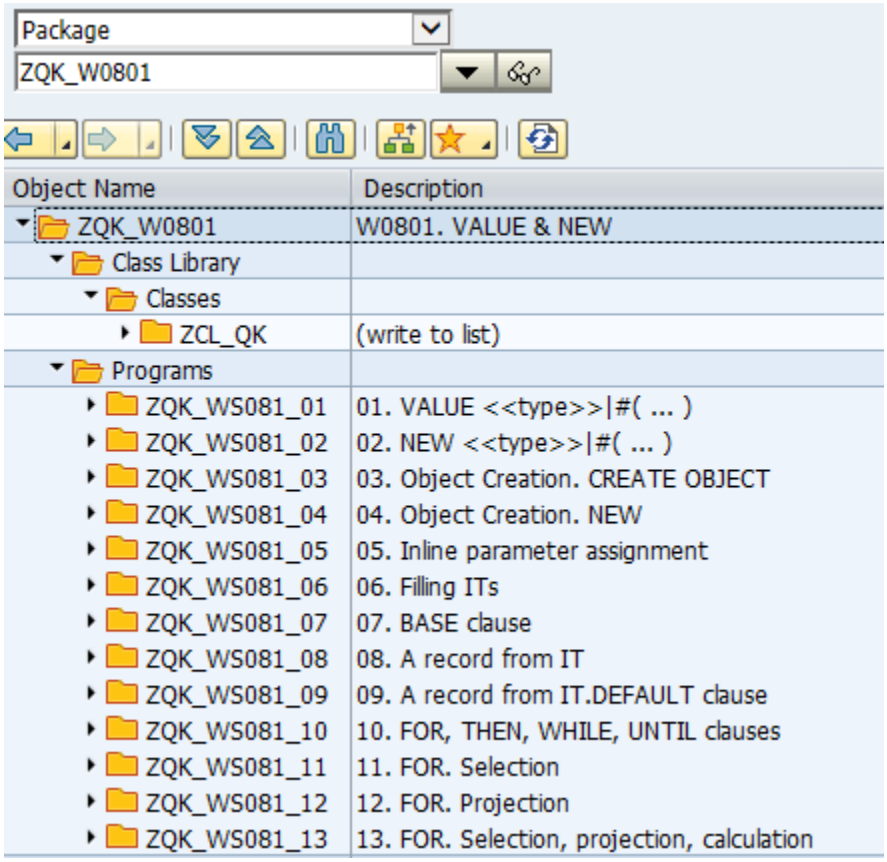
Вебинар Обучение ABAP.
Операторы конструкторы VALUE и NEW

Василий Ковальский, SAP CIS

12 октября 2021 12:00 – 13:00

Тексты программ программ



```abap
REPORT zqk_ws081_01."01. VALUE <<type>>|#( ... )
DATA
: gs_msg1 TYPE t100
, gs_msg2 TYPE t100
.
START-OF-SELECTION.
gs_msg1 = VALUE t100( sprsl = 'E' msgnr = 234 text = 'ABAP forever' ).
zcl_qk=>write_flat_structure( gs_msg1 ).

gs_msg2 = VALUE #( sprsl = 'E' arbgb = 'ZQK_WS081' text = 'I like ABAP' ).
zcl_qk=>write_flat_structure( gs_msg2 ).

SKIP.
ULINE.
WRITE: / 'BOT TAk!'.
```

```abap
REPORT zqk_ws081_02."02. NEW <<type>>|#( ... )
DATA
: gs_msg1 TYPE REF TO t100
, gs_msg2 TYPE REF TO t100
.
START-OF-SELECTION.
gs_msg1 = NEW t100( sprsl = 'E' msgnr = 234 text = 'ABAP forever' ).
zcl_qk=>write_flat_structure_by_ref( gs_msg1 ).

gs_msg2 = NEW #( sprsl = 'E' arbgb = 'ZQK_WS081' text = 'I like ABAP' ).
zcl_qk=>write_flat_structure_by_ref( gs_msg2 ).

SKIP.
ULINE.
WRITE: / 'BOT TAk!'.
```

```
REPORT zqk_ws081_03."03. Object Creation. CREATE OBJECT
DATA
: gt      TYPE STANDARD TABLE OF t100
, go_cont TYPE REF TO            cl_gui_docking_container
, go_grid TYPE REF TO            cl_gui_alv_grid
.
PARAMETERS dummy.

AT SELECTION-SCREEN OUTPUT.
  SELECT * FROM t100 UP TO 7 ROWS INTO TABLE gt WHERE sprsl = 'E'.
  CREATE OBJECT go_cont
    EXPORTING
      side  = cl_gui_docking_container=>dock_at_bottom
      ratio = 90.
  CREATE OBJECT go_grid
    EXPORTING
      i_parent = go_cont.
  go_grid->set_table_for_first_display(
    EXPORTING
      i_structure_name = 'T100'     " Internal Output Table Structure Name
    CHANGING
      it_outtab        =  gt        " Output Table
  ).

START-OF-SELECTION.
  WRITE: / 'BOT TAk!'.
```

```
REPORT zqk_ws081_04."04. Object Creation. NEW
DATA
: gt      TYPE STANDARD TABLE OF t100
, go_grid TYPE REF TO            cl_gui_alv_grid
.
PARAMETERS dummy.

AT SELECTION-SCREEN OUTPUT.
  SELECT * FROM t100 UP TO 7 ROWS INTO TABLE gt WHERE sprsl = 'E'.
  go_grid = NEW cl_gui_alv_grid(
      i_parent = NEW cl_gui_docking_container(
          side   = cl_gui_docking_container=>dock_at_bottom
          ratio  = 90
      )
  ).
  go_grid->set_table_for_first_display(
    EXPORTING
      i_structure_name = 'T100'     " Internal Output Table Structure Name
    CHANGING
      it_outtab        =  gt        " Output Table
  ).

START-OF-SELECTION.
  WRITE: / 'BOT TAk!'.
```

```
REPORT zqk_ws081_05." 05.Inline parameter assignment
DATA
: gt      TYPE STANDARD TABLE OF t100
, gs_layo TYPE lvc_s_layo
, go_grid TYPE REF TO            cl_gui_alv_grid
.
PARAMETERS dummy.

AT SELECTION-SCREEN OUTPUT.
  SELECT * FROM t100 UP TO 7 ROWS INTO TABLE gt WHERE sprsl = 'E'.
  IF go_grid IS NOT BOUND.
    go_grid = NEW cl_gui_alv_grid(
        i_parent = NEW cl_gui_docking_container(
            side   = cl_gui_docking_container=>dock_at_bottom
            ratio  = 90
        )
    ).
  ENDIF.
  go_grid->set_table_for_first_display(
    EXPORTING
      i_structure_name = 'T100'     " Internal Output Table Structure Name
      is_layout        = VALUE lvc_s_layo( zebra = 'X' )    " Layout
    CHANGING
      it_outtab        =  gt        " Output Table
  ).

*  DATA gs_layo TYPE lvc_s_layo.
*  gs_layo-zebra = 'X'.
*  go_grid->set_table_for_first_display(
*    EXPORTING
*      i_structure_name = 'T100'     " Internal Output Table Structure Name
*      is_layout        = gs_layo    " Layout
*    CHANGING
*      it_outtab        =  gt        " Output Table
*  ).

START-OF-SELECTION.
  WRITE: / 'BOT TAk!'.
```

```
REPORT zqk_ws081_06."06. Filling ITs
TYPES tt TYPE STANDARD TABLE OF t100 WITH EMPTY KEY.
DATA  gt TYPE tt.

START-OF-SELECTION.
  gt = VALUE tt(
        ( arbgb = 'Cat' text = 'Tiglatpalassar' )
        ( arbgb = 'Dog' text = 'Zlobermann'     )
        ( arbgb = 'Dog' text = 'Fido'           )
        ( arbgb = 'Cat' text = 'Basil'          )
     ).
  zcl_qk=>write_table( gt ).
  SKIP.
  ULINE.
  WRITE: / 'BOT TAk!'.
```

```
REPORT zqk_ws081_07."07. BASE clause
TYPES tt TYPE STANDARD TABLE OF t100 WITH EMPTY KEY.
DATA  gt TYPE tt.

START-OF-SELECTION.
  gt = VALUE tt( ( arbgb = 'Cat' text = 'Tiglatpalassar' ) ).
  zcl_qk=>write_table( gt ).
  gt = VALUE tt( ( arbgb = 'Dog' text = 'Zlobermann'     ) ).
  skip.
  zcl_qk=>write_table( gt ).
  gt = VALUE tt(  BASE gt ( arbgb = 'Dog' text = 'Fido'  ) ).
  skip.
  zcl_qk=>write_table( gt ).

  SKIP.
  ULINE.
  WRITE: / 'BOT TAk!'.
```

```
REPORT zqk_ws081_08."08. A record from IT
TYPES tt TYPE STANDARD TABLE OF t100 WITH EMPTY KEY.
DATA: gt TYPE tt.

START-OF-SELECTION.
  gt = VALUE tt(
        ( arbgb = 'Cat' text = 'Tiglatpalassar' )
        ( arbgb = 'Dog' text = 'Zlobermann'     )
        ( arbgb = 'Dog' text = 'Fido'           )
        ( arbgb = 'Cat' text = 'Basil'          )
     ).
  SKIP.
  zcl_qk=>write_table( gt ).
  DATA(gs) = VALUE #( gt[ text = 'Fido' ] ).
  SKIP.
  zcl_qk=>write_flat_structure( gs ).

  SKIP.
  ULINE.
  WRITE: / 'BOT TAk!'.
```

```
REPORT zqk_ws081_09."09. A record from IT.DEFAULT clause
TYPES tt TYPE STANDARD TABLE OF t100 WITH EMPTY KEY.
PARAMETERS p_error AS CHECKBOX.

START-OF-SELECTION.
  data(gt) = VALUE tt(
        ( arbgb = 'Cat' text = 'Tiglatpalassar' )
        ( arbgb = 'Dog' text = 'Zlobermann'     )
        ( arbgb = 'Dog' text = 'Fido'           )
        ( arbgb = 'Cat' text = 'Basil'          )
     ).
  SKIP.
  zcl_qk=>write_table( gt ).

  IF p_error = abap_true.
    data(gs) = VALUE #( gt[ text = 'Fid@' ] )." ! ERROR !
  ELSE.
    data(gs_default) = VALUE t100( arbgb = 'Horse' text = 'Sivka' ).
    gs = VALUE #( gt[ text = 'Fid@' ] DEFAULT  gs_default ).
*   gs = VALUE #( gt[ text = 'Fid@' ] DEFAULT  VALUE #( arbgb = 'Horse' text =
'Sivka' ) ).
  ENDIF.
  SKIP.
  zcl_qk=>write_flat_structure( gs ) .

  SKIP.
  ULINE.
  WRITE: / 'BOT TAk!'.
```

```
REPORT zqk_ws081_10."10. FOR, THEN, WHILE, UNTIL clauses
TYPES
: BEGIN OF ts
,   xx TYPE p DECIMALS 3
,   ff TYPE p DECIMALS 3
,   gg type string
, END OF ts
, tt type STANDARD TABLE OF ts WITH EMPTY KEY
.
START-OF-SELECTION.
  DATA(gt)  = VALUE tt(
      FOR   i = -10
      THEN  i + 1
      WHILE I <= 10
*      UNTIL i > 10
      ( xx = i
        ff = ( exp( i / 10 ) + exp( - i / 10 ) ) / 2
        gg = `|`
           && repeat( val = ` ` occ = ( exp( i / 10 ) + exp( - i / 10 ) ) * 20 -
1 )
           && `*`
      )
    ).

  WRITE: / 'Hyperbolic cosine: y = ch( x )'.
  zcl_qk=>write_table( gt ).

  SKIP.
  ULINE.
  WRITE: / 'BOT TAk!'.
```

```
REPORT zqk_ws081_11."11. FOR. Selection
TYPES tt TYPE STANDARD TABLE OF t100 WITH EMPTY KEY.
DATA: gt TYPE tt .

START-OF-SELECTION.
  gt = VALUE tt(
        ( arbgb = 'Cat' text = 'Tiglatpalassar' )
        ( arbgb = 'Dog' text = 'Zlobermann'     )
        ( arbgb = 'Cow' text = 'Booryonka'      )
        ( arbgb = 'Dog' text = 'Fido'           )
        ( arbgb = 'Cat' text = 'Basil'          )
     ).
  zcl_qk=>write_table( gt ).

  data(g1) = VALUE tt(
          FOR   wa                " Work area definition
          IN    gt                " Source table
          WHERE ( arbgb = 'Cat' ) " Condition
          ( wa )                  " Result
       ).
  skip.
  write: / 'Selection by field values. Only Cats'.
  zcl_qk=>write_table( g1 ).

  data(g2) = VALUE tt(
          FOR   wa                " Work area definition
          IN    gt                " Source table
          from  2                 " From
          to    4                 " To
          ( wa )                  " Result
       ).
  skip.
  write: / 'Selection by index. From 2 to 4'.
  zcl_qk=>write_table( g2 ).

  SKIP.
  ULINE.
  WRITE: / 'BOT TAk!'.
```

```
REPORT zqk_ws081_12."12. FOR. Projection
TYPES: tt TYPE STANDARD TABLE OF t100 WITH EMPTY KEY
     , BEGIN OF ts
     ,   species TYPE t100-arbgb
     ,   nick    TYPE t100-text
     , END OF ts
     , td TYPE STANDARD TABLE OF ts WITH EMPTY KEY
    .
START-OF-SELECTION.
  DATA(gt) = VALUE tt(
        ( arbgb = 'Cat' text = 'Tiglatpalassar' )
        ( arbgb = 'Dog' text = 'Zlobermann'     )
        ( arbgb = 'Cow' text = 'Booryonka'      )
        ( arbgb = 'Dog' text = 'Fido'           )
        ( arbgb = 'Cat' text = 'Basil'          )
    ).
  zcl_qk=>write_table( gt ).

  DATA(g1) = VALUE td(
         FOR   wa                " Work area definition
         IN    gt                " Source table
         ( species = wa-arbgb
           nick    = wa-text
         )                       " Result
      ).
  SKIP.
  WRITE: / 'Projection. Only the fields you need'.
  zcl_qk=>write_table( g1 ).

  SKIP.
  ULINE.
  WRITE: / 'BOT TAk!'.
```

```
REPORT zqk_ws081_13."13. FOR. Selection, projection, calculation
TYPES: tt TYPE STANDARD TABLE OF t100 WITH EMPTY KEY
     , BEGIN OF ts
     ,   no      TYPE p DECIMALS 4
     ,   species TYPE t100-arbgb
     ,   nick    TYPE t100-text
     , END OF ts
     , td TYPE STANDARD TABLE OF ts WITH EMPTY KEY
.

START-OF-SELECTION.
  DATA(gt) = VALUE tt(
        ( arbgb = 'Cat' text = 'Tiglatpalassar' )
        ( arbgb = 'Dog' text = 'Zlobermann'     )
        ( arbgb = 'Cow' text = 'Booryonka'      )
        ( arbgb = 'Dog' text = 'Fido'           )
        ( arbgb = 'Cat' text = 'Basil'          )
    ).
  zcl_qk=>write_table( gt ).

  DATA(g1) = VALUE td(
          FOR   wa                " Work area definition
          IN    gt                " Sorce table
          INDEX INTO ind          " Index value definition
          WHERE ( arbgb CP 'C*' )" Condition
          ( no      = sin( ind ) * 11
            species = wa-arbgb
            nick    = wa-text && ` is a pet of mine`
          )                       " Result
      ).
  SKIP.
  WRITE: / 'Selection, projection, calculation'.
  zcl_qk=>write_table( g1 ).

  SKIP.
  ULINE.
  WRITE: / 'BOT TAk!'.
```

```
class ZCL_QK definition
  public
  final
  create public .

public section.

  class-methods WRITE_FLAT_STRUCTURE
    importing
      value(IS) type DATA .
  class-methods WRITE_FLAT_STRUCTURE_BY_REF
    importing
      value(IR) type ref to DATA .
  class-methods WRITE_TABLE
    importing
      !IT type ANY TABLE .
protected section.
private section.
ENDCLASS.



CLASS ZCL_QK IMPLEMENTATION.


* <SIGNATURE>---------------------------------------------------------------------------------------
+
* | Static Public Method ZCL_QK=>WRITE_FLAT_STRUCTURE
* +--------------------------------------------------------------------------------------------------
+
* | [--->] IS                             TYPE        DATA
* +--------------------------------------------------------------------------------------
</SIGNATURE>
  method WRITE_FLAT_STRUCTURE.
    FIELD-SYMBOLS <fld> type data.
    DATA
    : lo_elm  TYPE REF TO cl_abap_elemdescr
    , lv_dd   TYPE        dfies
    .
    SKIP.
    DO.
      ASSIGN COMPONENT sy-index OF STRUCTURE is TO <fld>.
      IF sy-subrc NE 0.
        EXIT.
      ENDIF.
      lo_elm = CAST cl_abap_elemdescr(
               cl_abap_elemdescr=>describe_by_data( <fld> )
               ).
      lv_dd  = lo_elm->get_ddic_field( ).
      WRITE: / lv_dd-scrtext_m, <fld> COLOR COL_POSITIVE.
    ENDDO.
  endmethod.


* <SIGNATURE>---------------------------------------------------------------------------------------
+
* | Static Public Method ZCL_QK=>WRITE_FLAT_STRUCTURE_BY_REF
* +--------------------------------------------------------------------------------------------------
+
* | [--->] IR                             TYPE REF TO DATA
* +--------------------------------------------------------------------------------------
</SIGNATURE>
  METHOD write_flat_structure_by_ref.
    FIELD-SYMBOLS <str> TYPE data.
    ASSIGN ir->* TO <str>.
    write_flat_structure( <str> ).
  ENDMETHOD.
```

```
* <SIGNATURE>---------------------------------------------------------------------------------------
+
* | Static Public Method ZCL_QK=>WRITE_TABLE
* +---------------------------------------------------------------------------------------------------
+
* | [--->] IT                             TYPE        ANY TABLE
* +-----------------------------------------------------------------------------------
</SIGNATURE>
  METHOD write_table.
    FIELD-SYMBOLS
    : <fld> TYPE any
    , <rec> TYPE any
    .
    LOOP AT it ASSIGNING <rec>.
      NEW-LINE.
      DO.
        ASSIGN COMPONENT sy-index OF STRUCTURE <rec> TO <fld>.
        IF sy-subrc NE 0.
          EXIT.
        ENDIF.
        WRITE: <fld> COLOR COL_POSITIVE.
      ENDDO.
    ENDLOOP.
  ENDMETHOD.
ENDCLASS.
```