Variant Maintenance for Selection Screen in Web Dynpro ABAP

Applies to:

SAP system with Web Dynpro functionality. For more information, visit the User Interface Technology homepage.

Summary

As we are aware that there is no standard functionality in Web Dynpro for maintaining selection screen variants as it is in ABAP. Here in this article, I have implemented custom logic to maintain selection screen variants for following simple application. This logic can be applied to all Web Dynpro applications that use selection screen.

Author: Anand Kolte

Company: Infosys

Created on: 18 April 2010

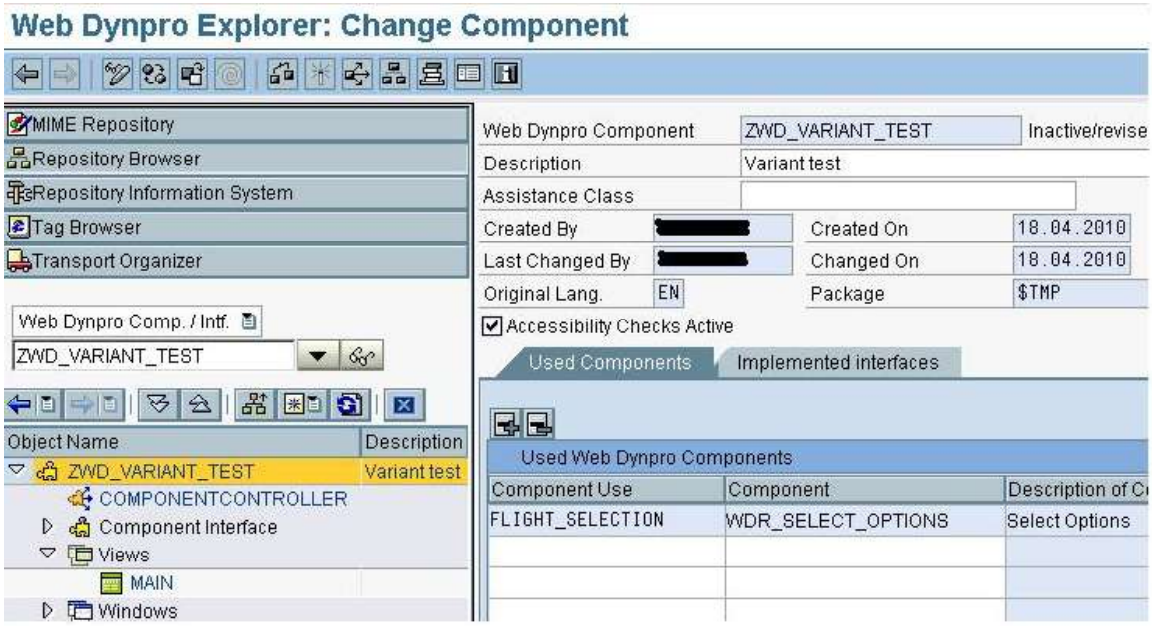
Author Bio

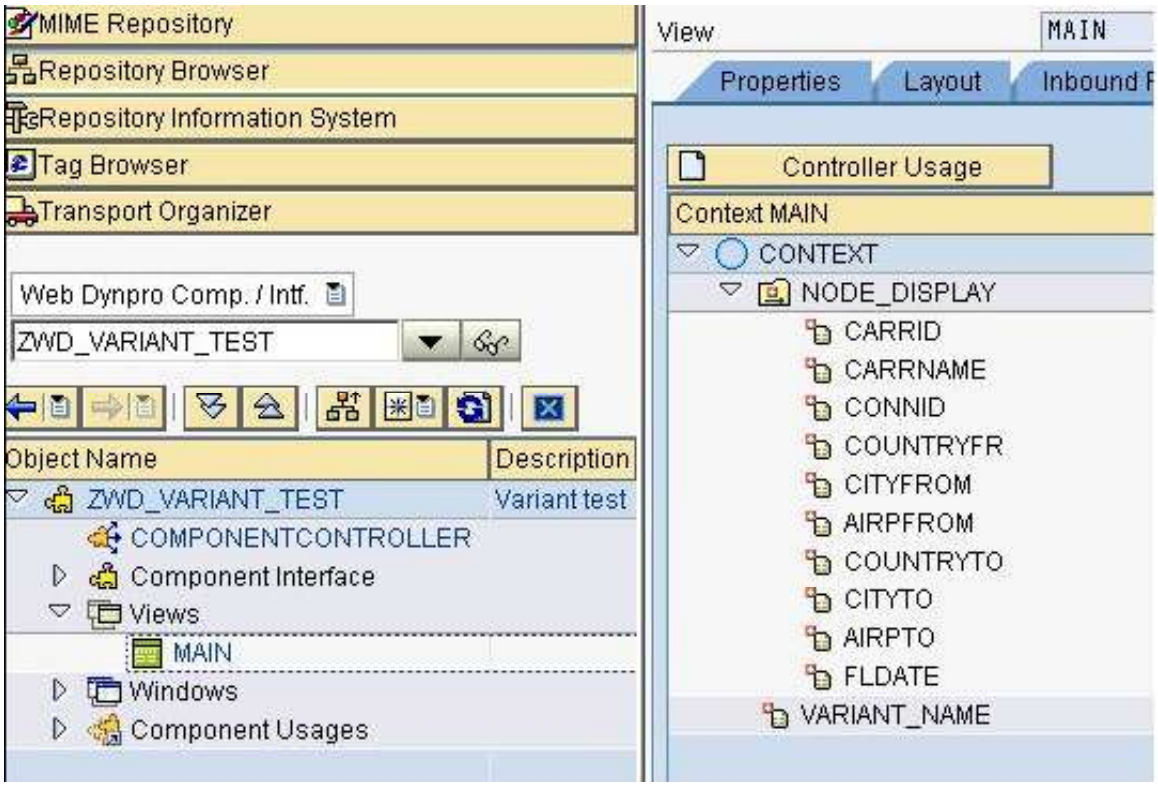
Hi, I am Anand Kolte and I am working with Infosys from more than 3 years in SAP ABAP technology. I got good exposure in SAP with variety of work in ABAP. I worked mainly on Reports, FMs, Interfaces, BAPIs, User exits, SAP Scripts, performance analysis etc. As a part of small development I also learned Web Dynpro and hence wanted to share my knowledge on maintenance of variants in Web Dynpro through this document.

Table of Contents

Step 1: Creating Web Dynpro Component

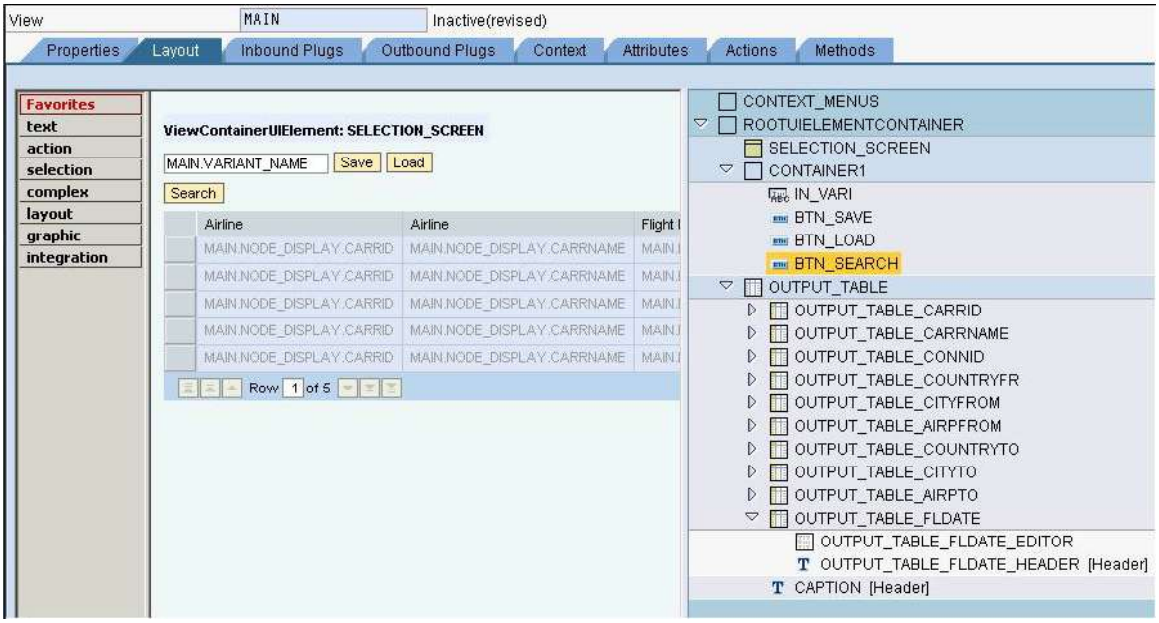
Start by creating a simple web dynpro application. Create an application say ZWD_VARIANT_TEST in this case and include standard component WDR_SELECT_OPTIONS for select options for searching flights. Also create a node 'NODE_DISPLAY' for holding result set.





Step 2: Creating Layout

Create a view as ‘MAIN’ and create layout in it as displayed below:



Let me explain you the layout details.

SELECTION_SCREEN is the ‘ViewContainerUIElement’ which will be holding our selection screen view.

IN_VARI is the input field for accepting name of variant. This field is bind to local attribute ‘VARIANT_NAME’ for this view.

BTN_SAVE is button we will be using for saving variant.

BTN_LOAD is button we will be using for loading saved variant.

Here I have created two actions for these buttons namely ACT_SAVE and ACT_LOAD.

BTN_SEARCH is button for displaying results in output table based on selections.

OUTPUT_TABLE is a table for displaying output results.

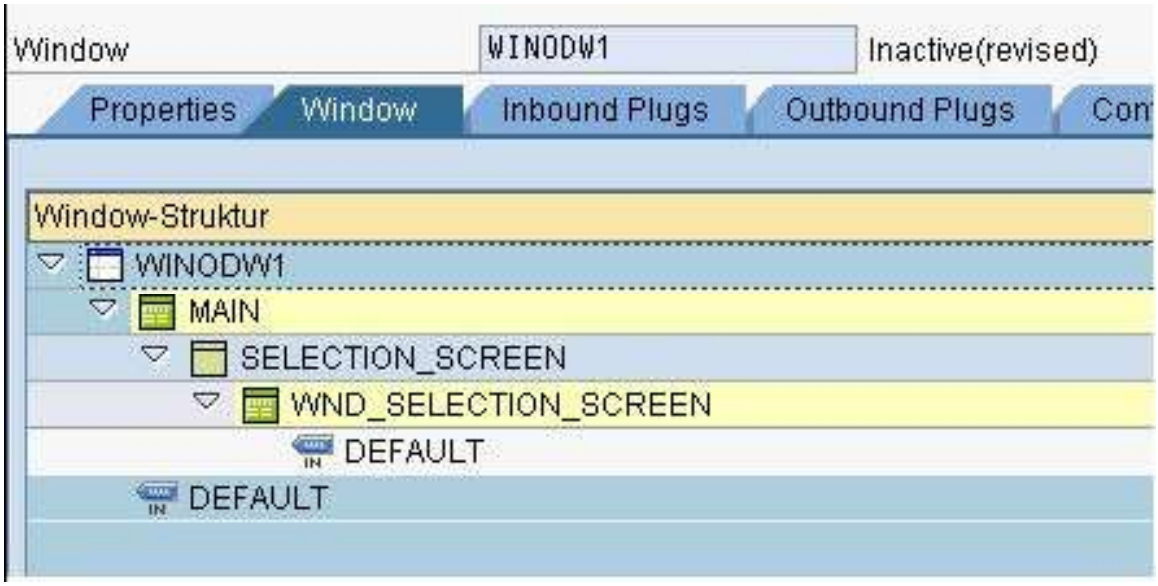
Step 3: Configuring Selection Screen

As usual, create controller usage for selection screen component in properties tab of view MAIN. Then add following attributes to the attribute tab of view MAIN.

M_HANDLER Type ref to IF_WD_SELECT_OPTIONS

M_WD_SEARCH Type ref to IWCI_WDR_SELECT_OPTIONS

Embed view MAIN to window WINDOW1 and again embed standard view of selection screen to ViewContainerUIElement SELECTION_SCREEN of MAIN view.



Step 4: Creating Selection Screen

Now, we will be building selection screen for our component. Let us consider only two fields for selection screen.

1. CARRID – Airline Code
2. CONNID – Flight Connection Number

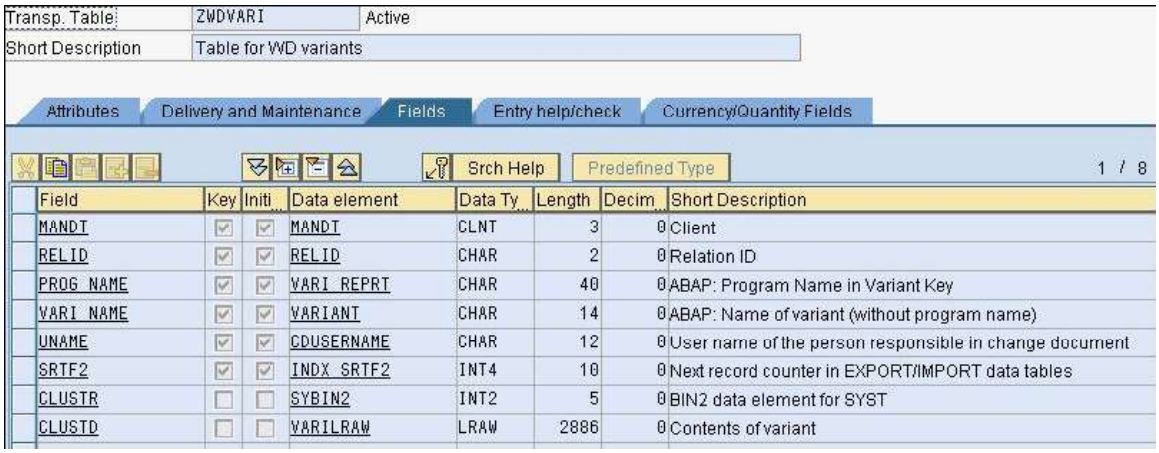
Now paste the below code in WDOINIT method of view MAIN.

```
--------------------------------------------------------------------------------
-
METHOD wddoinit .

  DATA lt_range_table       TYPE REF TO data.

  DATA lr_componentcontroller TYPE REF TO ig_componentcontroller.
  DATA l_cmp_usage TYPE REF TO if_wd_component_usage.

* create the used component
  l_cmp_usage = wd_this->wd_cpuse_flight_selection( ).
  IF l_cmp_usage->has_active_component( ) IS INITIAL.
    l_cmp_usage->create_component( ).
  ENDIF.

* get a pointer to the interface controller of the select options
*component
```

```
    wd_this->m_wd_search =
         wd_this->wd_cpifc_flight_selection( ).

  * init the select screen
    wd_this->m_handler =
         wd_this->m_wd_search->init_selection_screen( ).

  * create a range table for CARRID
    lt_range_table =
      wd_this->m_handler->create_range_table(
           i_typename = 'S_CARR_ID' ).

  * add a new field to the selection
    wd_this->m_handler->add_selection_field(
                  i_id = 'S_CARR_ID'
                  it_result = lt_range_table ).


  * create a range table for CONNID
    lt_range_table =
      wd_this->m_handler->create_range_table(
           i_typename = 'S_CONN_ID' ).

  * add a new field to the selection
    wd_this->m_handler->add_selection_field(
                  i_id = 'S_CONN_ID'
                  it_result = lt_range_table ).
  ENDMETHOD.
  ----------------------------------------------------------------------
```

Step 5: Creating Custom Table

Let us create a custom table ZWDVARI for storing variants.



Transp. Table: ZWDVARI — Active

Short Description: Table for WD variants

Attributes | Delivery and Maintenance | Fields | Entry help/check | Currency/Quantity Fields

Srch Help | Predefined Type | 1 / 8

Field	Key	Initi	Data element	Data Ty	Length	Decim	Short Description
MANDT	☑	☑	MANDT	CLNT	3	0	Client
RELID	☑	☑	RELID	CHAR	2	0	Relation ID
PROG_NAME	☑	☑	VARI_REPRT	CHAR	40	0	ABAP: Program Name in Variant Key
VARI_NAME	☑	☑	VARIANT	CHAR	14	0	ABAP: Name of variant (without program name)
UNAME	☑	☑	CDUSERNAME	CHAR	12	0	User name of the person responsible in change document
SRTF2	☑	☑	INDX_SRTF2	INT4	10	0	Next record counter in EXPORT/IMPORT data tables
CLUSTR	☐	☐	SYBIN2	INT2	5	0	BIN2 data element for SYST
CLUSTD	☐	☐	VARILRAW	LRAW	2886	0	Contents of variant

Step 6: Saving Variant

For storing the variant, first we need to read all the selection fields and then store all the details in table ZWDVARI.

Paste the following code in action handler method for action ACT_SAVE.

```
-------------------------------------------------------------------------------
METHOD onactionact_save .

  TYPES: BEGIN OF ty_key,
           prog_name      TYPE zwdvari-prog_name,
           vari_name      TYPE zwdvari-vari_name,
           uname          TYPE zwdvari-uname,
         END OF ty_key.

  DATA : lt_valueset       TYPE REF TO data,
         lt_selections     TYPE TABLE OF bapiselect,
         lwa_selections    TYPE bapiselect,
         lwa_key           TYPE ty_key.

  FIELD-SYMBOLS : <fs_tab> TYPE table,
                  <fs_wa>  TYPE ANY.


* Get range table of S_CARR_ID
  lt_valueset = wd_this->m_handler->get_range_table_of_sel_field( 'S_CARR_ID' ).
  ASSIGN lt_valueset->* TO <fs_tab>.

  LOOP AT <fs_tab> ASSIGNING <fs_wa>.
    MOVE-CORRESPONDING <fs_wa> TO lwa_selections.
    MOVE 'S_CARR_ID' TO lwa_selections-infoobject.
    APPEND lwa_selections TO lt_selections.
    CLEAR lwa_selections.
  ENDLOOP.

* Get range table of S_CONN_ID
  lt_valueset = wd_this->m_handler->get_range_table_of_sel_field( 'S_CONN_ID' ).
  ASSIGN lt_valueset->* TO <fs_tab>.

  LOOP AT <fs_tab> ASSIGNING <fs_wa>.
    MOVE-CORRESPONDING <fs_wa> TO lwa_selections.
    MOVE 'S_CONN_ID' TO lwa_selections-infoobject.
    APPEND lwa_selections TO lt_selections.
    CLEAR lwa_selections.
  ENDLOOP.

* Get the Variant name
  DATA lo_el_context TYPE REF TO if_wd_context_element.
  DATA lv_variant_name TYPE wd_this->element_context-variant_name.

* get element via lead selection
  lo_el_context = wd_context->get_element( ).

* get single attribute
  lo_el_context->get_attribute(
```

```
    EXPORTING
      name =  `VARIANT_NAME`
    IMPORTING
      value = lv_variant_name ).

  CHECK lv_variant_name IS NOT INITIAL.

* Save variant to Database
  lwa_key-prog_name = 'ZWD_VARIANT_TEST'.
  lwa_key-vari_name =  lv_variant_name.
  lwa_key-uname     =  sy-uname.


  EXPORT datatable = lt_selections
   TO DATABASE  zwdvari(wd)
    ID lwa_key.


ENDMETHOD.
--------------------------------------------------------------------------------
```

Step 7: Loading Variant

For loading variant, we need to import the table from database and then fill up the selection screen separately for every select-option.

Paste the following code in action handler method of action ACT_LOAD.

```
--------------------------------------------------------------------------------
METHOD onactionact_load .

  TYPES: BEGIN OF ty_key,
           prog_name      TYPE zwdvari-prog_name,
           vari_name      TYPE zwdvari-vari_name,
           uname          TYPE zwdvari-uname,
         END OF ty_key.

  DATA : lt_valueset       TYPE REF TO data,
         lt_selections     TYPE TABLE OF bapiselect,
         lwa_selections    TYPE bapiselect,
         lwa_key           TYPE ty_key,
         lwa_tab           TYPE REF TO data.

  FIELD-SYMBOLS : <fs_tab> TYPE table,
                  <fs_wa>  TYPE ANY.


* Read Variant that has to be loaded
  DATA lo_el_context TYPE REF TO if_wd_context_element.
  DATA lv_variant_name TYPE wd_this->element_context-variant_name.

* get element via lead selection
  lo_el_context = wd_context->get_element( ).

* get single attribute
  lo_el_context->get_attribute(
```

```
    EXPORTING
      name =  `VARIANT_NAME`
    IMPORTING
      value = lv_variant_name ).


  lwa_key-prog_name = 'ZWD_VARIANT_TEST'.
  lwa_key-vari_name = lv_variant_name.
  lwa_key-uname     = sy-uname.

  IMPORT datatable = lt_selections
      FROM DATABASE zwdvari(wd)
      ID lwa_key.


** Set values for S_CARR_ID
* Create range table
  lt_valueset = wd_this->m_handler->create_range_table( 'S_CARR_ID' ).

  ASSIGN lt_valueset->* TO <fs_tab>.
  CREATE DATA lwa_tab LIKE LINE OF <fs_tab>.
  ASSIGN lwa_tab->* TO <fs_wa>.

* Extract values for S_CARR_ID
  LOOP AT lt_selections INTO lwa_selections
      WHERE infoobject EQ 'S_CARR_ID'.
    MOVE-CORRESPONDING lwa_selections TO <fs_wa>.
    APPEND <fs_wa> TO <fs_tab>.
  ENDLOOP.

* Set extracted values to select-option
  CALL METHOD wd_this->m_handler->set_range_table_of_sel_field
    EXPORTING
      i_id           = 'S_CARR_ID'
      it_range_table = lt_valueset.


** Set values for S_CONN_ID
* Create range table
  lt_valueset = wd_this->m_handler->create_range_table( 'S_CONN_ID' ).

  ASSIGN lt_valueset->* TO <fs_tab>.
  CREATE DATA lwa_tab LIKE LINE OF <fs_tab>.
  ASSIGN lwa_tab->* TO <fs_wa>.

* Extract values for S_CONN_ID
  LOOP AT lt_selections INTO lwa_selections
      WHERE infoobject EQ 'S_CONN_ID'.
    MOVE-CORRESPONDING lwa_selections TO <fs_wa>.
    APPEND <fs_wa> TO <fs_tab>.
  ENDLOOP.

* Set extracted values to select-option
  CALL METHOD wd_this->m_handler->set_range_table_of_sel_field
    EXPORTING
      i_id           = 'S_CONN_ID'
```

```
      it_range_table = lt_valueset.

ENDMETHOD.
-----------------------------------------------------------------------------------
```

Step 8: Fetch Data

After entering values on selection screen, press the Search button. Paste the code given below in action handler method of action SEARCH. Here we are reading the select-options and querying database table depending on them.

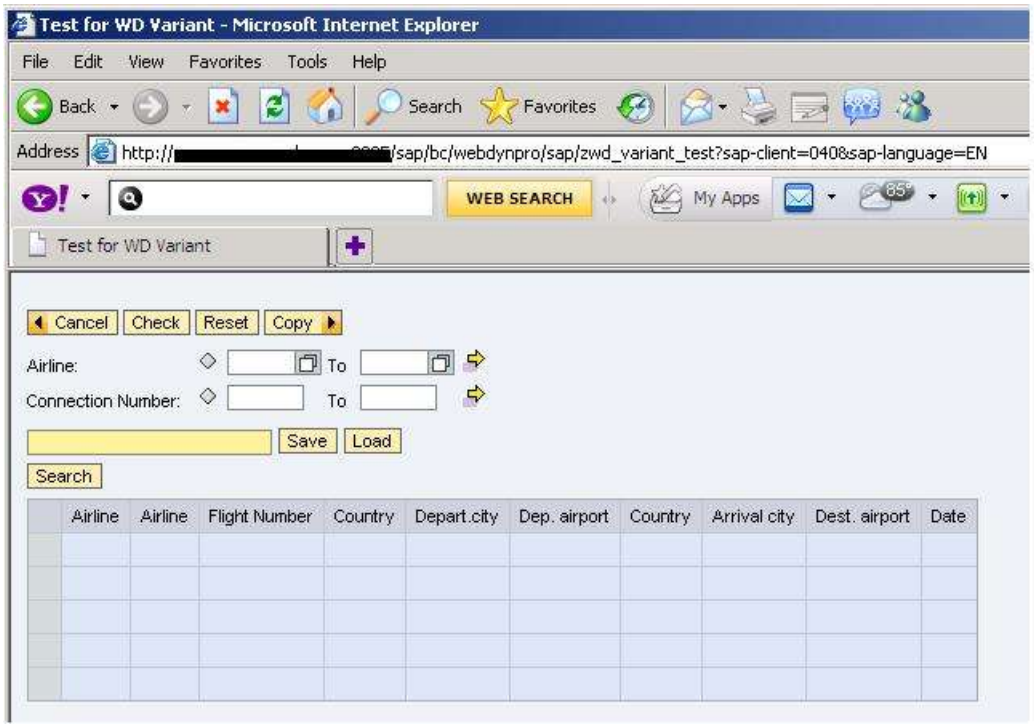
```
-----------------------------------------------------------------------------------
METHOD onactionact_search .

  DATA : lt_valueset       TYPE REF TO data.

  FIELD-SYMBOLS : <fs_tab1> TYPE table,
                  <fs_tab2> TYPE table.
  DATA lo_nd_node_display TYPE REF TO if_wd_context_node.
  DATA lt_node_display TYPE TABLE OF wd_this->element_node_display.

* Get range table of S_CARR_ID
  lt_valueset = wd_this->m_handler->get_range_table_of_sel_field( 'S_CARR_ID' ).
  ASSIGN lt_valueset->* TO <fs_tab1>.

* Get range table of S_CONN_ID
  lt_valueset = wd_this->m_handler->get_range_table_of_sel_field( 'S_CONN_ID' ).
  ASSIGN lt_valueset->* TO <fs_tab2>.

* Select data from database
  SELECT * FROM sflights
    INTO CORRESPONDING FIELDS OF TABLE lt_node_display
    UP TO 500 ROWS
    WHERE carrid IN <fs_tab1>
      AND connid IN <fs_tab2>.

* navigate from <CONTEXT> to <NODE_DISPLAY> via lead selection
  lo_nd_node_display = wd_context->get_child_node( name = wd_this->wdctx_node_display
).

* Bind table to node
  lo_nd_node_display->bind_table( lt_node_display ).
ENDMETHOD.
-----------------------------------------------------------------------------
```

Step 9: Demo!

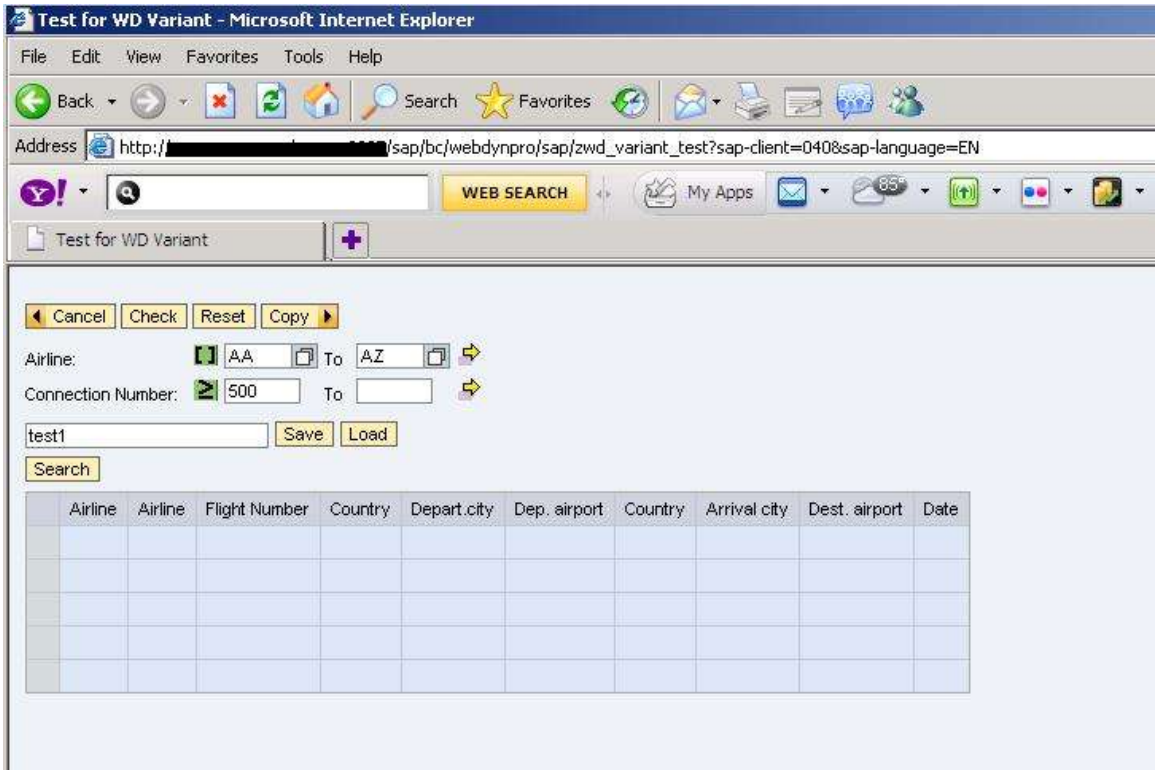
DEMO TIME !!

Create a web dynpro application for our component and save it. After executing the application, we will get following screen in Internet browser:

After executing the application we will get below screen:

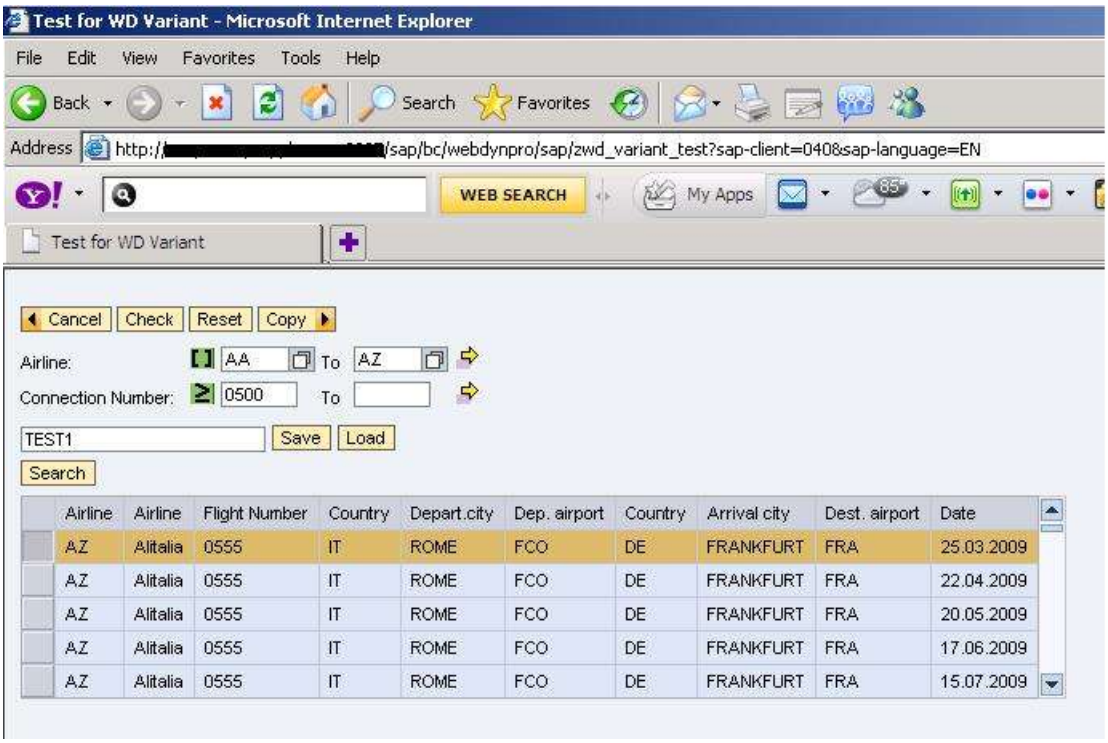


We will populate some values in both the select options, give some variant name (‘test1’ in this case) in the input field and hit SAVE button. This will save the user-specific variant in our table ZWDVARI.



Exit the application and run it once again. Now just give variant name (‘test1’ in this case) and hit LOAD button. This will fetch the already saved variant from table ZWDVARI, extract the values for each select option and then load those values back into selection fields.

On hitting the search button we will get below output:



Related Content

http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/c09fec07-0dab-2a10-dbbe-c9a26bdff03e

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.