How to…

Validate manual input in WEB planning application

BUSINESS INFORMATION WAREHOUSE

Applicable Releases: BW 3.5

SAP (SAP America, Inc. and SAP AG) assumes no responsibility for errors or omissions in these materials.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

mySAP BI “How-To” papers are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using mySAP BI. Should you wish to receive additional information, clarification or support, please refer to SAP Professional Services (Consulting/Remote Consulting).

1 Business Scenario

You are using the WIB to create web based planning applications. A very important part of this planning is the manual capturing of data using planning layouts. In order to achieve a high quality of captured data you like to apply certain validation rules on the key figure values entered.

The current workaround is to create planning functions that validate the key figure values. This approach has some disadvantages, e.g. the overhead costs for this execution. The solution proposed in this How to paper will perform the check with the layout component itself using ABAP coding.

This solution doesn't require any modification of SAP source code.

2 Introduction

Since BW 3.5 Web based planning applications can be enhance by using a so-called Exit Class. This new feature offers many ways of changing the standard behavior. This How to paper gives you an example how the standard way of processing layout data can be changed/enhanced.

The sample described will invent a validation rule to check the key figure values of a given layout. The sample expects key figures with a number format like 1,000.00 or 1.000,00 (no date or time).

If you take the sample as it is, the validation rules are active for one layout (name is specified in the constant c_comp_name).

The rule itself is simple: it checks if the key figure value is greater than 0 and less than 500 (use constants c_int_low and c_int_high to change the limits).

Some technical background information:

- When creating a Web Interface using the Web Interface Builder (transaction bps_wb) the system generates a BSP page based on the customizing given. The BSP event handlers are implemented in the class cl_upwb_bsp_appl.
- By creating subclasses and using the redefinition capability of ABAP OO, you can change the standard behavior and add new functionalities.
- During the input processing the system created a list of tasks to be done (which components to use and which actions to perform) and dispatches the work to the different component. In order to add the validation feature, we redefine the dispatcher method.

Possible enhancements:

From a maintenance perspective it would be easier to specify the layouts to be validated in the customizing of the WEB Interface Builder itself. We can do this by adding a hidden input field containing a list of layout names (in the example given separated by '-'). This list is read and evaluated by the enhancement to determine if the layout should be validated or not (following this idea you could e.g. also specify the validation interval).

Sample Html-coding (add by using the text component of the WIB)

< input type=hidden name='validatelayout' value='MyLayout1-MyLayout2'>

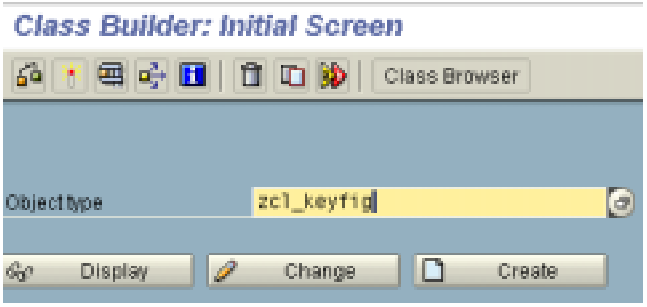
Sample ABAP-Coding for enhancement

Please see section 3.2. more details.

3 The Step By Step Solution

1. Create a subclass to cl_upwb_bsp_appl

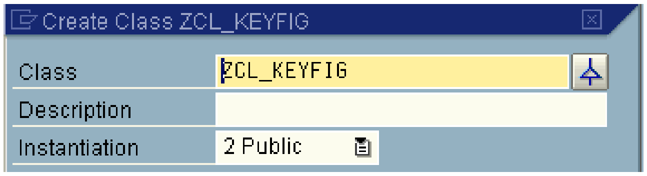
1. Call transaction SE24, enter the new class name and press create.



Class Builder: Initial Screen

Object type: zcl_keyfig

2. Press the Button on the pop-up to get the additional input field for the superclass.



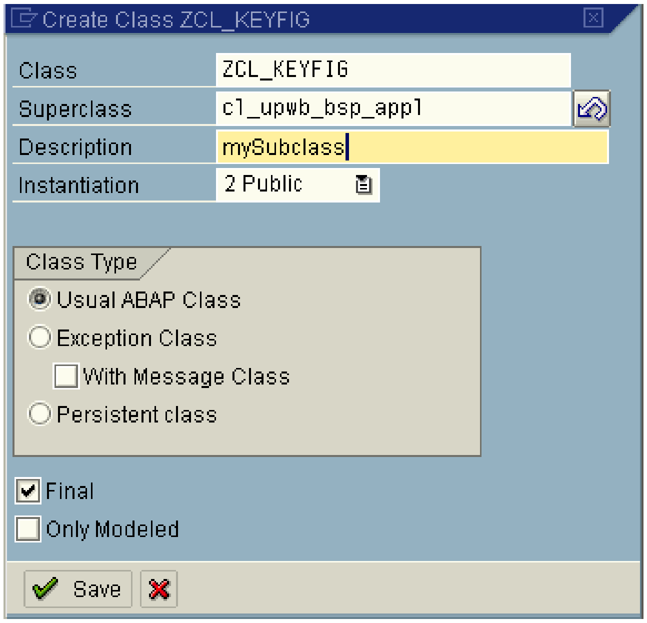
Create Class ZCL_KEYFIG

Class: ZCL_KEYFIG

Description:

Instantiation: 2 Public

3. Complete the pop-up as shown in the screenshot.



Create Class ZCL_KEYFIG

Class: ZCL_KEYFIG

Superclass: cl_upwb_bsp_appl

Description: mySubclass

Instantiation: 2 Public

Class Type

● Usual ABAP Class

○ Exception Class

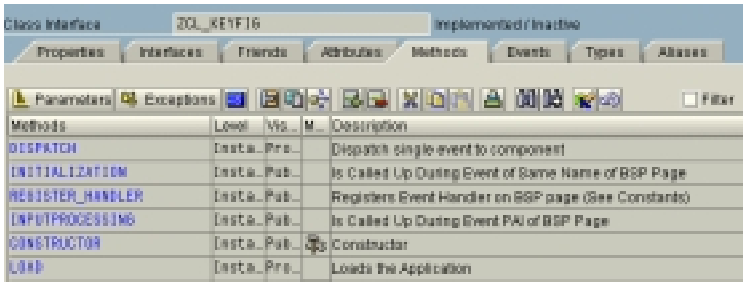
□ With Message Class

○ Persistent class

☑ Final

□ Only Modeled

4. Now you should see this screen. Select the dispatch method and press the button to redefine the method.



Class Interface ZCL_KEYFIG Implemented / Inactive

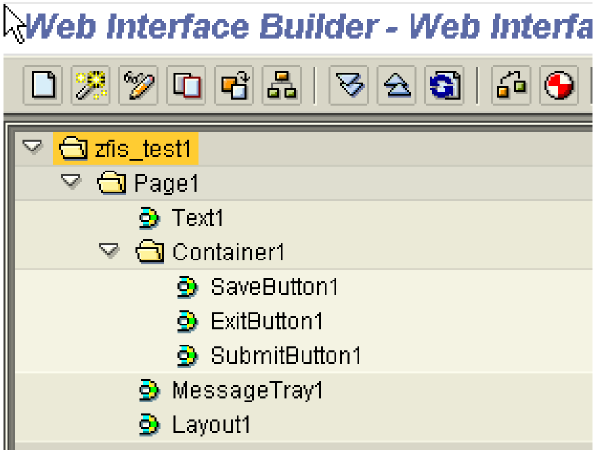
Methods	Level	Vis...	M...	Description
DISPATCH	Insta...	Pro...		Dispatch single event to component
INITIALIZATION	Insta...	Pub...		Is Called Up During Event of Same Name of BSP Page
REGISTER_HANDLER	Insta...	Pub...		Registers Event Handler on BSP page (See Constants)
INPUTPROCESSING	Insta...	Pub...		Is Called Up During Event PAI of BSP Page
CONSTRUCTOR	Insta...	Pub...		Constructor
LOAD	Insta...	Pro...		Loads the Application

5. Now an editor opens showing the following default ABAP code. Delete the coding and insert the ABAP coding given in the appendix of this how to.

```
method DISPATCH.
*CALL METHOD SUPER->DISPATCH
*   EXPORTING
*     IS_HANDLER =
*   RECEIVING
*     R_SUBRC    =
*     .
endmethod.
```

6. After activating the class (press button) we have now created a subclass of the standard implementation and we have the environment for validation in place. Congratulations!

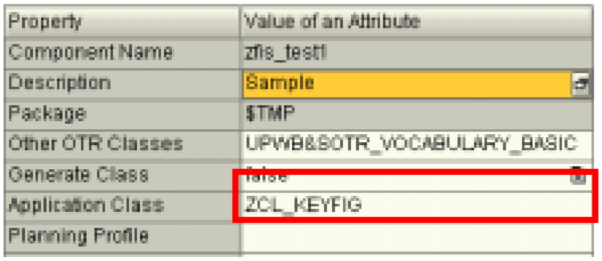
7. Now we have to assign this subclass to a Web Interface. Start transaction BPS_WB and open your planning application by pressing . The sample application we use here is zfis_test1.



Web Interface Builder - Web Interfa

```
zfis_test1
  Page1
    Text1
    Container1
      SaveButton1
      ExitButton1
      SubmitButton1
    MessageTray1
    Layout1
```

8. Double click on the root node (high lighted above) to open the properties list. Add the name of the subclass in the field for application class.



Property	Value of an Attribute
Component Name	zfis_test1
Description	Sample
Package	$TMP
Other OTR Classes	UPWB&SOTR_VOCABULARY_BASIC
Generate Class	false
Application Class	ZCL_KEYFIG
Planning Profile	

9. Generate the planning application by pressing .

2. Adjust the ABAP Coding

Now we have to change some constants to get the example live. Hence you should have the following information in place:

- Chose a layout from the Web application that we want to validate and note the component name of layout (e.g. Layout1)
- The sample validation rule will check if the key figure is part of a given interval [a, b]. Specify the lower and the upper limit.

1. Search for the constants c_int_low (lower limit) and c_int_high (upper limit) and replace the default values with your limits.

```
CONSTANTS:
    c_int_low  TYPE f VALUE '0',

    c_int_high TYPE f VALUE '500',
```

2. Decide if you like to specify the layout names in the backend (go to step 3a) or in the WIB itself (goto step 3b)

3a. Search for the constant `c_comp_name` and replace the value with your layout name. It can be found in line 37.

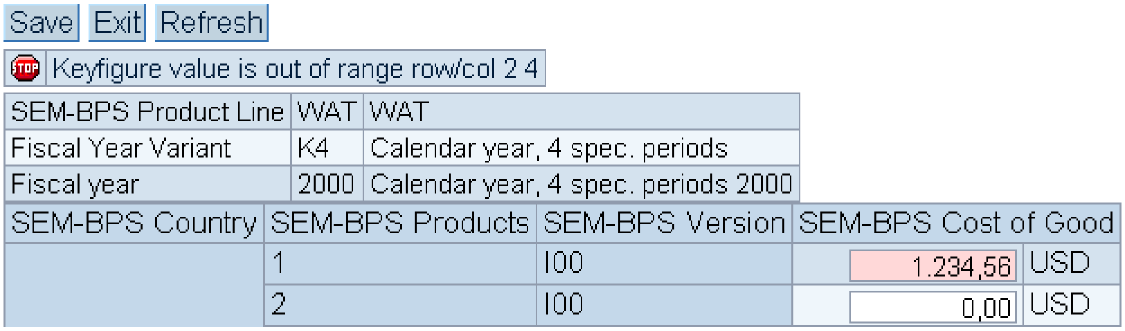
```
CONSTANTS:

    c_comp_name TYPE string VALUE 'myLayout'.
```

3b. Please comment the coding between *option1-start* and *option1-end* and uncomment the coding between *option2-start* and *option2-end*.

Add a text component with the text shown on the right and the property html set to true.

< input type=hidden name=’validatelayout’ value=’MyLayout1-MyLayout2’>

4. When executing the planning application the wrong entries will be highlighted in red and a message in the message log is shown. Congratulations!



Keyfigure value is out of range row/col 2 4

SEM-BPS Product Line	WAT	WAT
Fiscal Year Variant	K4	Calendar year, 4 spec. periods
Fiscal year	2000	Calendar year, 4 spec. periods 2000

SEM-BPS Country	SEM-BPS Products	SEM-BPS Version	SEM-BPS Cost of Good	
	1	100	1.234,56	USD
	2	100	0,00	USD

4 Appendix

```
METHOD dispatch.

  CALL METHOD super->dispatch
    EXPORTING
      is_handler = is_handler
    RECEIVING
      r_subrc    = r_subrc.

* ----------------------------------------------------------
* Example: validation of key figure values
* ----------------------------------------------------------
* 1. Check if dispatcher calls a layout - ready to validate
* ----------------------------------------------------------
  CONSTANTS:
    c_comp_class  TYPE string VALUE 'LAYOUT',
    c_comp_method TYPE string VALUE 'PROCESS_INPUT'.

* conditions to filter the approbiate component
  DATA:
    l_component TYPE REF TO if_upwb_c_component,
    l_class     TYPE string.

  l_component ?= is_handler-component.
  l_class      = l_component->get_class( ).

  IF NOT ( is_handler-method = c_comp_method
              AND l_class = c_comp_class ).
    EXIT.
  ENDIF.

* ------------------------------------------------------------
* 2. Add your own rule to determine if data of this spezific
* layout should be validated.
* In this example the validation will be done for Layout1.
* ------------------------------------------------------------

* ------------------------------------------------------------
* Option1-start
 CONSTANTS:
    c_comp_name TYPE string VALUE 'Layout1'.

  DATA:
    l_id TYPE string.

  l_id = l_component->get_id( ).
  IF NOT l_id = c_comp_name. EXIT. ENDIF.
* Option1-end
* ------------------------------------------------------------

** -----------------------------------------------------------
** Option2-start
** This is the html-coding expected in the WIB
** <input type="hidden" name="checklayout" value="mylayout1-mylayout2">
*
*  CONSTANTS:
*   c_field_name TYPE string VALUE 'checklayout'.
*
*  DATA:
*    l_id TYPE string,
*    lt_layout_name TYPE TABLE OF string,
*    l_layout_names TYPE string.
*
*  l_id = l_component->get_id( ).
*  l_layout_names = mr_request->get_form_field( c_field_name ).
*
*  SPLIT l_layout_names AT '-' INTO TABLE lt_layout_name.
*
*  READ TABLE lt_layout_name FROM l_id TRANSPORTING NO FIELDS.
*   IF sy-subrc <> 0. EXIT. ENDIF.
*
** Option2-end
** -----------------------------------------------------------

* ------------------------------------------------------------
* 3. Specify the set of cells that should be validated and
* apply the validation rule.
* In this example we use all cells in the data area and check
* if the reside within an interval [c_int_low, c_int_high].
* ------------------------------------------------------------

  CONSTANTS:
    c_int_low  TYPE f VALUE '0',
    c_int_high TYPE f VALUE '500',
    c_example  TYPE p DECIMALS 1 VALUE '1.5'.

  FIELD-SYMBOLS:
    <ls_cell> TYPE upwb_ys_html_cell.

  DATA:
   l_layout    TYPE REF TO if_upwb_c_layout2,
   ls_msg      TYPE bapiret2,
   lt_msg      TYPE TABLE OF bapiret2,
   l_example_in_c(4) TYPE c,
   l_decimal_sep     TYPE c,
   l_number          TYPE f,
   l_number_str      TYPE string.

* determine decimal seperator for external => internal conversion
  WRITE c_example TO l_example_in_c.
  l_decimal_sep = l_example_in_c+1(1).

  l_layout ?= l_component.

  CLEAR lt_msg.

* loop at cells which are ready for input (input_flag) and reside in
* the data area of the layout(ll-ratio=D,Y).
  LOOP AT l_layout->mt_html_cell ASSIGNING <ls_cell>
       WHERE input_flag = 'X'
         AND ( ll-ratio = 'D'
          OR   ll-ratio = 'Y' ).

    l_number_str = <ls_cell>-value.

* convert external to internal format
    IF l_decimal_sep = ','.
      TRANSLATE l_number_str USING ',..,'. "format is now 1,000.00
    ENDIF.
    TRANSLATE   l_number_str USING ', '.   "delete thousands
    CONDENSE    l_number_str NO-GAPS.

    CATCH SYSTEM-EXCEPTIONS convt_no_number    = 1
                            convt_overflow     = 2
                            bcd_field_overflow = 3
                            bcd_overflow       = 4.
      l_number = l_number_str.
    ENDCATCH.

    IF sy-subrc <> 0.
*     mark cell as incorrect
      <ls_cell>-alert_flag = 'X'.
      r_subrc = 10.
      CLEAR ls_msg.
      ls_msg-type   = 'E'.
      ls_msg-id     = 'UPF'.
      ls_msg-number = '001'.
      ls_msg-message_v1
      = 'Conversion of keyfigure not possible. Expected formats:'.
      ls_msg-message_v2 = '1.000,00 or 1,000.00. Row/Col:'.
      ls_msg-message_v3 = <ls_cell>-row.
      ls_msg-message_v4 = <ls_cell>-col.
      APPEND ls_msg TO lt_msg.
      CONTINUE.
    ENDIF.

* ----------------------------------------------------------------
* 3a. Add your own validation rule.
* ----------------------------------------------------------------
    IF l_number > c_int_high OR l_number <= c_int_low.

*     mark cell as incorrect
      <ls_cell>-alert_flag = 'X'.
      r_subrc = 10.

      CLEAR ls_msg.
      ls_msg-type   = 'E'.
      ls_msg-id     = 'UPF'.
      ls_msg-number = '001'.
      ls_msg-message_v1 = 'Keyfigure value is out of range'.
      ls_msg-message_v2 = 'row/col'.
      ls_msg-message_v3 = <ls_cell>-row.
      ls_msg-message_v4 = <ls_cell>-col.
      APPEND ls_msg TO lt_msg.
    ENDIF.
  ENDLOOP.

* add messages to error log
  IF NOT lt_msg IS INITIAL.
    l_layout->mo_msg_log->add( EXPORTING it_bapiret = lt_msg ).
  ENDIF.

ENDMETHOD.
```