ALV and Standard Table as Hierarchy in Web Dynpro ABAP

Applies to:

SAP ECC 6.0 (Release 700, SP 13). For more information, visit the User Interface Development with Web Dynpro for ABAP Page .

Summary

This Article deals with the Hierarchy (tree structure) using ALV and Standard Table in Web Dynpro ABAP

Author: Abhimanyu Lagishetti

Company: Satyam Computer Services Ltd.

Created on: 27 August 2008

Author Bio

Abhimanyu Lagishetti is working for Abhimanyu L, Satyam Computer Services Ltd. He is B.Tech Computer Science Graduate, and working on Technologies like Web Dynpro Java, Web Dynpro ABAP, Enterprise Portals and Business Workflows.

Table of Contents

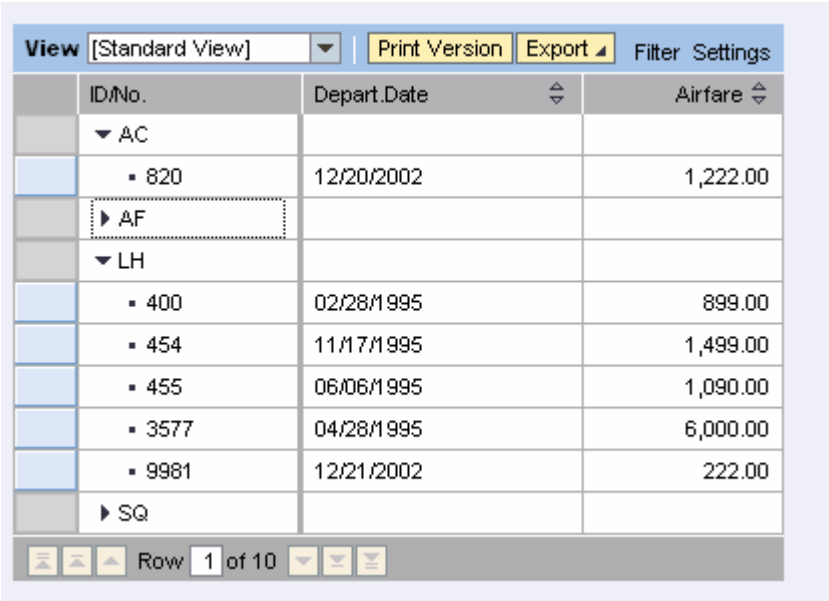
Introduction 3
Introduction 3
ALV as Hierarchy 3
Standard Table as Hierarchy 3
Get Started with an Application 4
Add the Interface controller to the used controllers/component of the component controller 6
Table as Hierarchy 6
Specifying the Display Type 6
Specifying the Hierarchy Column 7
Displaying Data of the Last Hierarchy Column as a Leaf 7
Standard Table as Hierarchy 7
Summary 11
Related Content 11
Disclaimer and Liability Notice 12

Introduction

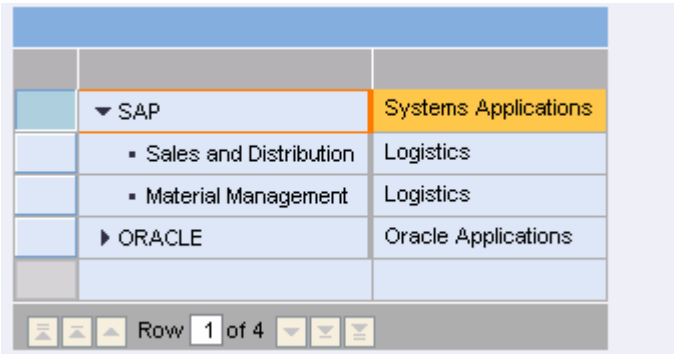
ALV table as hierarchy is different from standard table hierarchy. There is little information available in help document about hierarchy in ALV using Web Dynpro ABAP, this document demonstrates a typical scenario how hierarchy is achieved in ALV also some SAP modules as hierarchy using Standard Table.

The goal of the document is to show list flight details as a hierarchy in ALV and some SAP Modules as hierarchy in Standard Table.

ALV as Hierarchy

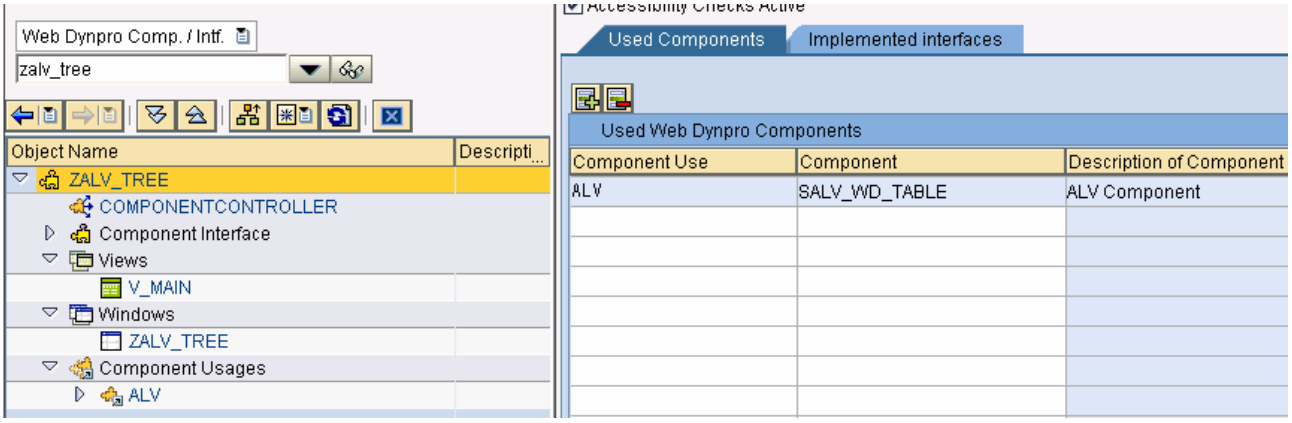


Standard Table as Hierarchy

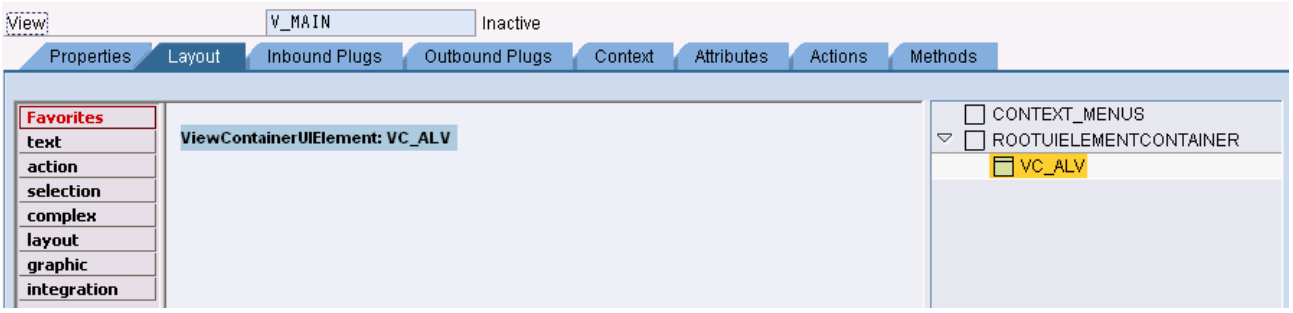


Get Started with an Application

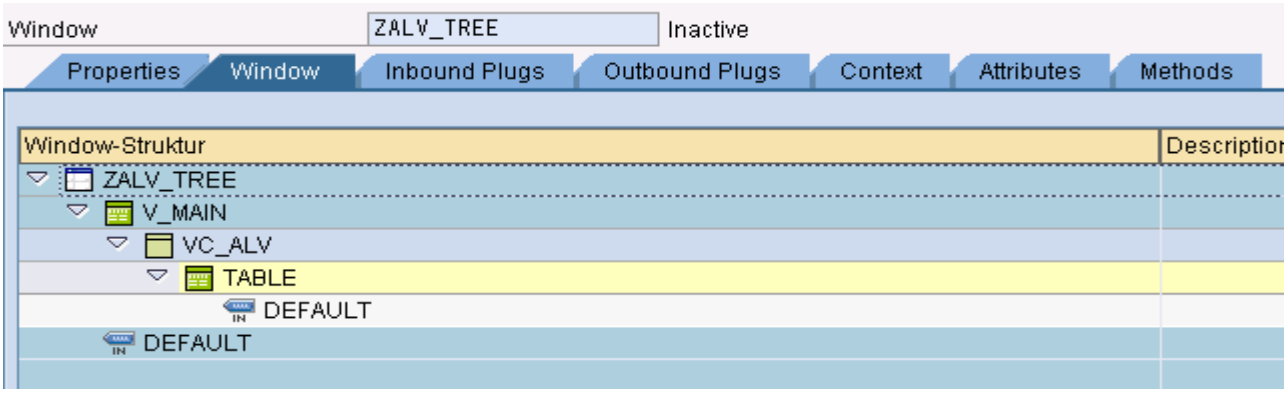
- Create a web Dynpro component ZALV_TREE
- Add Web Dynpro component SALV_WD_TABLE as component usage with name ALV



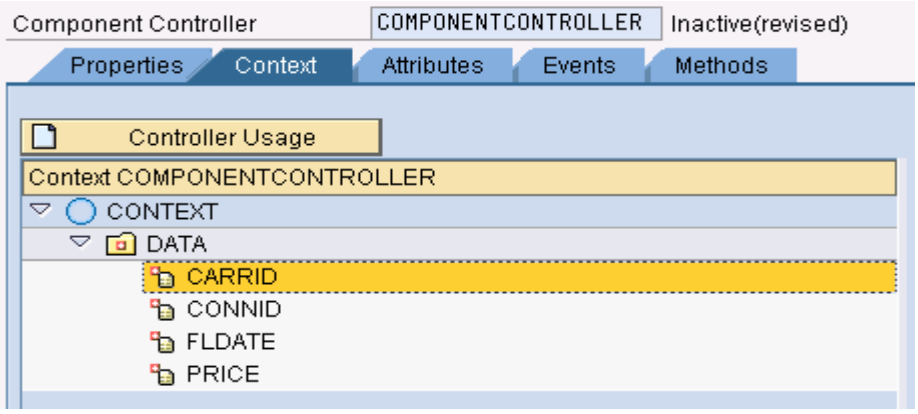
- Create a View V_MAIN to show the hierarchy tables.
- Place a view container ui element to display ALV component.



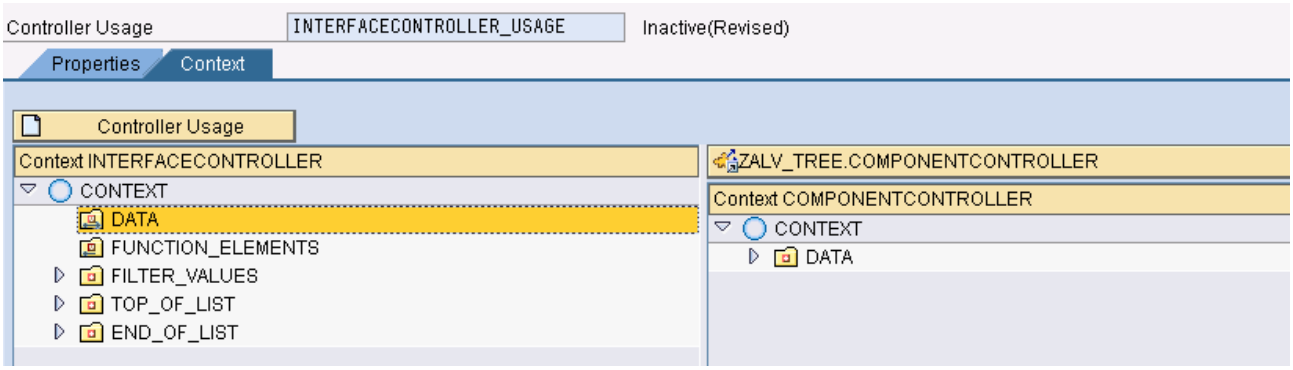
- Embed V_MAIN in the window.
- Embed TABLE interface view of the ALV component in the view container.



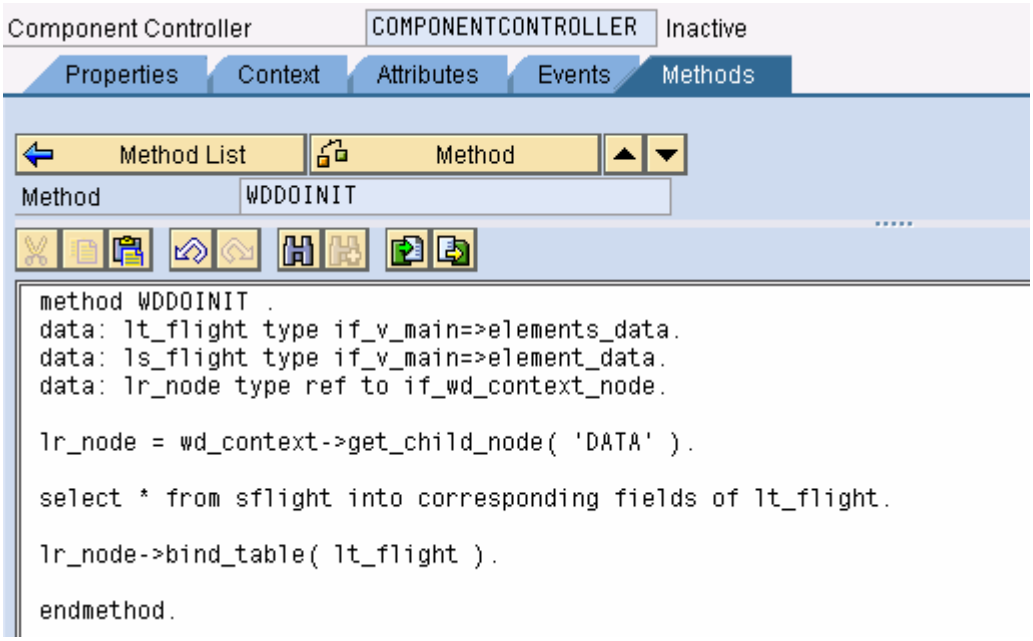
- Create Context in the component controller as shown



- Perform external mapping to the ALV Interface controller node with the node in created in component controller



- Fill the context node DATA with values in WDDOINIT of the component controller.



Add the Interface controller to the used controllers/component of the component controller

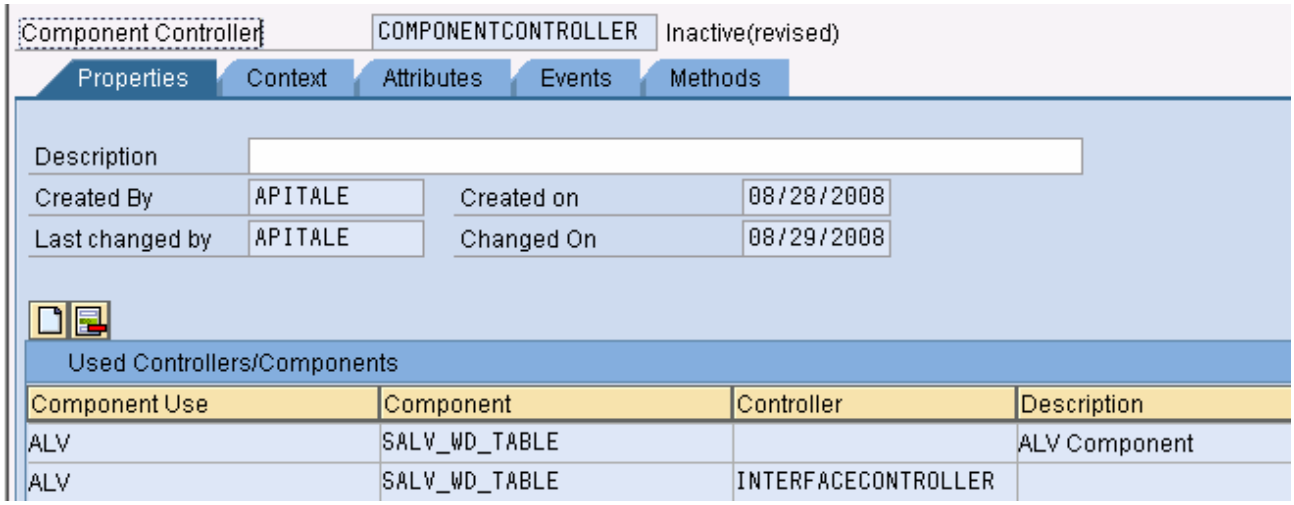


Table as Hierarchy

- Specify hierarchy column
- Specify display type

➢ Create the component and get the model of ALV component in WDDOINIT of component controller.

```
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

data: l_ref_cmp_usage type ref to if_wd_component_usage.
DATA: l_ref_INTERFACECONTROLLER TYPE REF TO IWCI_SALV_WD_TABLE .
data: l_VALUE type ref to Cl_Salv_Wd_Config_Table.

l_ref_cmp_usage =   wd_This->wd_CpUse_Alv( ).
if l_ref_cmp_usage->has_active_component( ) is initial.
  l_ref_cmp_usage->create_component( ).
endif.

l_ref_INTERFACECONTROLLER = wd_This->wd_CpIfc_Alv( ).
l_VALUE = l_ref_INTERFACECONTROLLER->Get_Model( ).

```

Specifying the Display Type

To define your ALV output as a hierarchy and thereby define the type of display, you use the methods of interface class IF_SALV_WD_TABLE_SETTINGS (implementing class CL_SALV_WD_CONFIG_TABLE).

```
  l_value->IF_SALV_WD_TABLE_SETTINGS~SET_DISPLAY_TYPE(
IF_SALV_WD_C_TABLE_SETTINGS=>DISPLAY_TYPE_HIERARCHY ).
```

Specifying the Hierarchy Column

To define a column as a hierarchy column, you use the methods of interface class IF_SALV_WD_COLUMN_HIERARCHY (implementing class CL_SALV_WD_COLUMN).

- In our case set CARRID and CONNID as hierarchy columns to get a hierarchy as desired

```
data: lr_column type ref to CL_SALV_WD_COLUMN.

  lr_column =
  l_value->IF_SALV_WD_COLUMN_SETTINGS~GET_COLUMN( 'CARRID' ).
  LR_column->if_salv_wd_column_hierarchy~set_hierarchy_column( abap_true
 ).

   lr_column =
  l_value->IF_SALV_WD_COLUMN_SETTINGS~GET_COLUMN( 'CONNID' ).
  LR_column->if_salv_wd_column_hierarchy~set_hierarchy_column( abap_true
 ).
```

Displaying Data of the Last Hierarchy Column as a Leaf

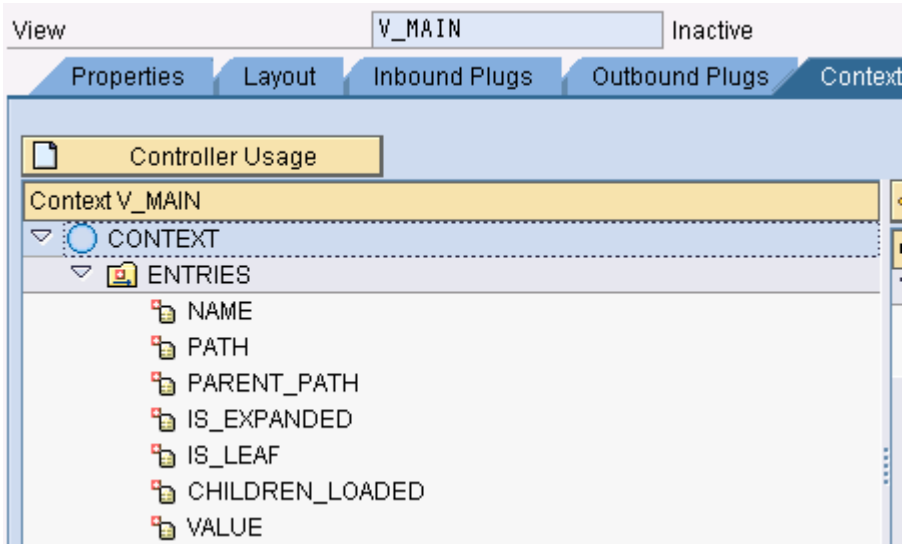
To display the data of the last hierarchy column in the first column of the hierarchy as a leaf, you use the methods of interface class IF_SALV_WD_TABLE_HIERARCHY (implementing class CL_SALV_WD_CONFIG_TABLE).

```
L_VALUE->IF_SALV_WD_TABLE_HIERARCHY~SET_LAST_HIER_COLUMN_AS_LEAF( abap_true ).
```

- Create an Application and run

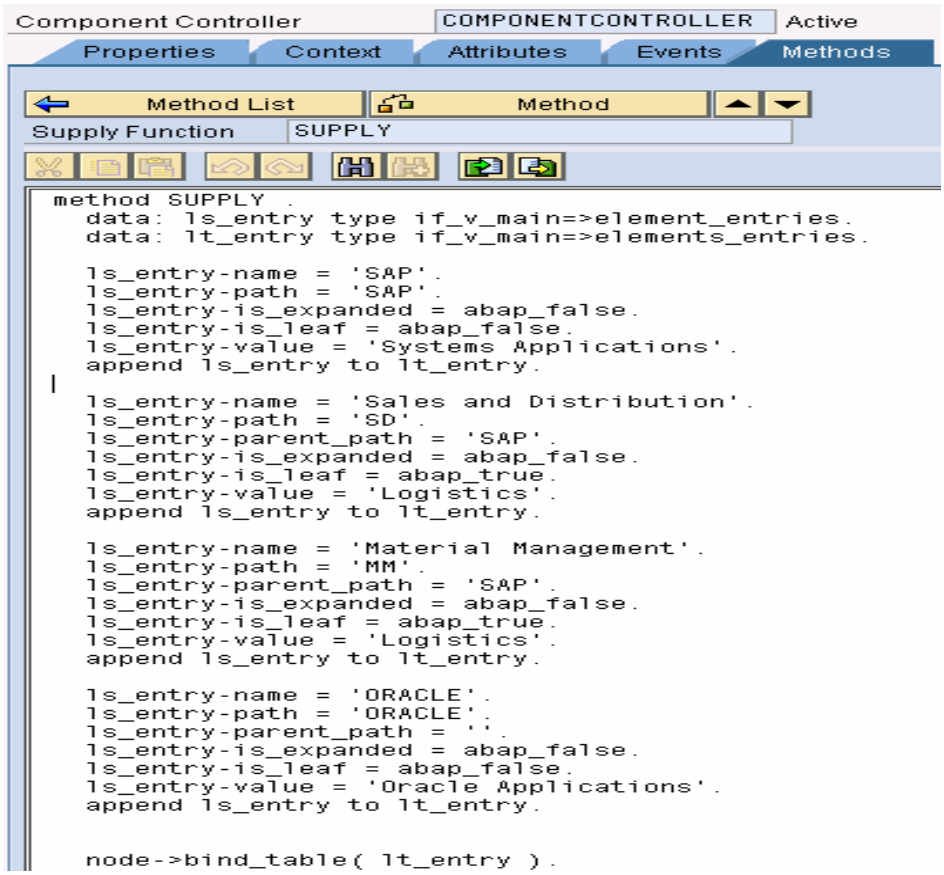
Standard Table as Hierarchy

- Create Context in the View As follows with a supply function



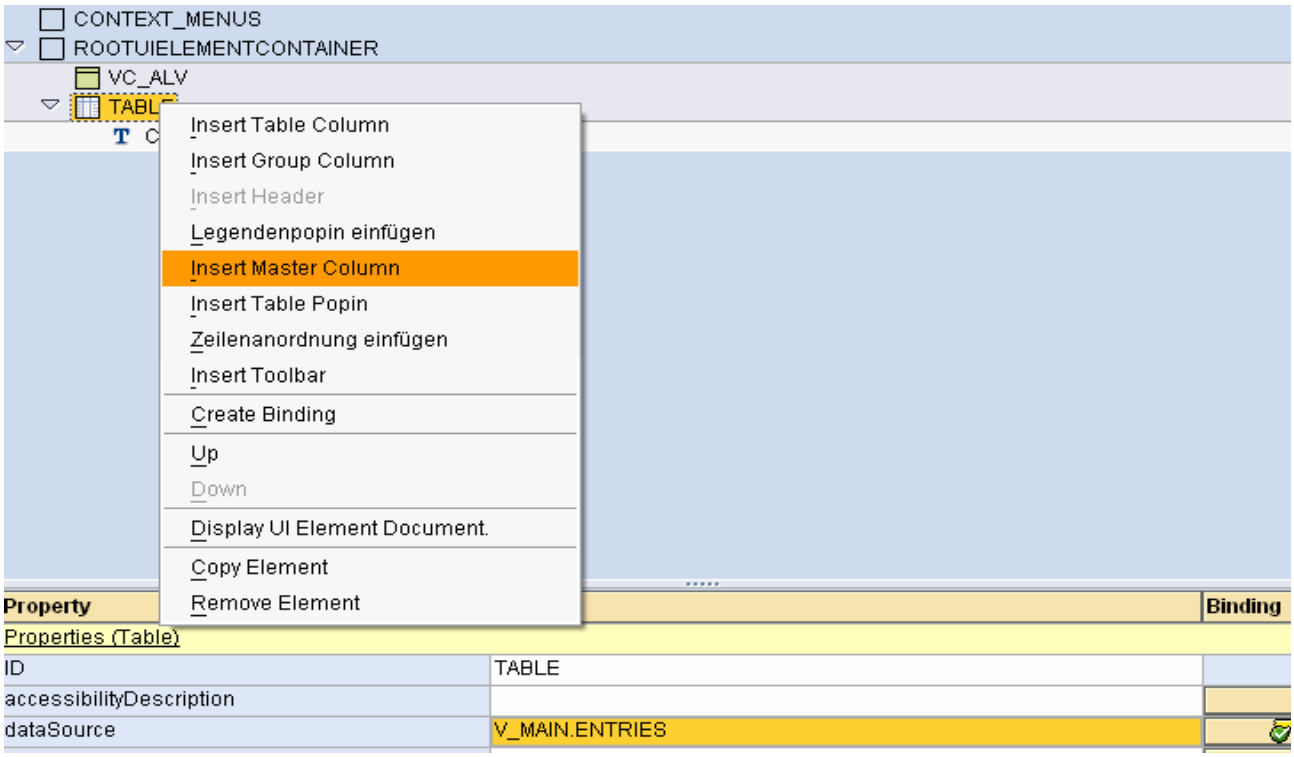
NAME type STRING, PATH type STRING, PARENT_PATH type STRING, IS_EXPANDED type WDY_BOOLEAN, IS_LEAF type WDY_BOOLEAN, CHILDREN_LOADED type WDY_BOOLEAN, VALUE type STRING.

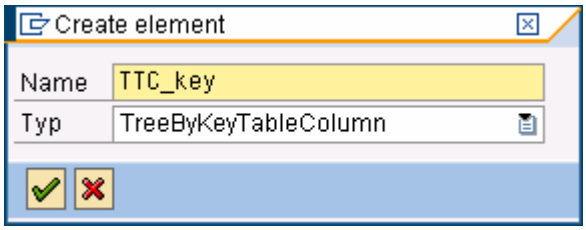
- Code to fill the values in node ENTRIES in the supply function



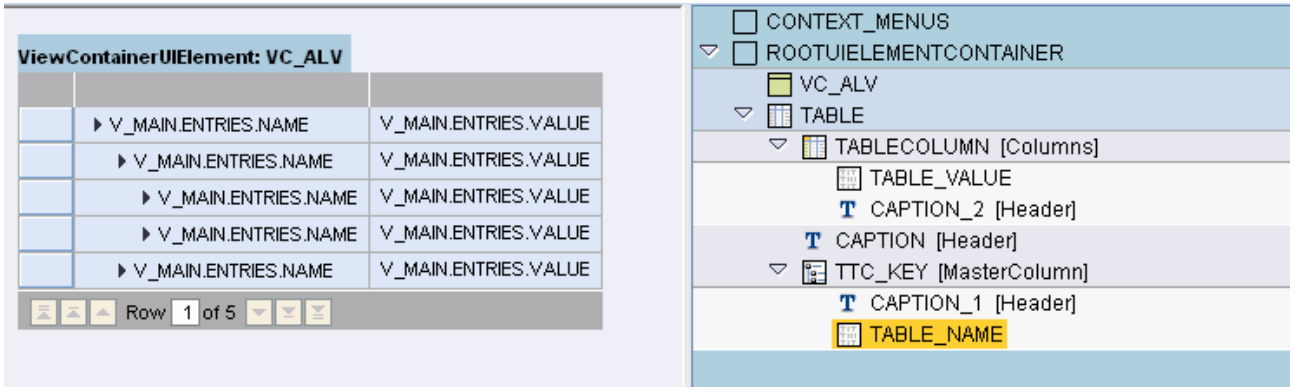

```
method SUPPLY .
  data: ls_entry type if_v_main=>element_entries.
  data: lt_entry type if_v_main=>elements_entries.

  ls_entry-name = 'SAP'.
  ls_entry-path = 'SAP'.
  ls_entry-is_expanded = abap_false.
  ls_entry-is_leaf = abap_false.
  ls_entry-value = 'Systems Applications'.
  append ls_entry to lt_entry.

  ls_entry-name = 'Sales and Distribution'.
  ls_entry-path = 'SD'.
  ls_entry-parent_path = 'SAP'.
  ls_entry-is_expanded = abap_false.
  ls_entry-is_leaf = abap_true.
  ls_entry-value = 'Logistics'.
  append ls_entry to lt_entry.

  ls_entry-name = 'Material Management'.
  ls_entry-path = 'MM'.
  ls_entry-parent_path = 'SAP'.
  ls_entry-is_expanded = abap_false.
  ls_entry-is_leaf = abap_true.
  ls_entry-value = 'Logistics'.
  append ls_entry to lt_entry.

  ls_entry-name = 'ORACLE'.
  ls_entry-path = 'ORACLE'.
  ls_entry-parent_path = ''.
  ls_entry-is_expanded = abap_false.
  ls_entry-is_leaf = abap_false.
  ls_entry-value = 'Oracle Applications'.
  append ls_entry to lt_entry.


  node->bind_table( lt_entry ).
```

- Design the Layout, place a table control and bind the data source property with the created node
- Insert Master Column under the Table control node





- Insert cell editor under the master column and bind it with the NAME attribute of ENTIRES node
- Also insert a Table column to show VALUE attribute of ENTRIES node.



- Set the Properties as follows for the Master Column Created. isLeaf with IS_LEAF and expanded with IS_EXPANDED, childrendLoaded with CHILDREN_LOADED, parentKey with PARENT_PATH, rowKey with PATH attributes respectively.

Property	Value	Binding
Properties (TreeByKeyTableColumn)		
ID	TTC_KEY	
accessibilityDescription		
cellDesign	standard	
childrenLoaded	☐	
expanded	☐	
isLeaf	☐	
parentKey	V_MAIN.ENTRIES.PARENT_PATH	
resizable	☑	
rowKey	V_MAIN.ENTRIES.PATH	
visible	Visible	
width		

- Activate and run the application

NOTE: SET_LAST_HIER_COLUMN_AS_LEAF method not present in SP10

Summary

Tree structure is possible in ALV to a limited extend as the framework itself handles the hierarchy unlike standard table. You can not hide the master columns using settings in the ALV. The ALV output is automatically sorted according to all hierarchy columns.

Related Content

Table as Hierarchy

For more information, visit the User Interface Development with Web Dynpro for ABAP Page .

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.