

# Web Dynpro for ABAP: Tutorial 3 - Navigation



**SAP NetWeaver 04s**



## Copyright

© Copyright 2005 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.






JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

## 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
<i>Example text</i>	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.
<b>Example text</b>	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.
<b>Example text</b>	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.

---

Web Dynpro for ABAP: Tutorial 3 – Navigation.....	5
Development Objectives.....	5
Procedure .....	6
Creating a new Web Dynpro Component ZZ_00_BAPINAV .....	6
Creating a new View NOFLIGHTSVIEW with corresponding text field and back button ...	7
Creating the Context nodes, Mapping and Attributes .....	9
Define Inbound and Outbound plugs for navigation between two views.....	12
Enhance the coding of methods using Web Dynpro code wizard .....	13
Create the action for the back button on NOFLIGHTSVIEW.....	18
Embed the View into the window .....	19
Activation, Creation of a Web Dynpro Application and Execution .....	21
Result.....	22
SAP Online Help.....	22

---

# Web Dynpro for ABAP: Tutorial 3 – Navigation

## Development Objectives

This tutorial demonstrates how easy it is to navigate between different views of the same Web Dynpro Application. It is shown by extending tutorial Web Dynpro for ABAP - BAPI Usage with additional view in order to display a message in the case that no flights are available for a particular start and destination combination. Ofcourse, you will be able to return back to search for a new combination.

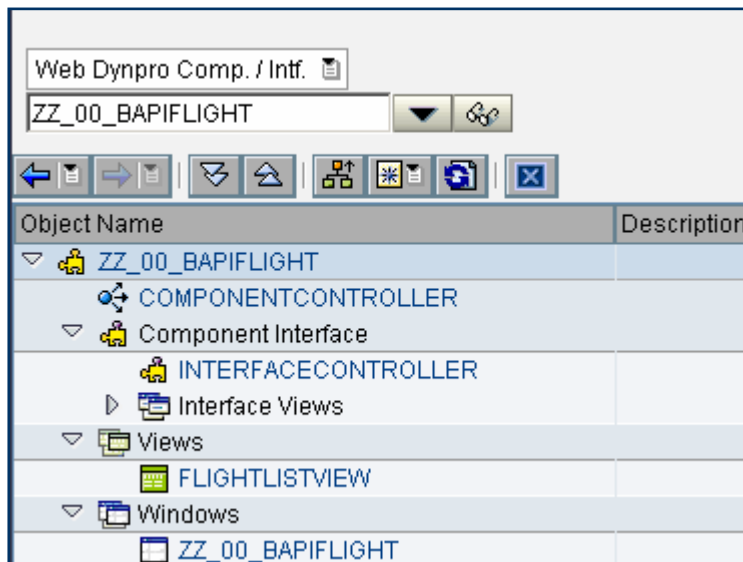
## Procedure

- Copy the Web Dynpro component `ZZ_00_BAPIFLIGHT` from Tutorial 2 to a new component with name `ZZ_00_BAPINAV`. Adjust the name of the window according to the component name.
- Create new view `NOFLIGHTSVIEW` for the message with a text field of type `TextView` and a back button. In UI element properties, set the layout data of the button `cellDesign = IPad` and `vGutter = medium`. The text on the button is *New Search*.
- Create the context nodes for `NOFLIGHTSVIEW` in order to get start and destination for the message. Map the nodes `DESTINATION_FROM` and `DESTINATION_TO` from the component controller to the view `NOFLIGHTSVIEW`. Add a further context attribute `TEXT` of type `STRING` to the view context.
- Define context binding between UI text field and context field `TEXT`.
- Define the plugs for the navigation in a way that that you can move from `FLIGHTLISTVIEW` to `NOFLIGHTSVIEW` and vice versa.
- Modify the coding of method `ONACTIONGET_FLIGHTS` as follows with the Web Dynpro code wizard.
  - Read context node `FLIGHT_LIST` and start navigation to view `NOFLIGHTLIST` if the node `FLIGHT_LIST` is initial.
  - Build a message string saying 'No flights available between <city from> 'to' <city to> '.' to display on `NOFLIGHTSVIEW`.
- Create the action for the back button on the `NOFLIGHTSVIEW` view and check the implementation of the method.
- Embed the `NOFLIGHTSVIEW` view in the window.
- Generate the Web Dynpro application and test it.

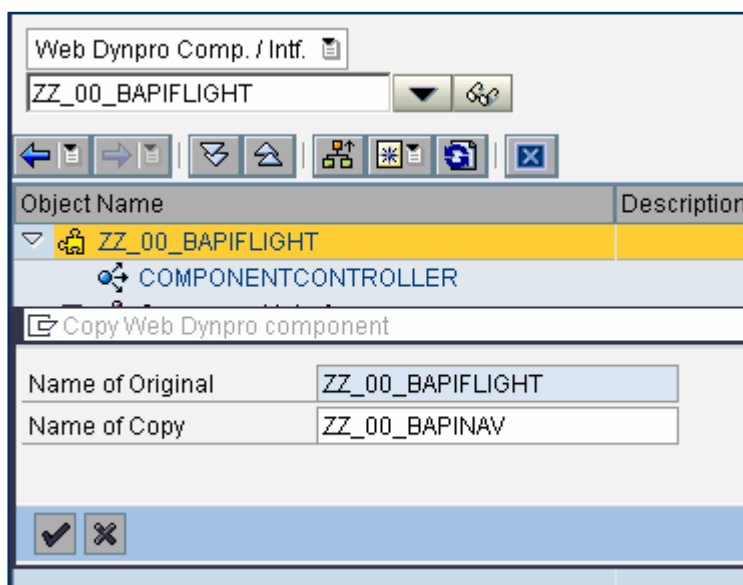
## Create the Web Dynpro component ZZ\_00\_BAPINAV

### Procedure

1. Copy the Web Dynpro component ZZ\_00\_BAPIFLIGHT from Tutorial 2 to a new component with name ZZ\_00\_BAPINAV. Adjust the name of the window according to the component name.



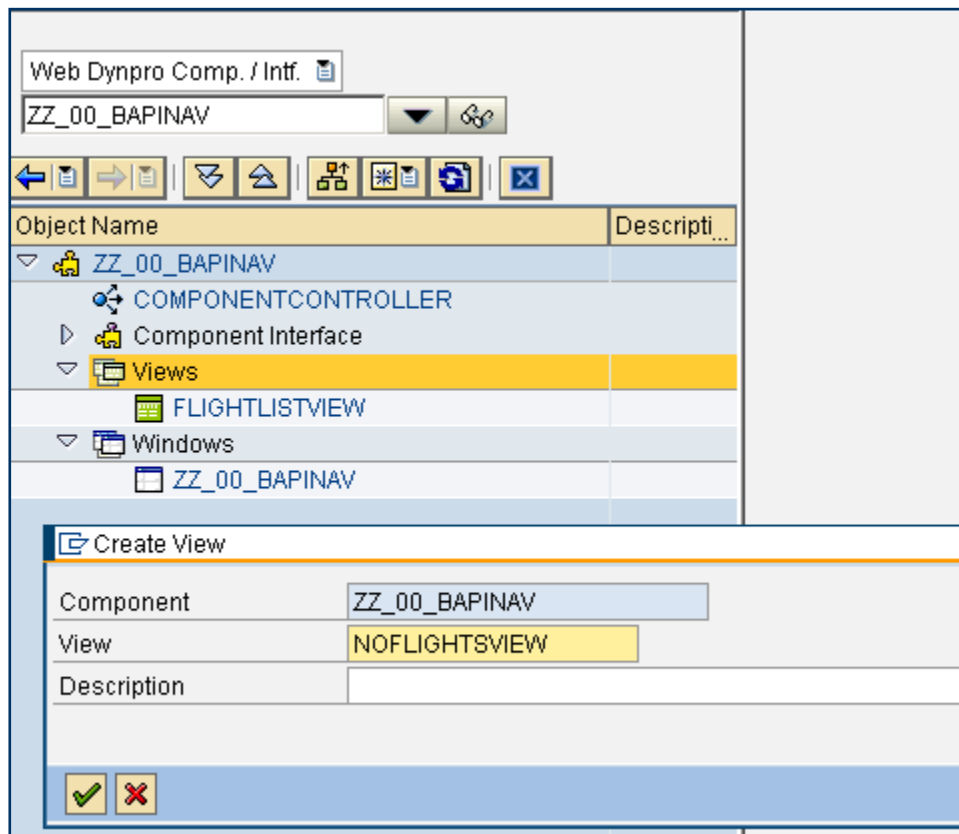
2. Please note that when copying Web Dynpro components the name of the window remain the old one. If you bother about that you can change the window name to the name of the component with the rename function.



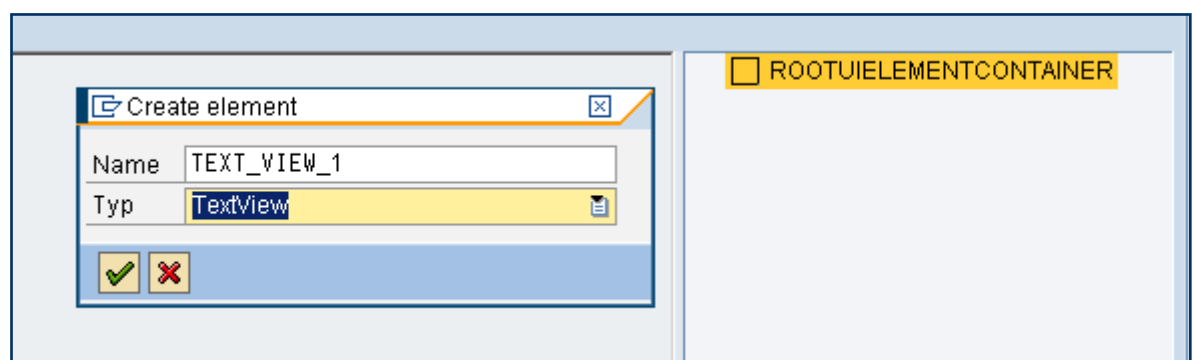
## Create new view NOFLIGHTSVIEW

### Procedure

1. Create a new view NOFLIGHTSVIEW for the message with a text field and a button with the text *New Search*. Select *Views*, open the context menu and choose *Create*. Then the create view dialog pops up asking for the new view name.



2. After clicking the *OK* button, the View Composer will start automatically. Open the context menu of *ROOTUIELEMENTCONTAINER*. Click on *Create element*. The dialog popup will start and you can now enter the name and the type.



Press the *Continue (Enter)* button.

- The new UI element will appear under the element *ROOTUIELEMENTCONTAINER*. In the same way you now can create the button with the name *BUTTON\_1*. The text in the button should be *New Search*. To separate the button from the text field set the Layout Data as shown in the next screenshot.

The screenshot shows the object inspector for a button named **BUTTON\_1**. The hierarchy is **ROOTUIELEMENTCONTAINER** > **TEXT\_VIEW\_1** > **BUTTON\_1**. The properties table is as follows:

Property	Value	Bin...
<b>Properties (Button)</b>		
ID	BUTTON_1	
design	standard	
enabled	<input checked="" type="checkbox"/>	
explanation		
imageFirst	<input checked="" type="checkbox"/>	
imageSource		
text	New Search	
textDirection	inherit	
tooltip		
visible	visible	
width		
<b>Events</b>		
onAction		
<b>Layout Data (FlowData)</b>		
cellDesign	IPad	
vGutter	medium	

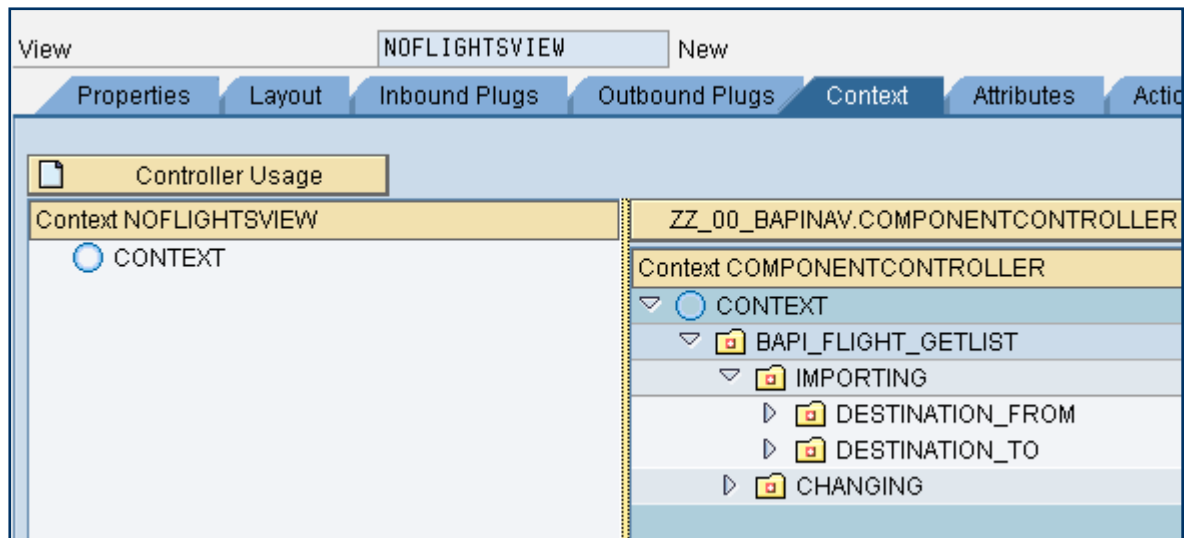
The **cellDesign** and **vGutter** properties are highlighted with a red box in the original image.



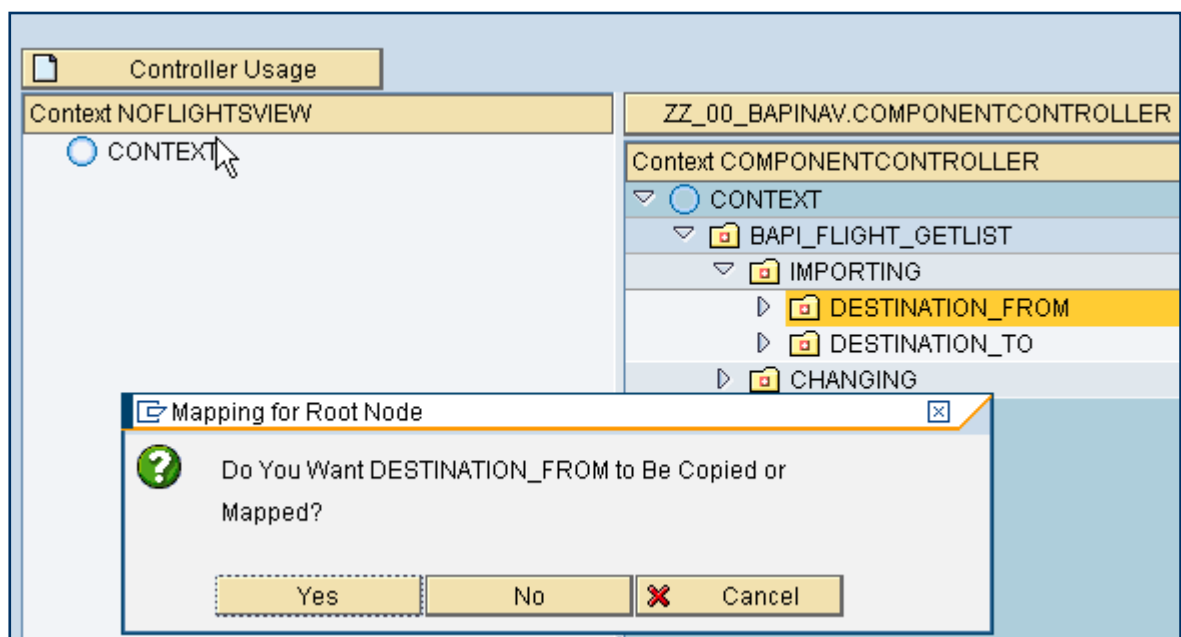
## Create the Context Nodes for NOFLIGHTSVIEW, define Context Mapping and Add attributes

### Procedure

1. Create the context nodes for NOFLIGHTSVIEW in order to get start and destination for the message. Go to the tab *Context* of the newly created view and open the node BAPI\_FLIGHT\_GETLIST of the componentcontroller context.

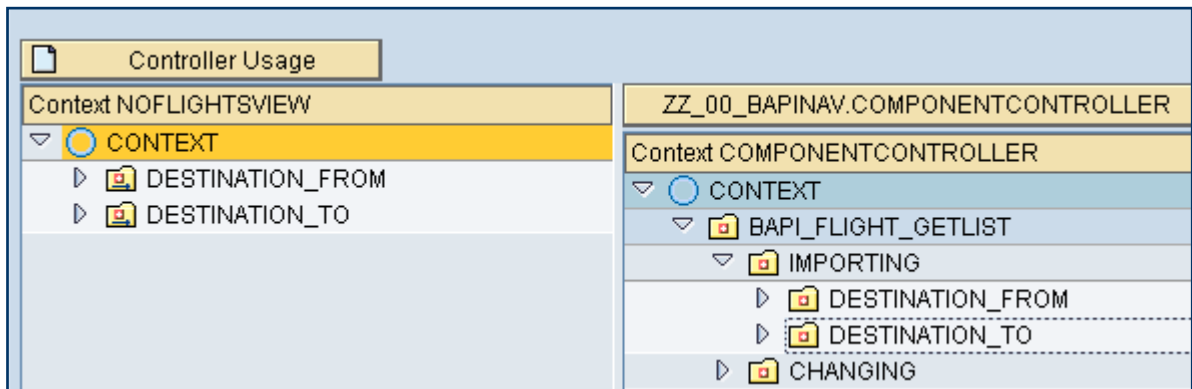


2. Using Drag and Drop you can easily map the two nodes *DESTINATION\_FROM* and *DESTINATION\_TO* from the Componentcontroller context to the view context. When dropping the node on the context of the view a popup will ask once more, whether you really want to copy and map this node.

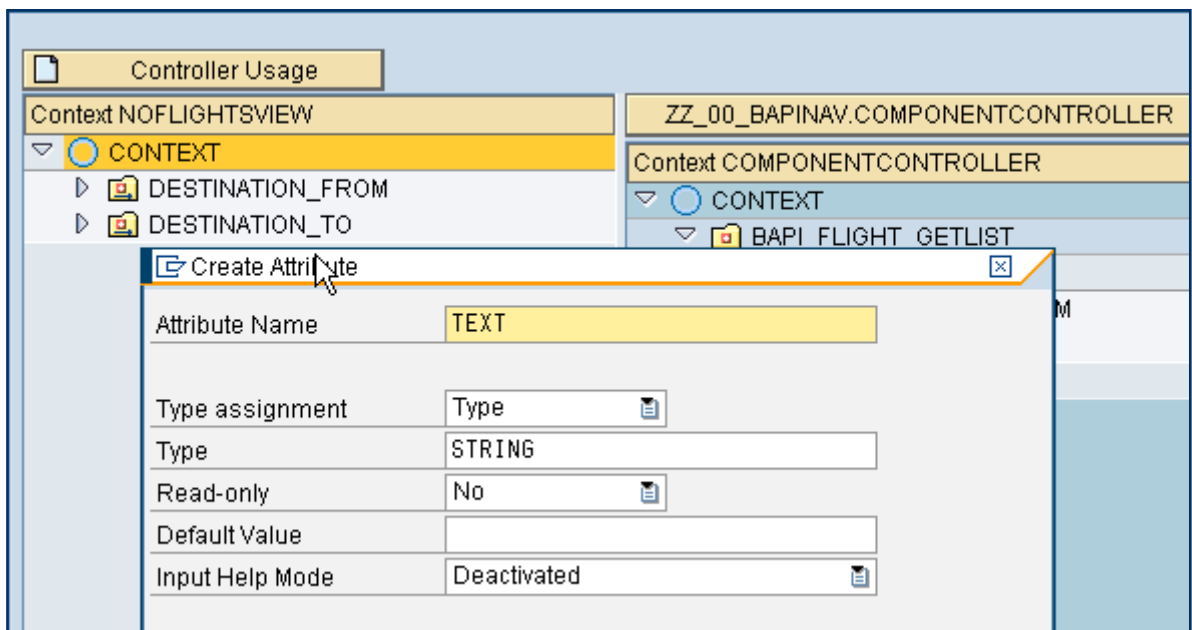




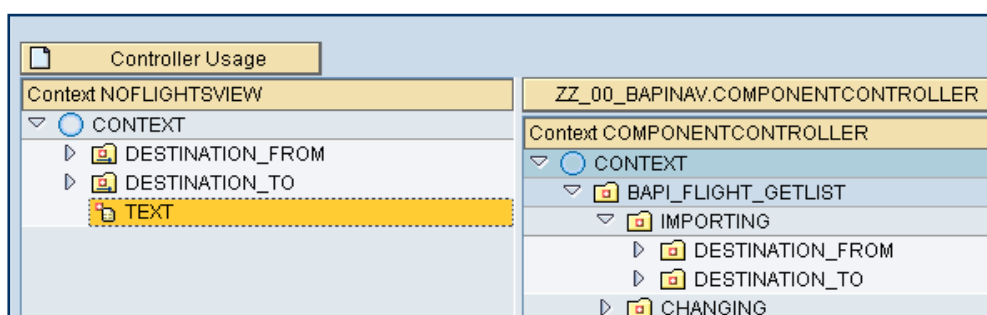
Select Yes. After mapping the two nodes to the view your view context should look like the screenshot below.



- Now one more context attribute must be created for the message text that will be displayed on the newly created view. Open the context menu of the view context and select *Create* → *Attribute*. A dialog popup will be opened for the necessary input. Create a new attribute named *TEXT* of the type *STRING*:



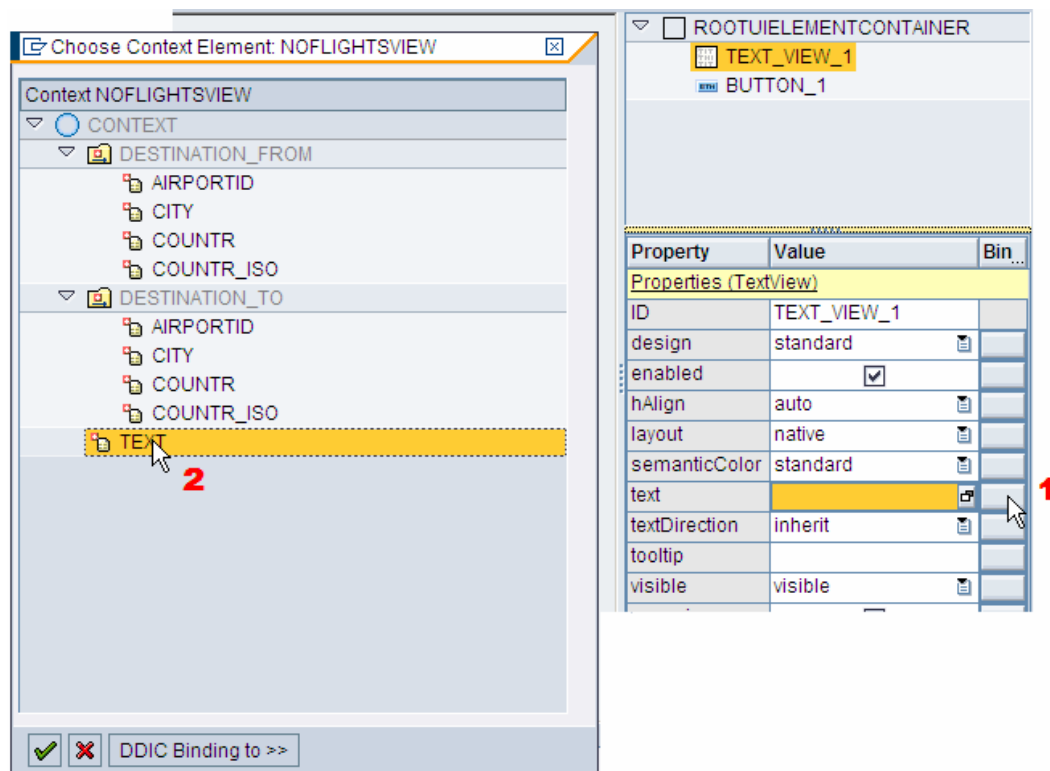
Your view context should now look like this:



## Define context binding between UI text field and context field *TEXT*

### Procedure

1. Create context binding between property *text* of UI element *TEXT\_VIEW\_1* and context attribute *TEXT*. Go to the tab *Layout* of the view *NOFLIGHTSVIEW*. Select the text field and click on the binding button (1) of the property *text*. A dialog popup will be opened where you now can select the new attribute from the context (2).



When the binding is done the property *text* of the *TEXT\_VIEW\_1* will look like the screenshot below.

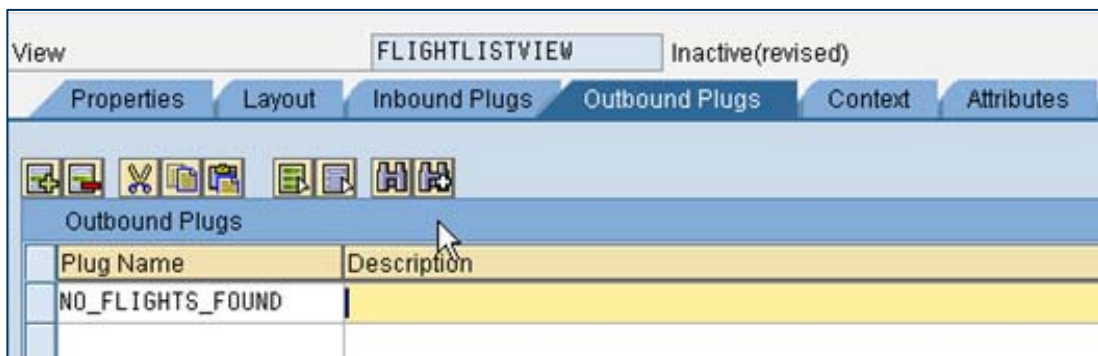
Property	Value	Bind...
<b>Properties (TextView)</b>		
ID	TEXT_VIEW_1	
design	standard	
enabled	<input checked="" type="checkbox"/>	
hAlign	auto	
layout	native	
semanticColor	standard	
text	NOFLIGHTSVIEW.TEXT	<input checked="" type="checkbox"/>
textDirection	inherit	
tooltip		
visible	visible	
wrapping	<input type="checkbox"/>	

## Define the plugs for the navigation between Views FLIGHTLISTVIEW to NOFLIGHTSVIEW

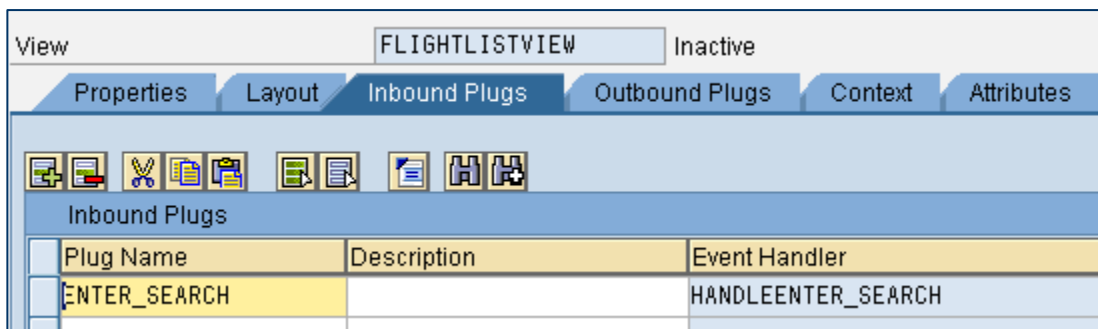
### Procedure

Create the plugs for the navigation in a way that both to and from NOFLIGHTSVIEW navigation is possible.

1. On the FLIGHTLISTVIEW you need an outbound plug. Go to the FLIGHTLISTVIEW click on tab *Outbound Plugs* and create the plug *NO\_FLIGHTS\_FOUND*. Save FLIGHTLISTVIEW.



2. To come back from the NOFLIGHTSVIEW for a new search an inbound plug is needed. Click on tab *Inbound Plugs* and create a plug *ENTER\_SEARCH*.



When creating an inbound plug an event handler will be created automatically. The name of the handler always starts with 'HANDLE' followed by the name of the plug.

3. On NOFLIGHTSVIEW you create an inbound plug *NO\_FLIGHTS\_FOUND* and an outbound plug *BACK\_TO\_SEARCH* in the same way as you did with the plugs of view FLIGHTLISTVIEW.

## Enhance the coding of methods using Web Dynpro code wizard.

### Procedure

1.

- a) Modify the coding of method ONACTIONGET\_FLIGHTS in order to navigate to the new view if there are no flights available.

If node *FLIGHT\_LIST* is empty the navigation will lead to view NOFLIGHTSVIEW. Go to view FLIGHTLISTVIEW and click on tab *Methods*. Start the editor for method ONACTIONGET\_FLIGHTS. Position the cursor behind the component controller call and start the code wizard for reading the node FLIGHT\_LIST node.

The screenshot displays the SAP Web Dynpro IDE interface. At the top, the view is identified as 'FLIGHTLISTVIEW' (Inactive(revised)). The 'Web Dynpro Statement Structure' dialog is open, showing various options for method generation. The 'Read Context' option is selected, with the 'Node/Attribute' field set to 'FLIGHT\_LIST'. Other options are unselected. The code editor shows the method signature 'method ONACTIONGET\_FLIGHTS' and a call to 'wd\_Comp\_Controller->E'. The 'Method List' on the left shows the 'ONACTIONGET\_FLIGHTS' method selected.

Select the radio button *Read Context* and insert the node name *FLIGHT\_LIST*. Press the *OK* button.

Now the wizard generates the code to access the node *FLIGHT\_LIST* in order to find out whether there are flights available or not. The coding now should look like the following screenshot:

```
method ONACTIONGET_FLIGHTS .
    wd_Comp_Controller->Execute_Bapi_Flight_Getlist(
    ).
    data:
        Node_Flight_List          type ref to If_Wd_Context_Node,
        Elem_Flight_List          type ref to If_Wd_Context_Element,
        Stru_Flight_List          type If_Flightlistview=>Element_Flight_List .
    * navigate from <CONTEXT> to <FLIGHT_LIST> via lead selection
    Node_Flight_List = wd_Context->get_Child_Node( Name = `FLIGHT_LIST` ).

    * @TODO handle not set lead selection
    if ( Node_Flight_List is initial ).

        endif.

    * get element via lead selection
    Elem_Flight_List = Node_Flight_List->get_Element( ).

    * @TODO handle not set lead selection
    if ( Elem_Flight_List is initial ).

        I

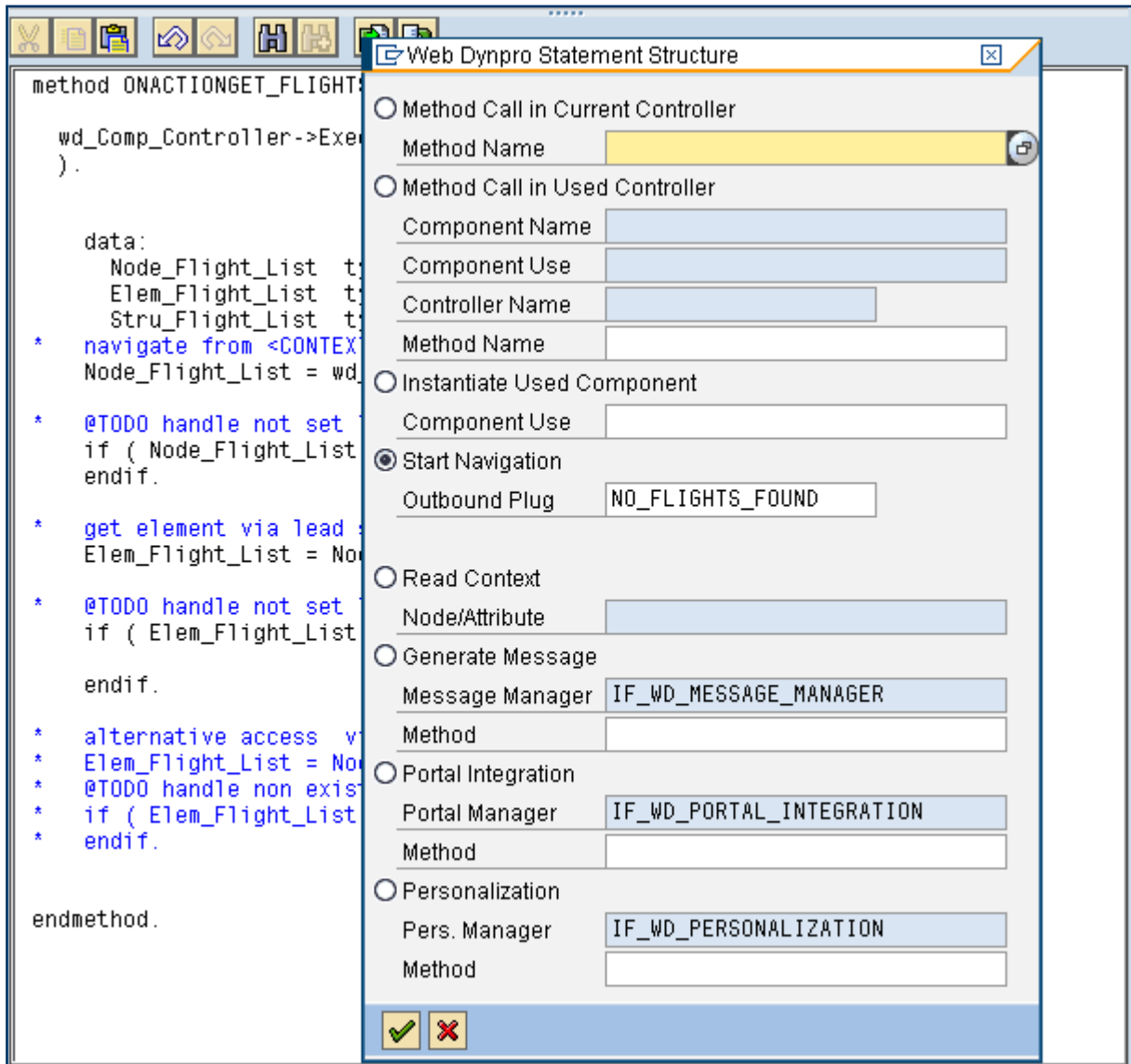
    endif.

    * alternative access via index
    * Elem_Flight_List = Node_Flight_List->get_Element( Index = 1 ).
    * @TODO handle non existant child
    * if ( Elem_Flight_List is initial ).
    * endif.

endmethod.
```



The last lines under ‘‘get all declared attributes’’ you do not need. They have to be deleted. For the navigation please position the cursor in the second if-statement and call the wizard again.



Select the radio button *Start Navigation* and enter the plug name. Click on the *OK* button.

```

method ONACTIONGET_FLIGHTS .

    wd_Comp_Controller->Execute_Bapi_Flight_Getlist(
    ).

    data:
        Node_Flight_List type ref to If_Wd_Context_Node,
        Elem_Flight_List type ref to If_Wd_Context_Element,
        Stru_Flight_List type If_Flightlistview=>Element_Flight_List .
    * navigate from <CONTEXT> to <FLIGHT_LIST> via lead selection
    Node_Flight_List = wd_Context->get_Child_Node( Name = `FLIGHT_LIST' ).

    * @TODO handle not set lead selection
    if ( Node_Flight_List is initial ).
        endif.

    * get element via lead selection
    Elem_Flight_List = Node_Flight_List->get_Element( ).

    * @TODO handle not set lead selection
    if ( Elem_Flight_List is initial ).
        | wd_This->Fire_No_Flights_Found_Plg(
        ).

        endif.

    * alternative access via index
    * Elem_Flight_List = Node_Flight_List->get_Element( Index = 1 ).
    * @TODO handle non existant child
    * if ( Elem_Flight_List is initial ).
    * endif.

endmethod.

```

Now the wizard has generated the coding in order to fire the navigation plug. In case that the node *FLIGHT\_LIST* is empty the outbound plug of the view *FLIGHTLISTVIEW* will be fired.



- b) Build a message string saying 'No flights available between' <city from> 'to' <city to> to display on NOFLIGHTSVIEW.  
Go to the view NOFLIGHTSVIEW and click on the tab *Methods*. There you can find an empty method HANDLENO\_FLIGHTS\_FOUND that was generated when creating the inbound plug. This method should be implemented like the following screenshot shows.

```

method HANDLESNO_FLIGHTS .
  data:
    Node_Destination_From      type ref to If_Wd_Context_Node,
    Elem_Destination_From     type ref to If_Wd_Context_Element,
    Stru_Destination_From     type If_Noflightsview=>Element_Destination_From ,
    Item_CITY_FROM            like Stru_Destination_From-CITY,
    Item_CITY_TO              like Stru_Destination_From-CITY,
    text                      type string.

  * navigate from <CONTEXT> to <DESTINATION_FROM> via lead selection
  Node_Destination_From = wd_Context->get_Child_Node( Name = `DESTINATION_FROM` ).

  * get element via lead selection
  Elem_Destination_From = Node_Destination_From->get_Element( ).

  * get single attribute
  Elem_Destination_From->get_Attribute(
    exporting
      Name = `CITY`
    importing
      Value = Item_City_FROM ).

  * navigate from <CONTEXT> to <DESTINATION_FROM> via lead selection
  Node_Destination_From = wd_Context->get_Child_Node( Name = `DESTINATION_TO` ).

  * get element via lead selection
  Elem_Destination_From = Node_Destination_From->get_Element( ).

  * get single attribute
  Elem_Destination_From->get_Attribute(
    exporting
      Name = `CITY`
    importing
      Value = Item_City_TO ).

  concatenate 'No flights available from' Item_City_From 'to' Item_city_to into text separated by ' '.

  wd_context->set_attribute( name = 'TEXT' value = text ).

endmethod.


```

You can use the code wizard to generate the access to the attribute *CITY* of node *DESTINATION\_FROM* and modify it for the second attribute. The *concatenate* statement assembles the string together. The method *set\_attribute* links the message to UI element.

## Create the action for the back button on the NOFLIGHTSVIEW view

### Procedure

1. Create the action for the back button on the NOFLIGHTSVIEW view and implement the method in the action handler. Go to the layout of NOFLIGHTSVIEW and click on the button *BUTTON\_1* in the list of the UI elements.

Property	Value	Binding
<b>Properties (Button)</b>		
ID	BUTTON_1	
design	standard	
enabled	<input checked="" type="checkbox"/>	
explanation		
imageFirst	<input checked="" type="checkbox"/>	
imageSource		
text	New Search	
textDirection	inherit	
tooltip		
visible	visible	
width		
<b>Events</b>		
onAction		
<b>Layout Data (FlowData)</b>		
cellDesign	padless	
vGutter	none	

In the property window click on the right button of the property *onAction*.

**Create Action**


Component: ZZ\_00\_BAPINAV

View: NOFLIGHTSVIEW

Action: BACK\_TO\_SEARCH

Description:

Select an outbound plug or enter an outbound plug for leaving the view by selecting the pushbutton

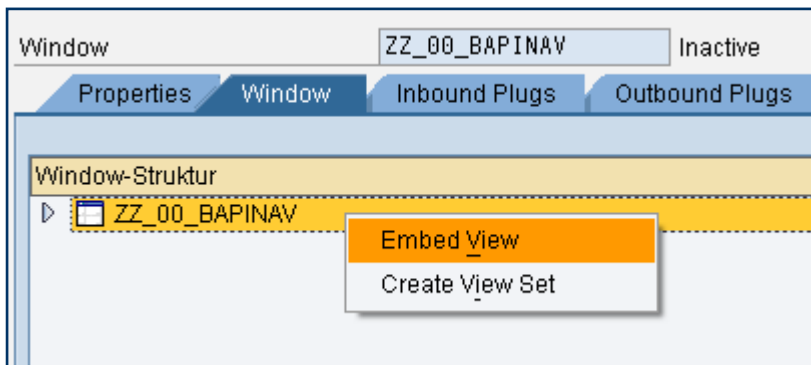
Outbound Plug: BACK\_TO\_SEARCH 

Fill in the action name and the plug that should be fired at the event. This will automatically create implementation of the event handler `ONACTIONBACK_TO_SEARCH`.

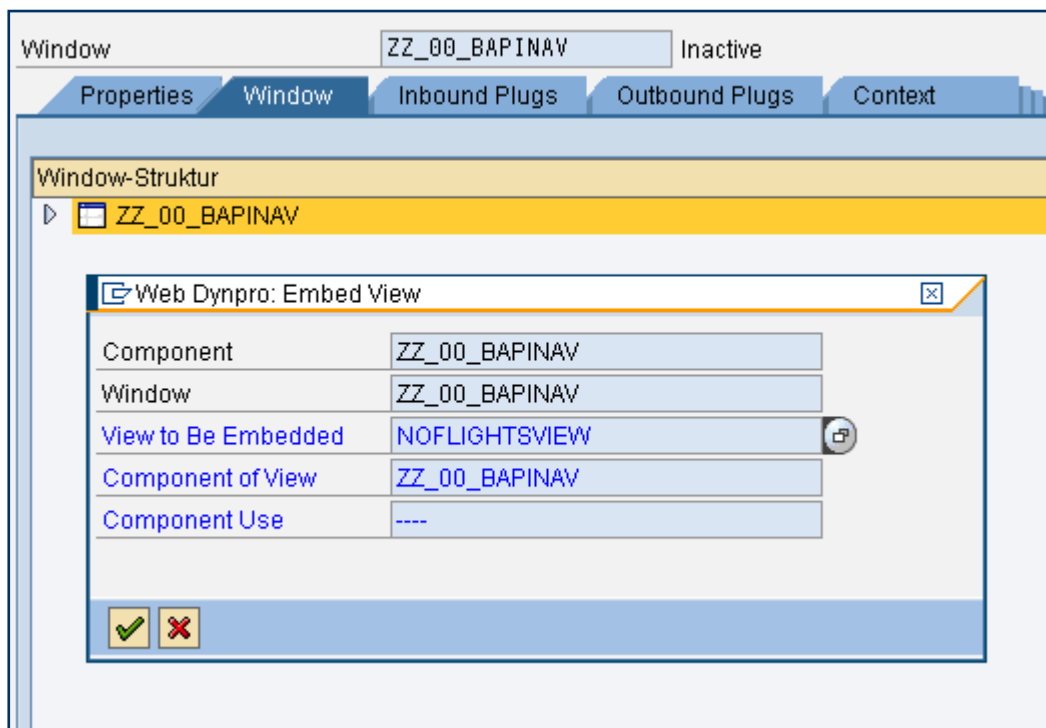
## Embedding the View into the Window

### Procedure

1. Embed the view `NOFLIGHTSVIEW` in the window and setup the navigation. Go to the window of your component.

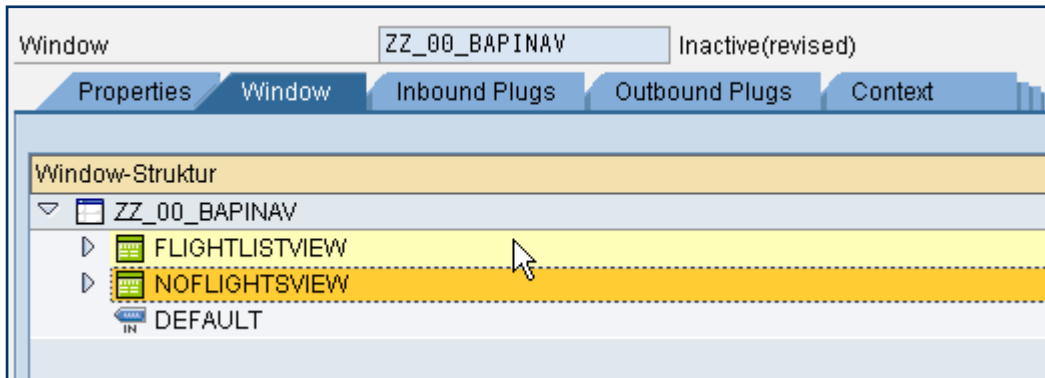


2. Open the context menu and select *Embed View*.



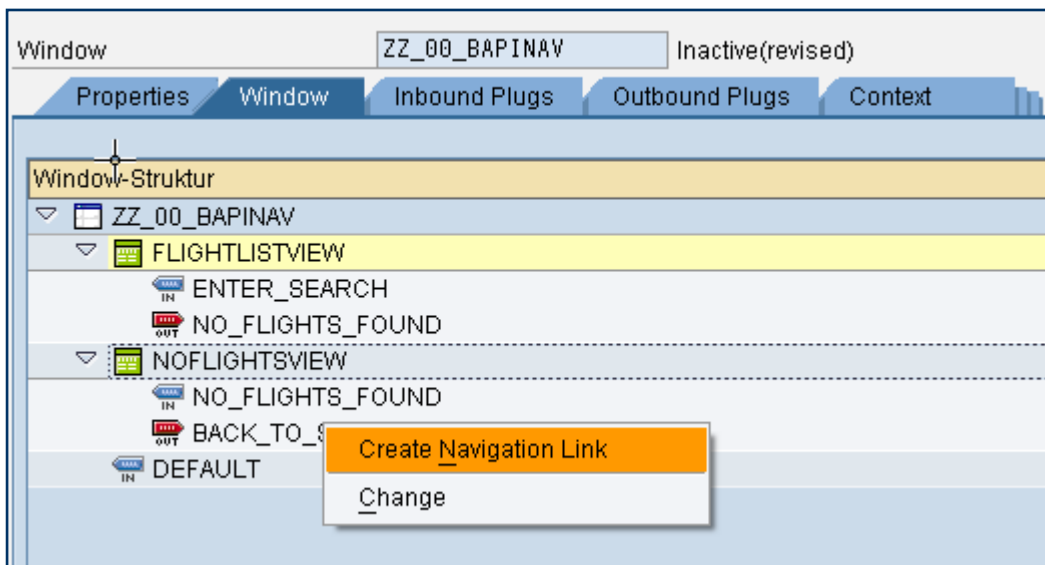
3. Click on the search help icon and select the view `NOFLIGHTSVIEW` and component `ZZ_00_BAPINAV`. Click on the *Continue (Enter)* button.

The new view is now embedded in the window.



4. For creating the navigation open both views.

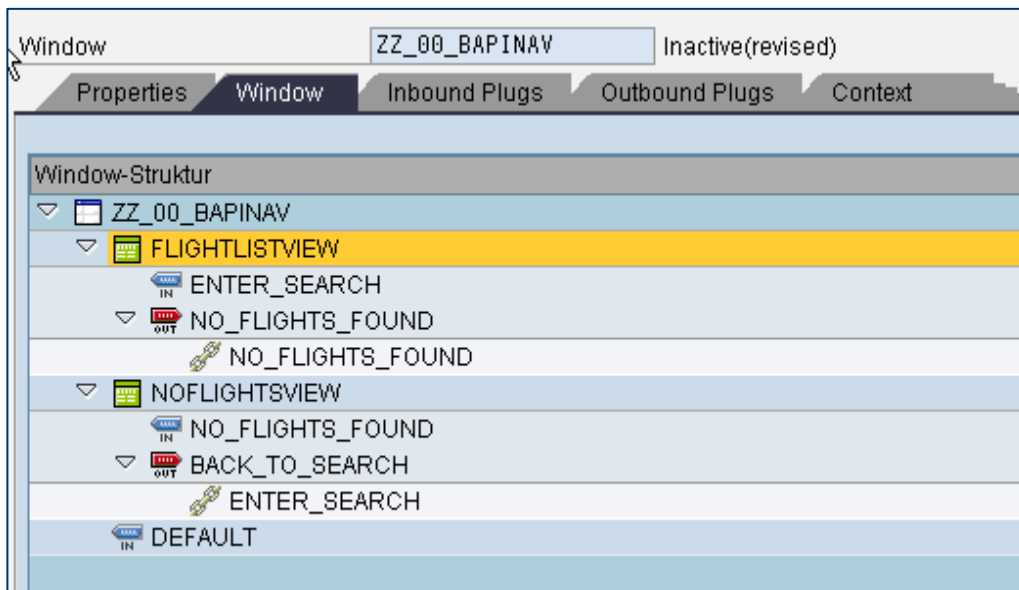
Select the outbound plug of NOFLIGHTSVIEW and open the context menu. Select *Create Navigation Link*.



A dialog popup comes up where you can enter the destination of the navigation.

Web Dynpro: Choose Destination for Navigation	
Start Component	ZZ_00_BAPINAV
Start View	NOFLIGHTSVIEW
Outbound Plug	BACK_TO_SEARCH
Dest. View	FLIGHTLISTVIEW
Embedding Position	ZZ_00_BAPINAV
Inbound Plug	ENTER_SEARCH
<input type="checkbox"/> <input type="checkbox"/>	

- For the outbound plug of FLIGHTLISTVIEW proceed in the same way with destination NO\_FLIGHTS\_FOUND. After finishing the navigation should look like the next screenshot.

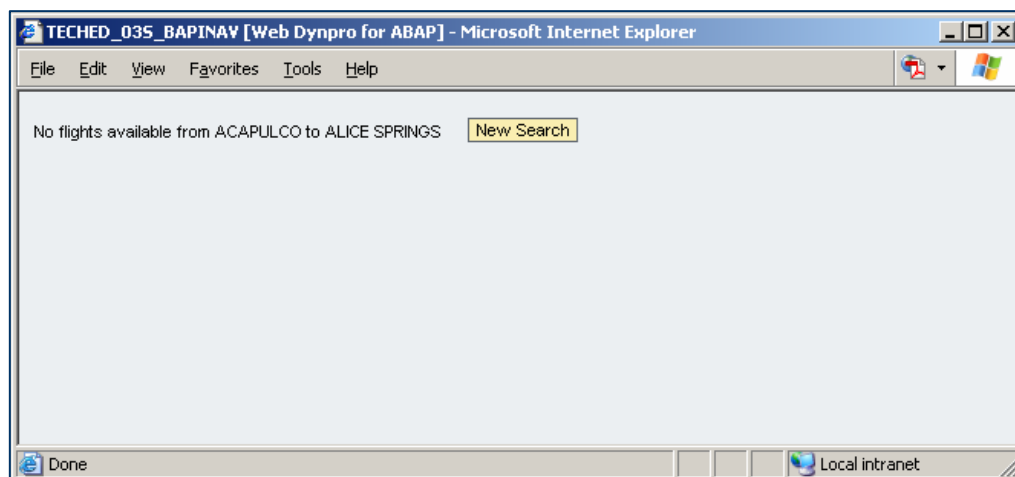


## Activation, Creation of a Web Dynpro Application and Execution

- Activate **all objects** of Web Dynpro component *ZZ\_00\_BAPINAV*.
- Create the Web Dynpro application *ZZ\_00\_BAPINAV* and assign it to package \$TMP (local object).
- Run your application.



If you choose exotic destinations like ACAPULCO and ALIC SPRINGS for which there are no flights in the database stored, then application will navigate to the new view.



## Result

As a result of this exercise, you have learned how to create a Web Dynpro component with two views and the navigation in-between them.

## SAP Online Help

More information on Web Dynpro for ABAP can be found at the SAP Help Portal under the short link

[http://help.sap.com/saphelp\\_nw04s/helpdata/en/77/3545415ea6f523e10000000a155106/frameiset.htm](http://help.sap.com/saphelp_nw04s/helpdata/en/77/3545415ea6f523e10000000a155106/frameiset.htm) or via path [help.sap.com](http://help.sap.com) → Documentation → SAP NetWeaver → SAP NetWeaver 2004s → English → SAP NetWeaver Library → SAP NetWeaver by Key Capability → Application Platform by Key Capability → ABAP Technology → UI Technology → Web UI Technology → Web Dynpro for ABAP.