Simple Example for Using ALV in Web Dynpro for ABAP

Release SAP NetWeaver 2004s

Copyright

Icons in Body Text

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help* → *General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Typographic Conventions

Type Style	Description
Example text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
Example text	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
Example text	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
EXAMPLE TEXT	Keys on the keyboard, for example, F2 or ENTER.

Table of Contents

Using an ALV Table in Web Dynpro

In this tutorial an ALV table is used to display the content of a context node.

Task

The task of this simple tutorial is to use an ALV table inside a window to display the content of context node *NODE_FLIGHTTAB*.

Objectives

By the end of this tutorial, you will be able to:

- ✓ Use an ALV inside a window

Knowledge

- ☐ Knowledge of ABAP OO programming language
- ☐ Basic knowledge of programming Web Dynpro applications
- ☐ Basic knowledge of ABAP workbench

Creating a New Web Dynpro Component and Building up the Component Controller Context

This tutorial starts with the creation of a new Web Dynpro component called **Z00_WDT_FLIGHTLIST_SIMPLE**.

Afterwards the component controller context is built up.

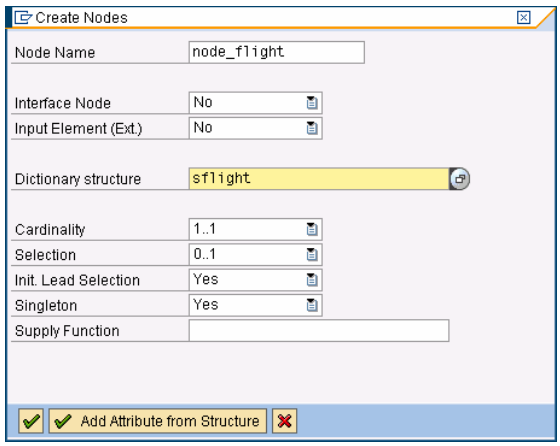
Procedure

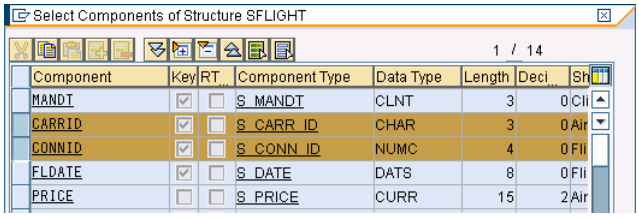
Create a new Web Dynpro component.

Start the ABAP Workbench (*se80*) and create the new Web Dynpro component *Z00_WDT_FLIGHTLIST*_SIMPLE. Use *MAIN* as window name. And delete the view name.

Create context element in component controller for storing the search criteria.

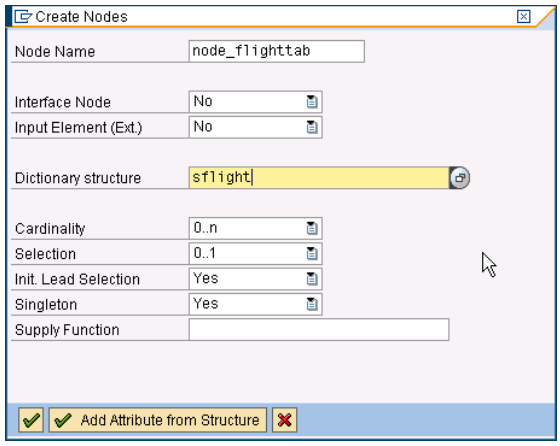
Create node *NODE_FLIGHT* based on dictionary structure *SFLIGHT* in the context of the component controller. Click on button *Add Attribute from Structure* and choose fields *CARRID* and *CONNID*.

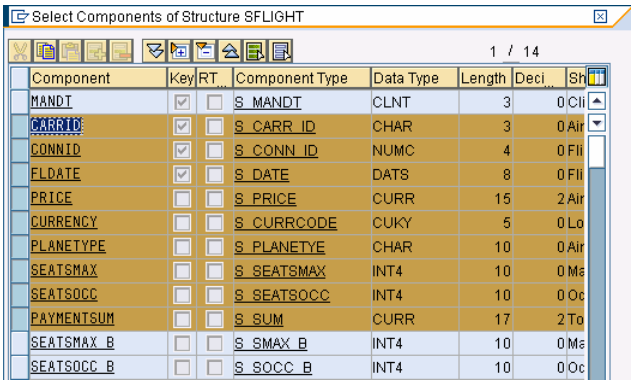




Create a context element in component controller for storing the flight list.

Create node *NODE_FLIGHTTAB* based on dictionary structure *SFLIGHT* in the context of the component controller. Set the cardinality to 0..N. Click on button *Add Attribute from Structure* and choose fields *CARRID, CONNID, FLDATE, PRICE, CURRENCY, PLANETYPE, SEATSMAX,SEATSOCC* and *PAYMENTSUM*.





An ALV table always displays all fields that are available in the dictionary, if the context node has a dictionary reference. To only display the selected fields inside the ALV table clear the dictionary reference in property *DICTIONARY STRUCTURE* in node *NODE_FLIGHTTAB*.

Property	Value
Nodes	
Node Name	NODE_FLIGHTTAB
Interface Node	☐
Input Element (Ext.)	☐
Dictionary structure	
Cardinality	0..n
Selection	0..1
Initialization Lead Selection	☑
Singleton	☑

Create a method for filling the flight list.

Create method FILL_*FLIGHTTAB* in the component controller to fill node *NODE_FLIGHTTAB*. (This method will be called later from out of the *SEARCHVIEW.*)

It is important that the table data of context node *NODE_FLIGHTTAB* is not copied. In order to ensure this, the variable holding the table (lt_flights) that is handed over to the context node via *BIND_TABLE* must be a local variable.

```
FILL_FLIGHTTAB()
```

```
method FILL_FLIGHTTAB .
* fill context node "node_flighttab"

  DATA:
    node_node_flight     TYPE REF TO if_wd_context_node,
    node_node_flighttab  TYPE REF to if_wd_context_node,
    elem_node_flight     TYPE REF TO if_wd_context_element,
    stru_node_flight     TYPE if_componentcontroller=>element_node_flight,

    ls_where(72)         TYPE c,
    lt_where             LIKE TABLE OF ls_where,

    lt_flights           TYPE TABLE OF sflight.

* navigate from <CONTEXT> to <NODE_FLIGHT> via lead selection
  node_node_flight = wd_context->get_child_node( name = `NODE_FLIGHT` ).

* get element via lead selection
  elem_node_flight = node_node_flight->get_element(  ).

* get all declared attributes
  elem_node_flight->get_static_attributes(
    IMPORTING
      static_attributes = stru_node_flight ).

* create where condition
  IF NOT stru_node_flight-carrid EQ ''.
    CONCATENATE 'CARRID = ''' stru_node_flight-carrid '''' INTO ls_where.
    APPEND ls_where TO lt_where.
  ENDIF.

  IF NOT stru_node_flight-connid EQ '0000'.
    CONCATENATE 'CONNID = ''' stru_node_flight-connid '''' INTO ls_where.
    IF stru_node_flight-carrid NE ''.
      CONCATENATE 'AND' ls_where INTO ls_where separated by space.
    ENDIF.
    APPEND ls_where TO lt_where.
  ENDIF.

* read data
  select * from sflight into table lt_flights
    WHERE (lt_where).

* navigate from <CONTEXT> to <NODE_FLIGHT> via lead selection
  node_node_flighttab = wd_context-
>get_child_node( name = `NODE_FLIGHTTAB` ).

* fill context node
  node_node_flighttab->bind_table( lt_flights ).

endmethod.
```

Search View

The view *SEARCHVIEW* will be used to enter the search criteria for the database selection that fills context node *NODE_FLIGHTTAB*. The search criteria will be stored inside context node *NODE_FLIGHT*.

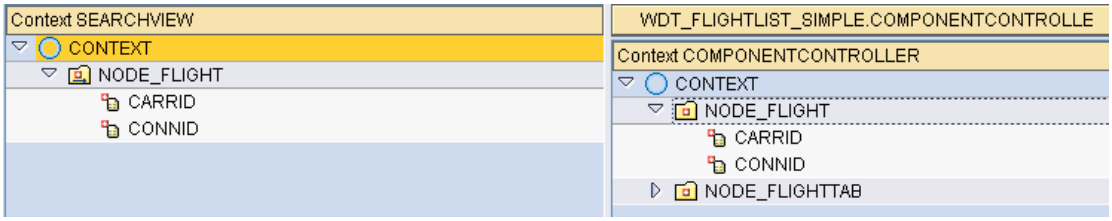
Procedure

Create view for searching.

Create view SEARCHVIEW.

Map context element for storing search criteria.

Copy and map context node *NODE_FLIGHT* from the component controller's context to the context of view *SEARCHVIEW*.

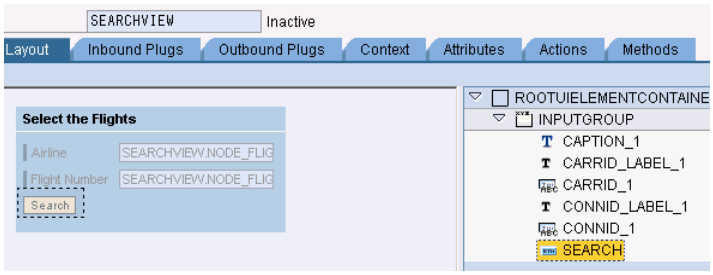


Create the layout of the search view

In the layout of view *SEARCHVIEW* create a group called *INPUTGROUP*. Enter "Select the Flights" in the *TEXT* property of *CAPTION_1* of group *INPUTGROUP*.

Inside *INPUTGROUP* create labels and input fields for the two attributes *CARRID* and *CONNID* of context node *NODE_FLIGHT*. Hint: Use function *Create Container Form* out of the context menu of *INPUTGROUP*.

Create button *SEARCH* inside group *INPUTGROUP*. Enter "Search" as text. Create action *SEARCH* and invoke component controller method *FILL_FLIGHTTAB()* in the corresponding method *ONACTIONSEARCH*.



ONACTIONSEARCH()

```
method ONACTIONSEARCH .
  wd_Comp_Controller->Fill_Flighttab( ).
endmethod.
```

Layout View

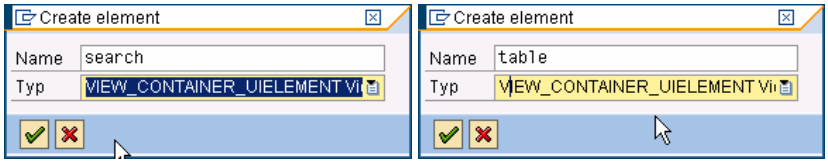
To display more than one view at the same time inside a window, we need a view for creating the layout.

Procedure

Create Layout View.

Create view LAYOUTVIEW and navigate to the layout tab. Change the *Layout* property of the ROOTUIELEMENTCONTAINER to MatrixLayout.

Create two new elements, VIEW_CONTAINER_UIEELEMENT, each as a child of the ROOTUIELEMENTCONTAINER. Name them SEARCH and TABLE.

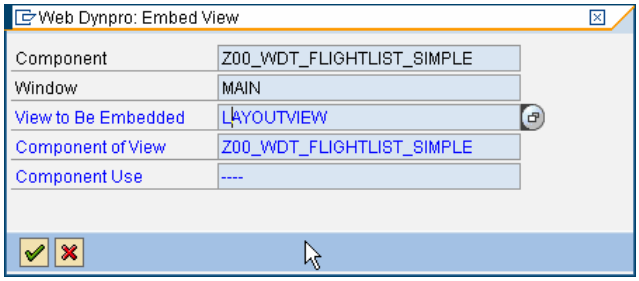


Change the *Layout Data* property of both view container UI elements to *MatrixHeadData*.

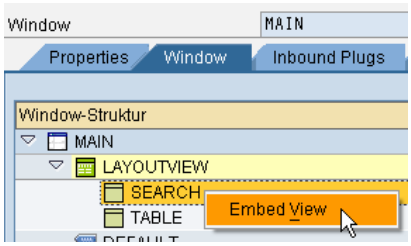
Assign LAYOUTVIEW and SEARCHVIEW to window.

It's not possible to display a view directly inside a browser, but a window. Therefore the views just created have to be assigned to the window *MAIN*.

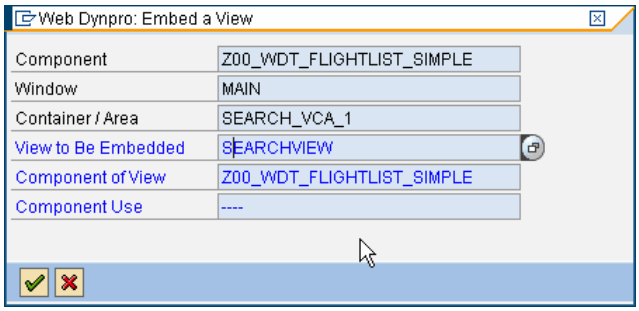
Open the window MAIN and navigate to the window tab. Open the context menu of the root element *MAIN* and choose *Embed View.* First embed view LAYOUTVIEW:



After view LAYOUTVIEW has been embedded into window MAIN, open the context menu of the view container SEARCH and choose *Embed* View.



Embed view SEARCHVIEW:



Using the Component ALV Table in the Component Controller

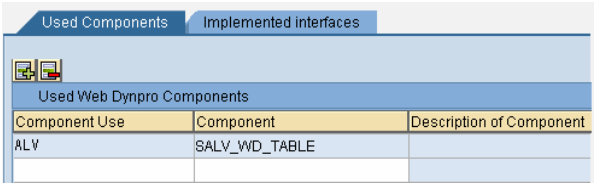
When you search for special flights by entering a carrier and a connection, you want to see the result in an ALV table.

Procedure

Define component usage.

If you want to see the data within a Web Dynpro ALV, you have to define the Web Dynpro component for ALV **SALV_WD_TABLE** as a usage component of your Web Dynpro component.

Double-click on your component and define the component usage of SALV_WD_TABLE as follows:

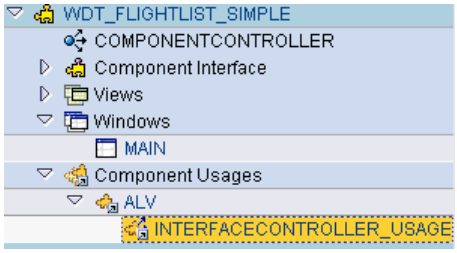


Set data to ALV for display (via reverse context mapping).

The selected flights are inside context node *NODE_FLIGHTTAB*. To display them in the ALV it is necessary to map the context node *NODE_FLIGHTTAB* to the context node *DATA* of the ALV interface controller.

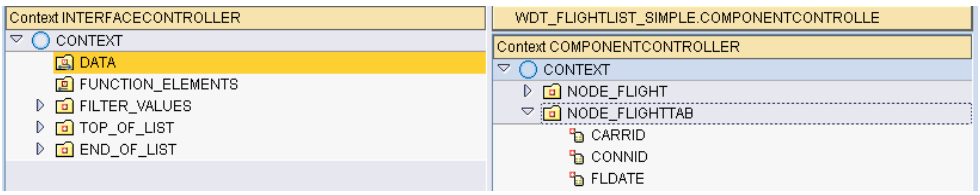
Open your Web Dynpro component's node *Component Usages -> ALV -> INTERFACECONTROLLER_USAGE*.

Hint: If the node *Component Usage* is not displayed in the object tree, save your component and refresh the tree display.



Click on the Controller Usage button. The component controller of your Web Dynpro component appears on the right side of the screen.

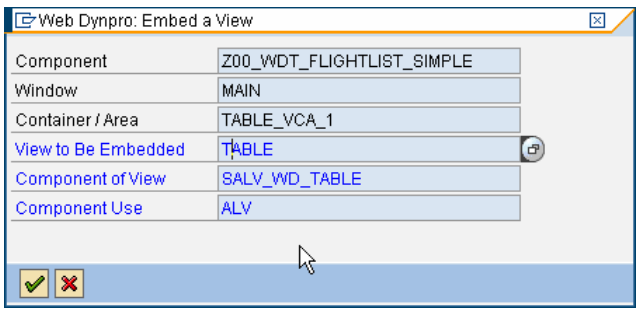
Map context node NODE_*FLIGHTTAB* of your Web Dynpro component to context *DATA* of the interface controller of the ALV component.

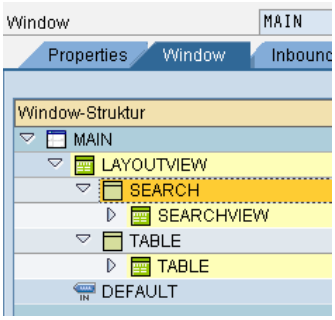


After you have done the mapping, the context node *DATA* has a double-arrow inside the folder icon.

Embed view TABLE of component SALV_WD_TABLE into LAYOUTVIEW.

To embed the ALV table into the *LAYOUTVIEW*, go to the window *MAIN* and switch to tab page *window.* Embed the ALV view *TABLE* to view container *TABLE* by choosing the context menu entry *embed view.* A popup appears. Use the F4 help and select the following entry.





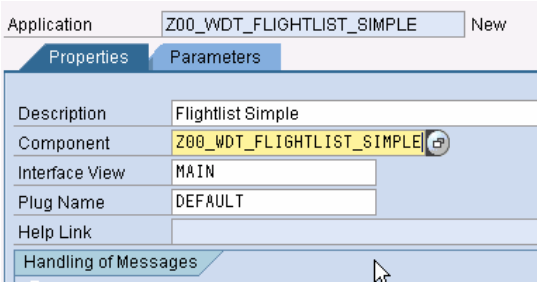
Activate your Web Dynpro Component.

Create and Test Web Dynpro Application

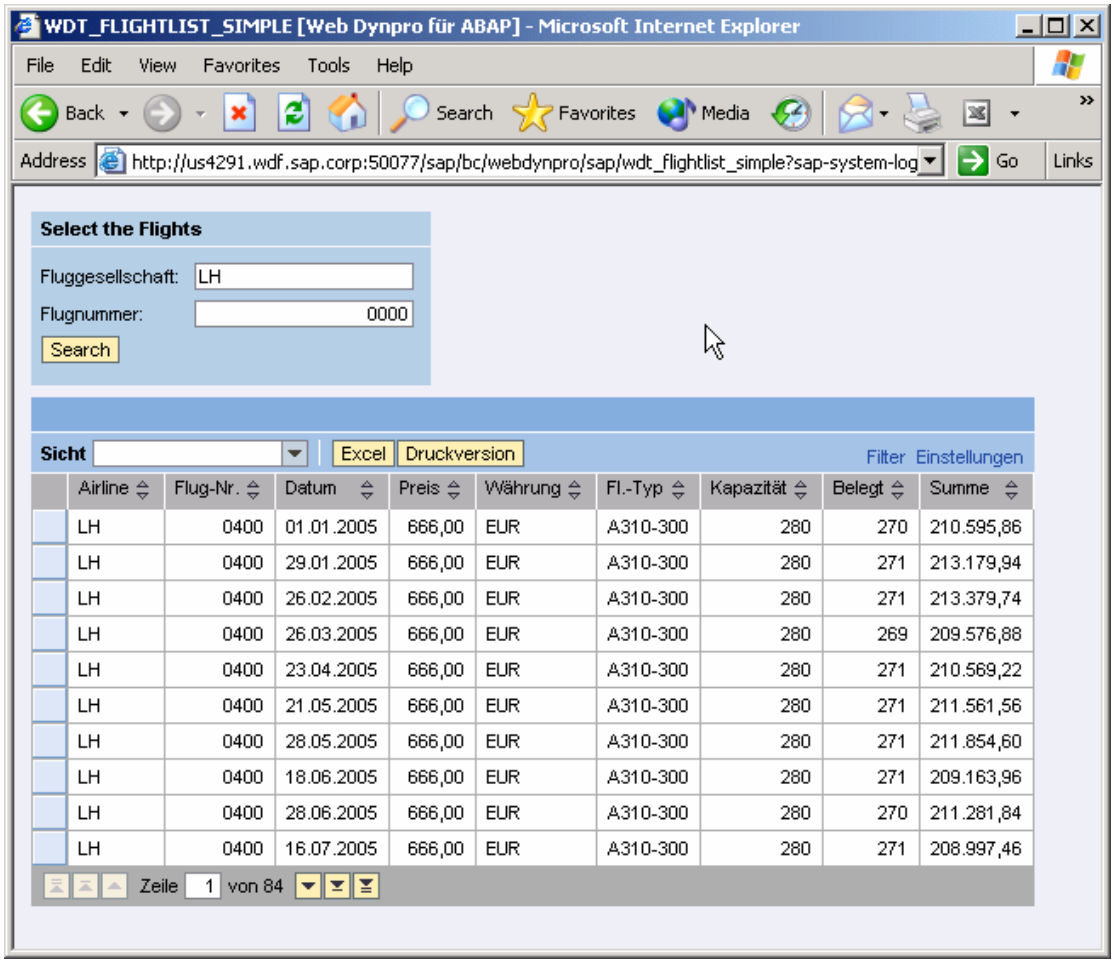
Each Web Dynpro component needs a Web Dynpro application to be executed.

Procedure

Create a Web Dynpro application for your Web Dynpro component:



Test your Web Dynpro application. The result will look like the following:



Author Bio

Claudia Dangers is a senior development consultant in SAP's Software Technology and Development department. Since she joined SAP in 1999 she has worked on numerous projects and gained practical experience in ABAP and BSP development, in the creation of concepts, in coaching and code reviews, and as a sub-project lead and training instructor. Claudia is very interested in new technologies. Currently she is dealing with Web Dynpro ABAP, kernel-based BADI's and the Switch and Enhancement Framework.