Freely Programmed Help- Web Dynpro

Applies to:

SAP ABAP Workbench that supports Web dynpro development. For more information, visit the Web Dynpro ABAP homepage.

Summary

In addition to the Dictionary Search Helps and OVS helps, Web dynpro ABAP framework allows us to create our own input help components to make way for providing solutions to complex business requirements. Here in this article, I have listed down 15 easy steps to create Freely Programmed Help.

Author: Gokul Narasimhalu

Company: Infosys

Created on: 5 October 2010.

Author Bio

Gokul has been working for Infosys as an ABAP developer . In his 2 years of work experience, he has been exposed to a variety of ABAP topics and the recent one being Web Dynpro for ABAP.

Table of Contents

Applies to:.... 1
Introduction.... 3
Freely Programmed Help In Web Dynpro.... 3
Overview:.... 3
Steps to create Free Help.... 4
15 Easy Steps to Create Freely Programmed Help.... 4
Phase I: Building a Free Value Help Component.... 4
Related Content.... 16
Disclaimer and Liability Notice.... 17

Introduction.

Web dynpro framework enables us to design our own Input Help components, thereby providing a greater flexibility to meet complex business requirements.

The following Input Help types are common in WD components:

Automatic

Dictionary Search Help

OVS-Object Value Selector.

Freely Programmed Help In Web Dynpro

Unlike the above-mentioned types, Freely Programmed Help requires additional development and maintenance efforts. Since this involves the development of a new WD component, we can formulate complex search queries using Select Options for the individual search fields.

This documents illustrates the development of one such Free Help component and also its usage in providing Input Helps to the fields present in the implementing WD components.

In this document, I have provided 15 easy steps to create and implement the Freely Programmed Help. Since our main purpose is to understand the concept, I have overly simplified the business logic that is shown in the examples.

But, remember that this type of Input Helps is only used to meet complex business requirements. So try and make use of dictionary and OVS helps as much as possible in your applications.

Before we look into the step-by-step guide to develop Freely Programmed Help, here is the overview.

Overview:

(1) Creating an input help component and implementing the IWD_VALUE_HELP component interface.

(2) Creating a component usage for the input help component in your application component.

(3) Attaching the component usage to the context attribute.

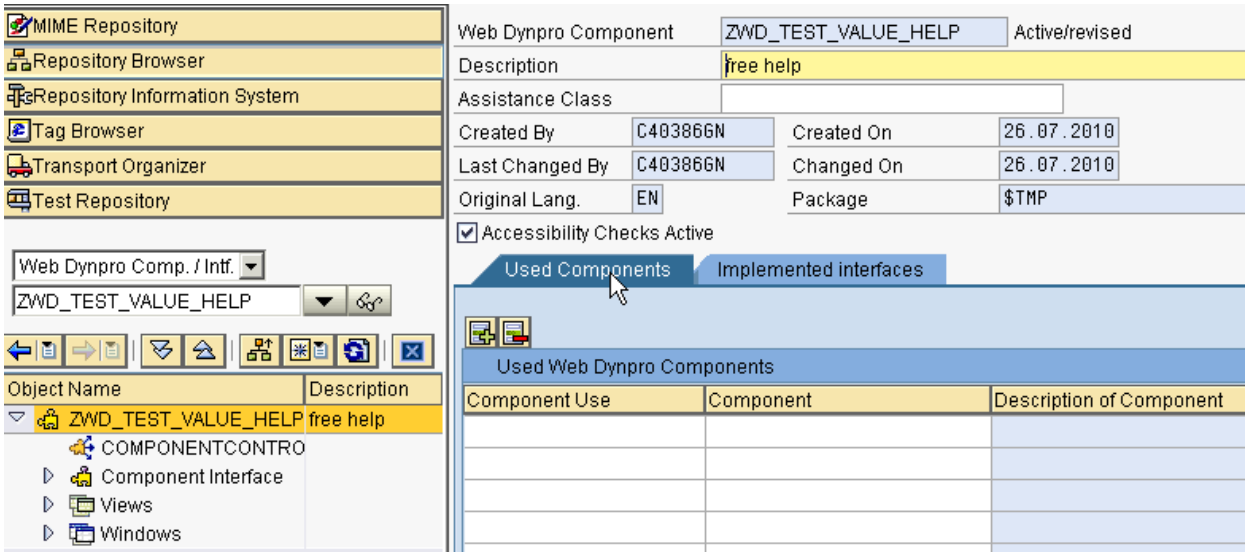
Steps to create Free Help

15 Easy Steps to Create Freely Programmed Help

Phase I: Building a Free Value Help Component.

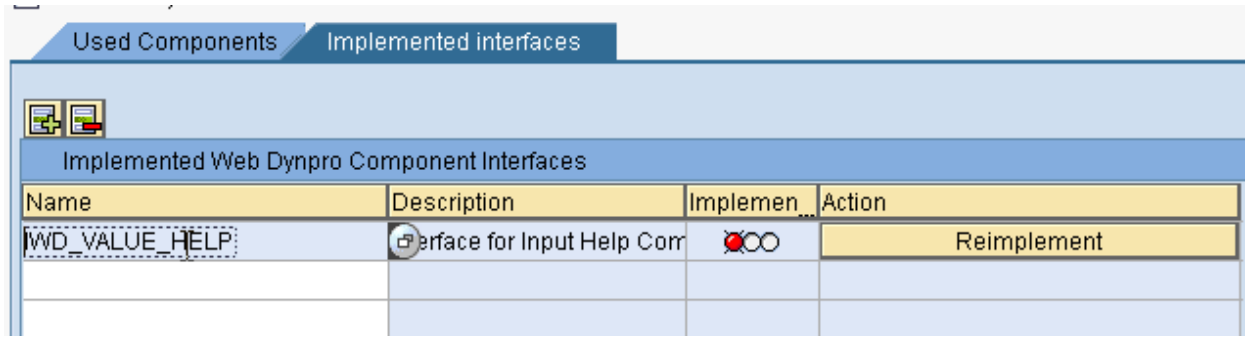
Step 1:

Create a Free Value Help component in *ABAP Development Workbench*.

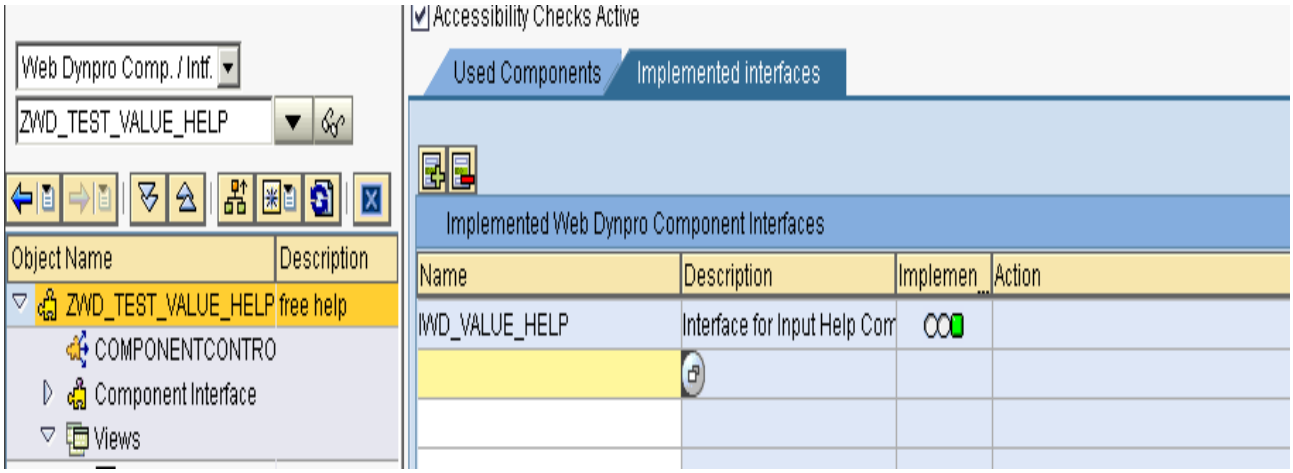


Step 2:

GOTO *Implemented Interface* tab and implement the interface IWD_VALUE_HELP. This interface will provide the methods and events necessary to implement Free Value Help.

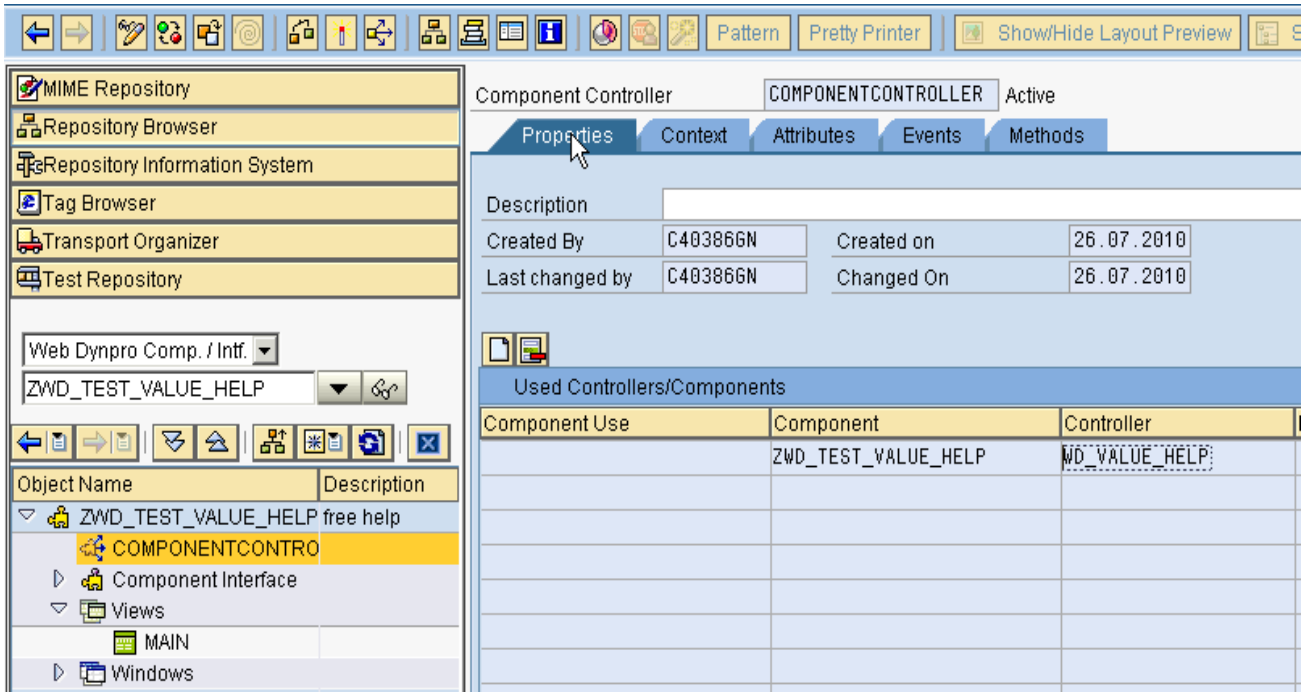


Click on the *Reimplement* button.



Step 3:

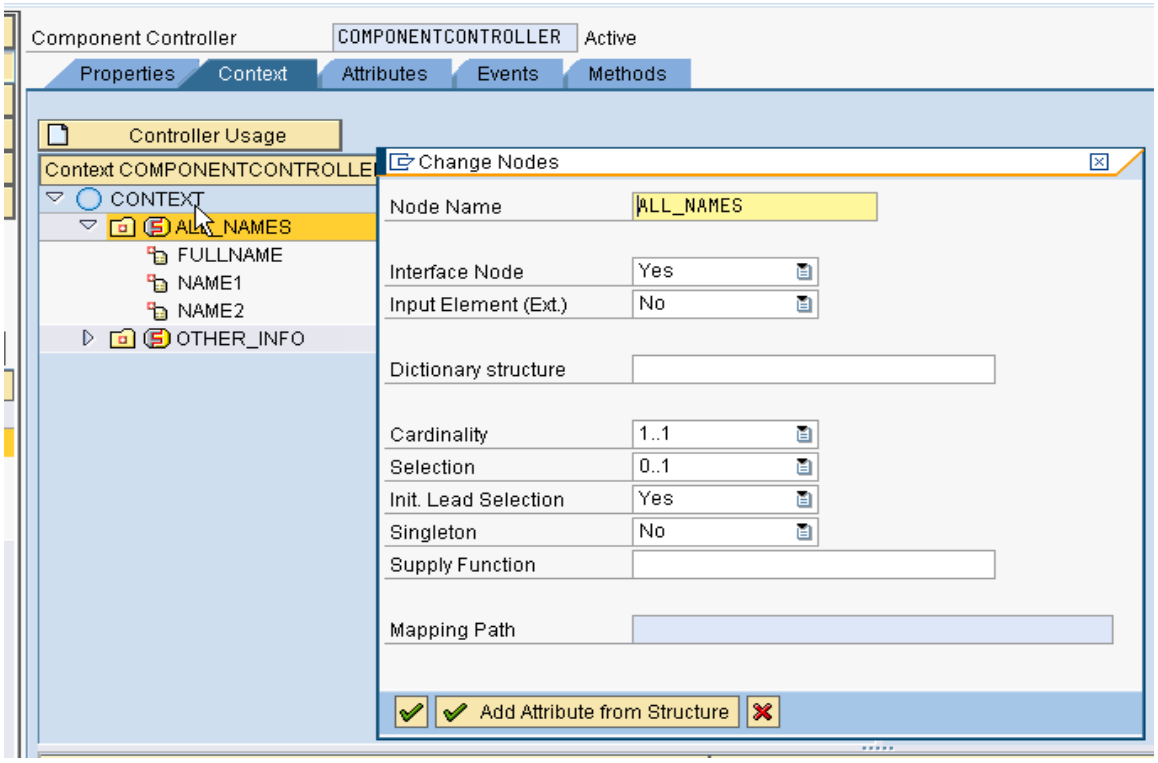
GOTO the *Component Controller* in the *Properties* tab declare WD_VALUE_HELP as a used controller.



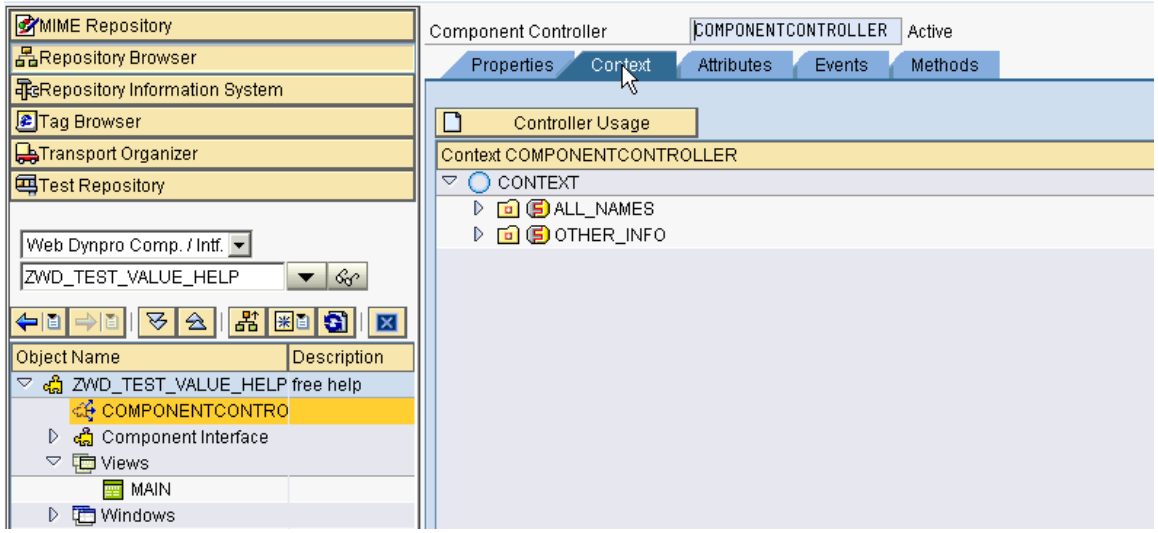
Click on the *create controller usage* button to select from the list.

Step 4:

In the *context* tab model your data as per the requirement. Make sure you use *Interface node* for those attributes that you would want to share across the components that implements this Free Value Help component.



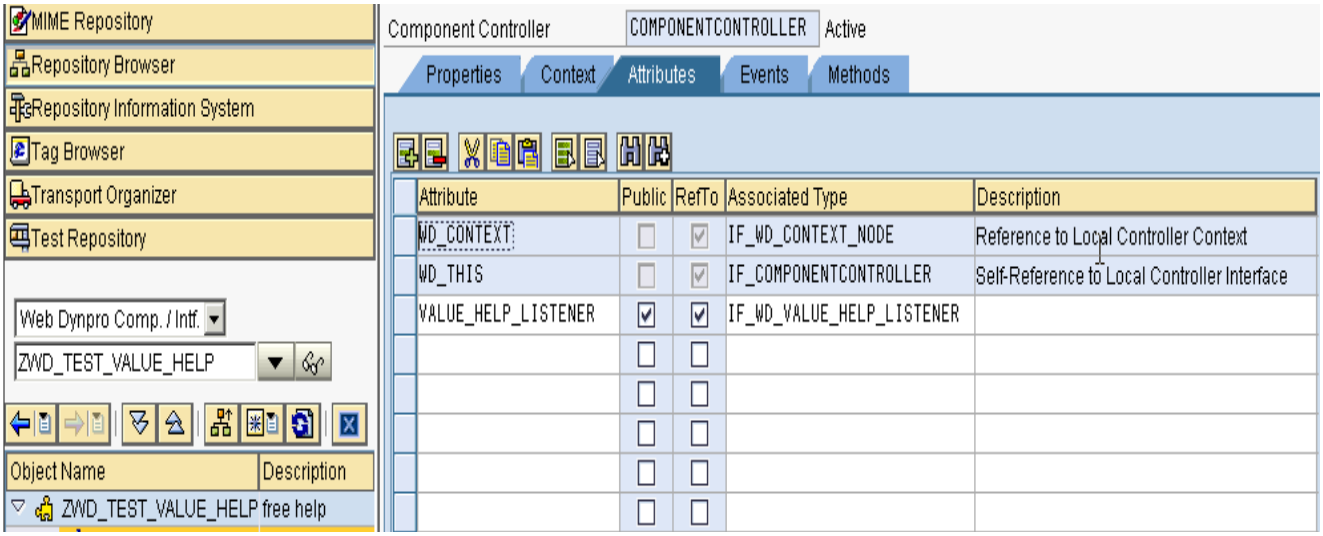
Say *Interface node* as yes, to make it an Interface Node.



Here, both the nodes are Interface nodes.

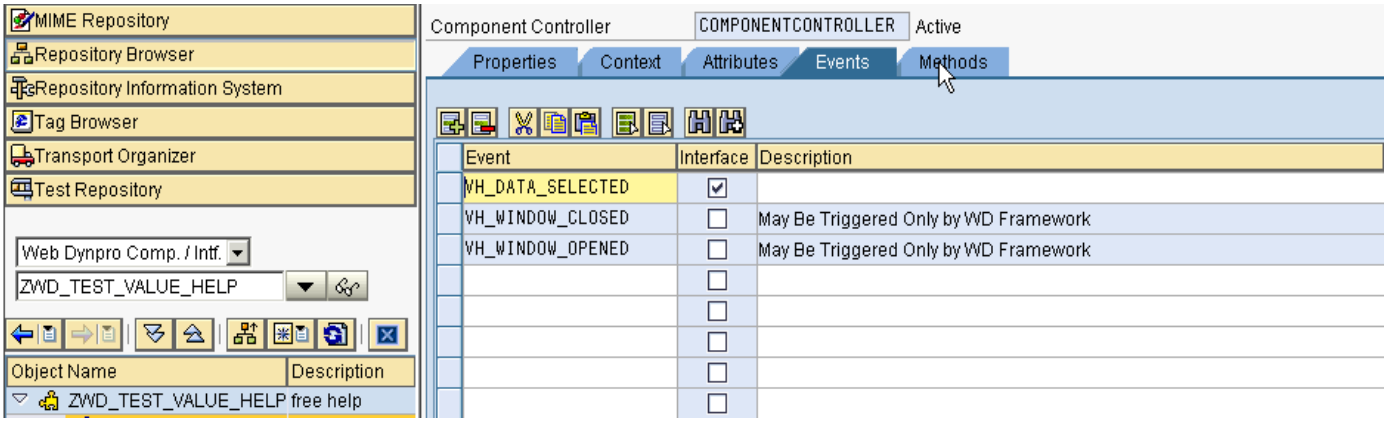
Step 5:

In the *Attributes* tab add a variable as a reference to the interface IF_WD_VALUE_HELP_LISTENER.



Step 6:

In the *Events* tab add an event and check the interface check box to make it visible across the components that implements this Free Value Help component.



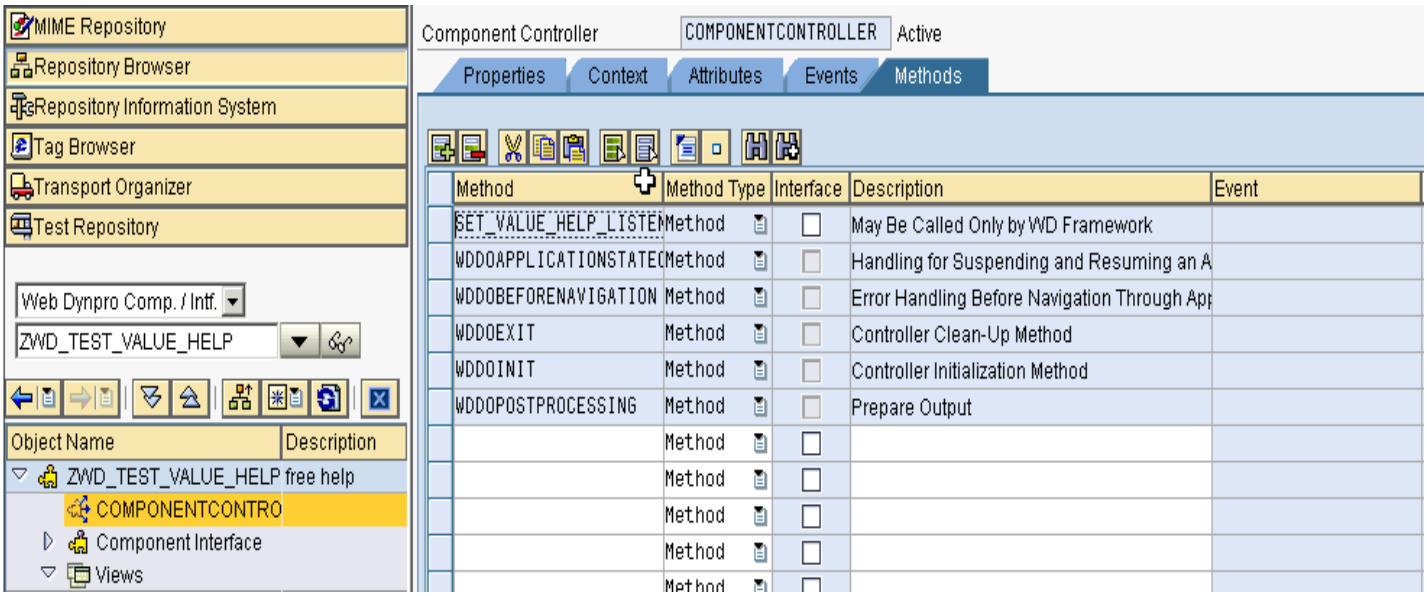
Here, VH_DATA_SELECTED is the interface event added.

The framework may only trigger the VH_WINDOW_CLOSED and VH_WINDOW_OPENED events of the component interface. The events indicate the points in time when the input help window is closed or opened.

A suitable event handler will be implemented in the component that makes use of this Free Help component.

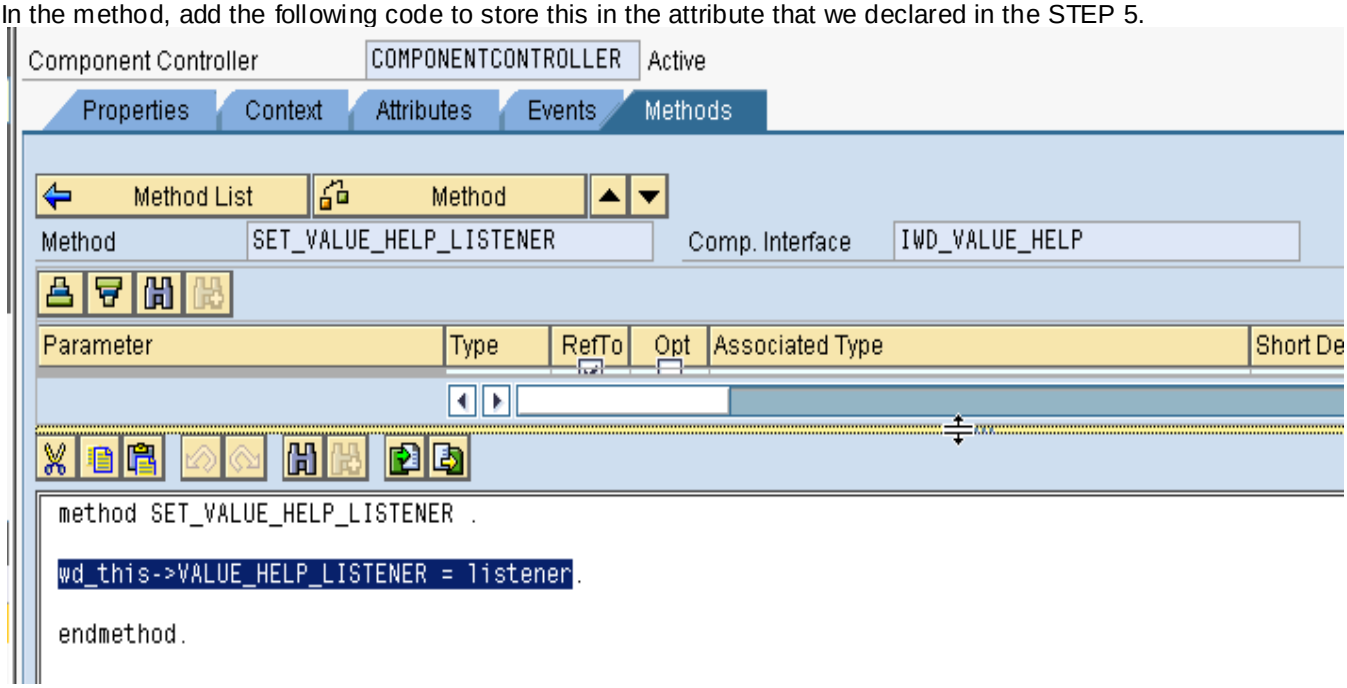
Step 7:

In the methods we can see a method named SET_VALUE_HELP_LISTENER. The Web dynpro framework only calls this method.



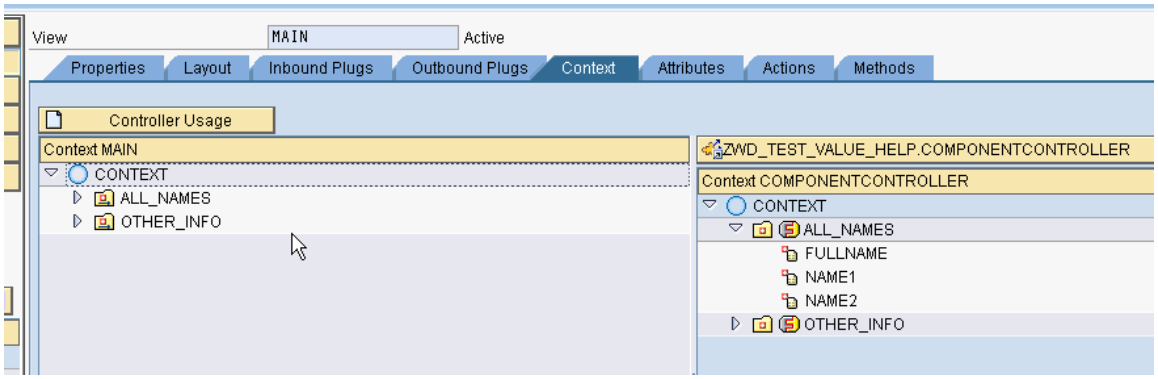
When this method is called, an instance of the interface IF_WD_VALUE_HELP_LISTENER is obtained as an *importing parameter*.

In the method, add the following code to store this in the attribute that we declared in the STEP 5.



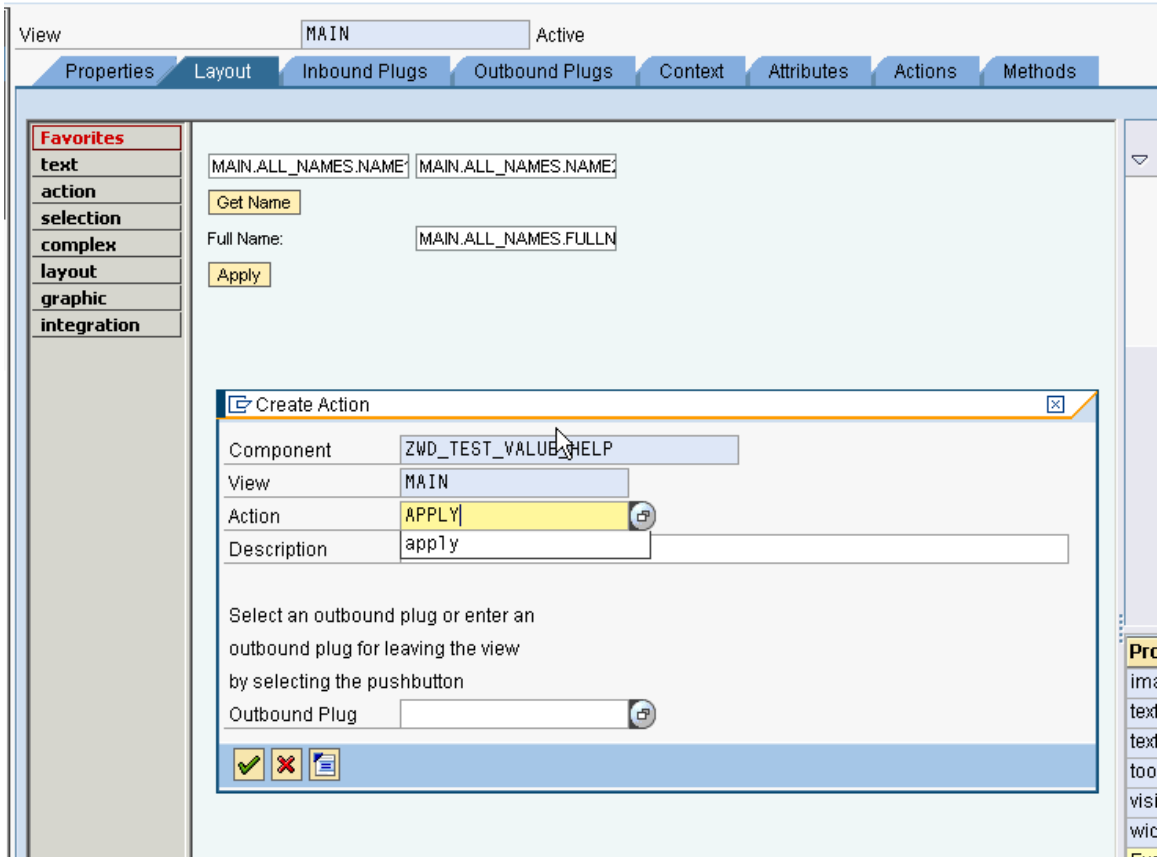
Step 8:

Navigate to the MAIN *view context*. Drag and drop the interface nodes in the view context. Then go ahead and design the *Layout*. The view layout that we design here appears as a popup when the F4 help is rendered.



Step 9:

Have a button in the layout, say APPLY, that would finally close the F4 help window and transfer the desired values back to the Parent Component.

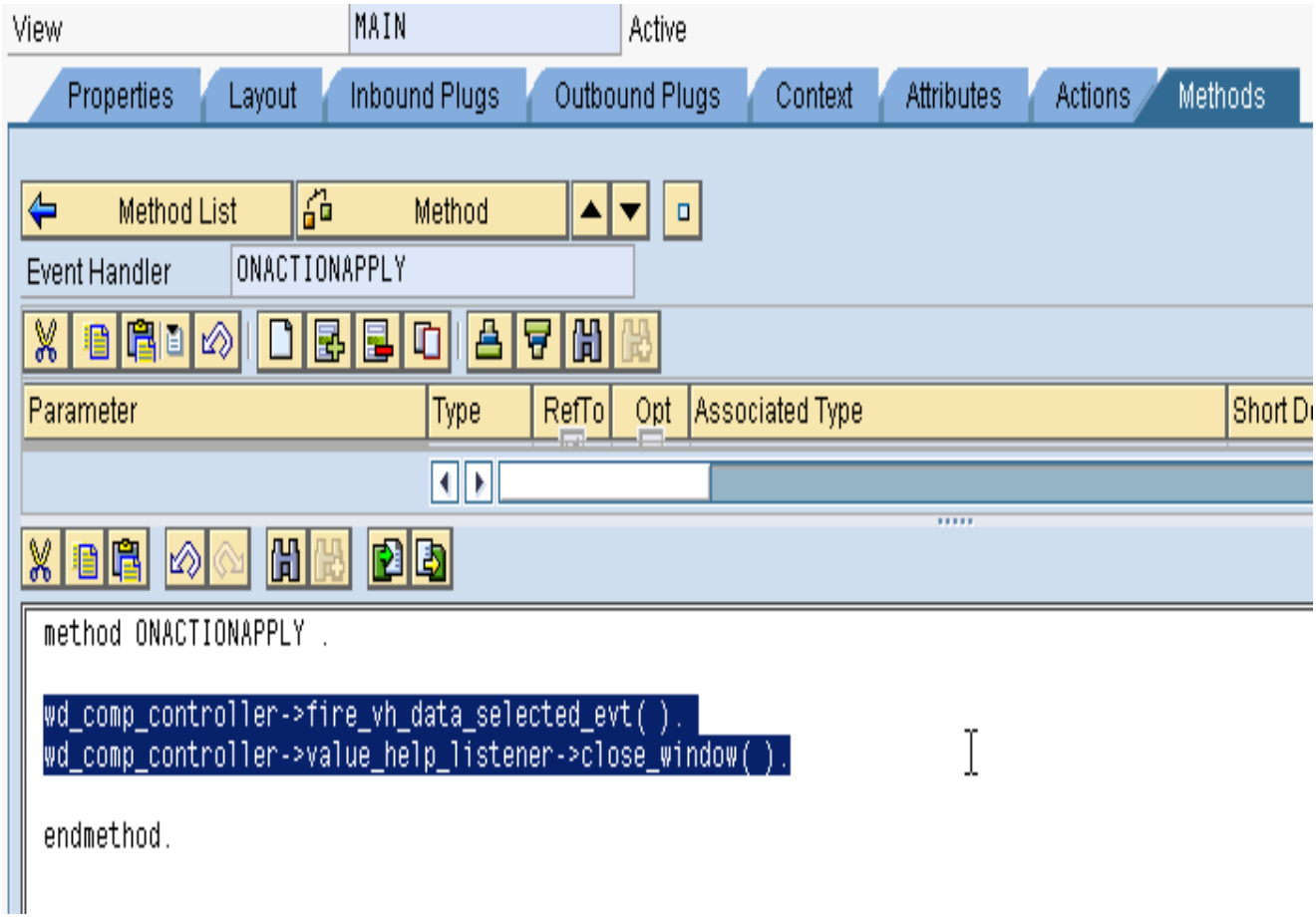


Step 10:

Do the following in the APPLY button:

→ Fire the event that we declared in STEP 6.

→ Close the F4 help window.



The interface IF_WD_VALUE_HELP_LISTENER offers methods to close the window.

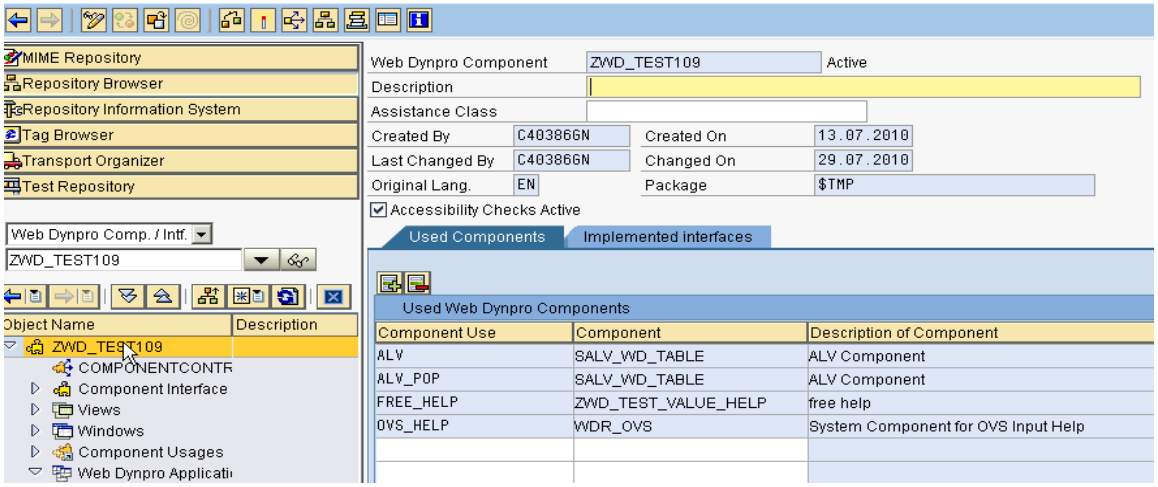
Note: Do not forget to add your MAIN view in the WD_VALUE_HELP window as this belongs to the interface view.

PHASE II: Building the parent component that implements this Free Value Help Component.

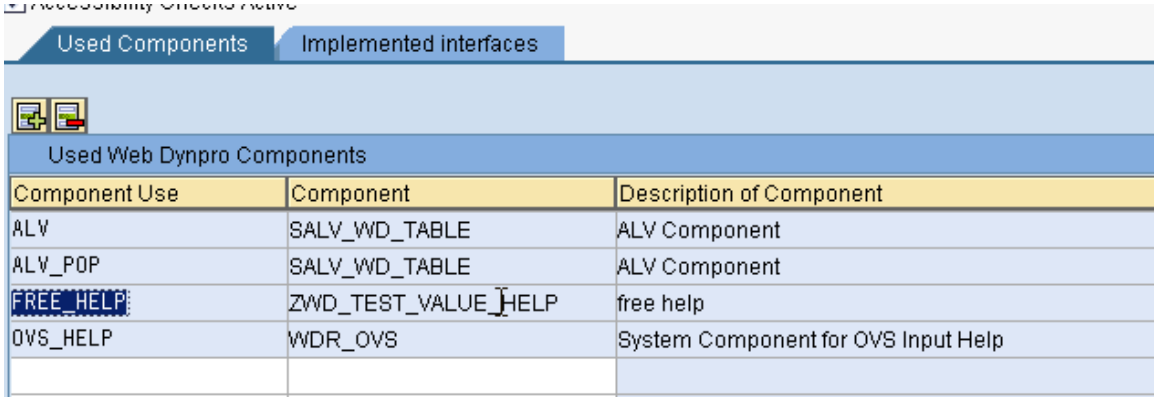
Step 11:

In the parent component, declare the already built Free Help Component as a *Used Component* in the *Used Components* tab. Navigate to the Component Controller, in the *Properties* tab create a controller usage for the already built Free Help component.

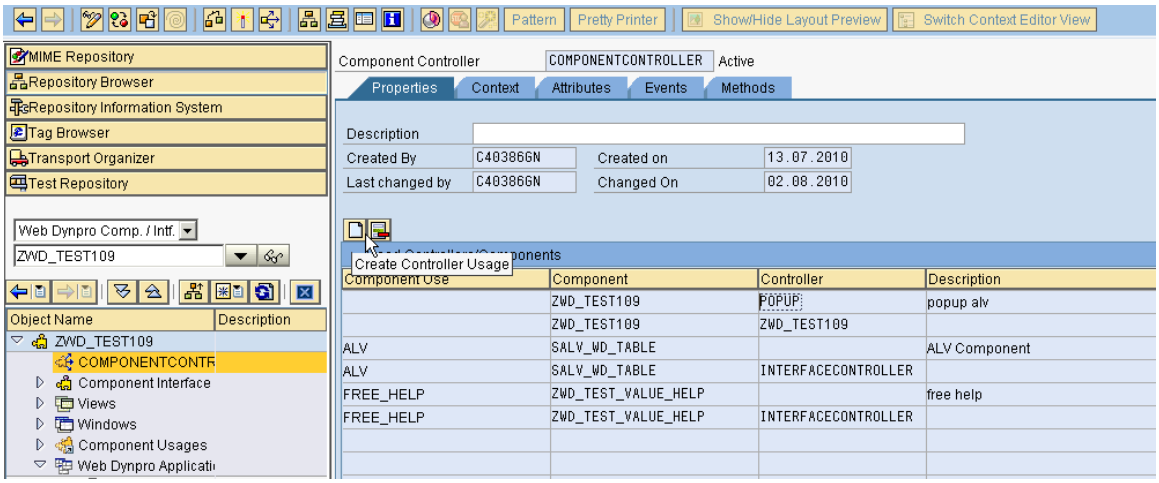
Follow the below screenshots.



Parent component implementing Free Help component.



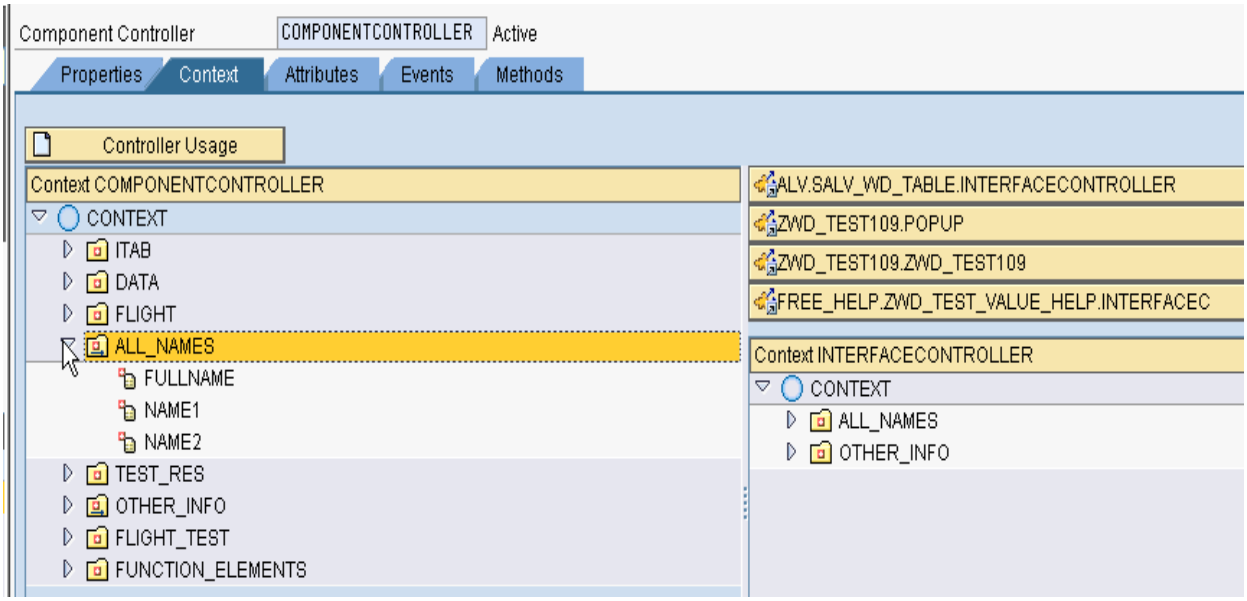
Add the component built for Free Input Help.



Use create controller usage button to select from the available list.

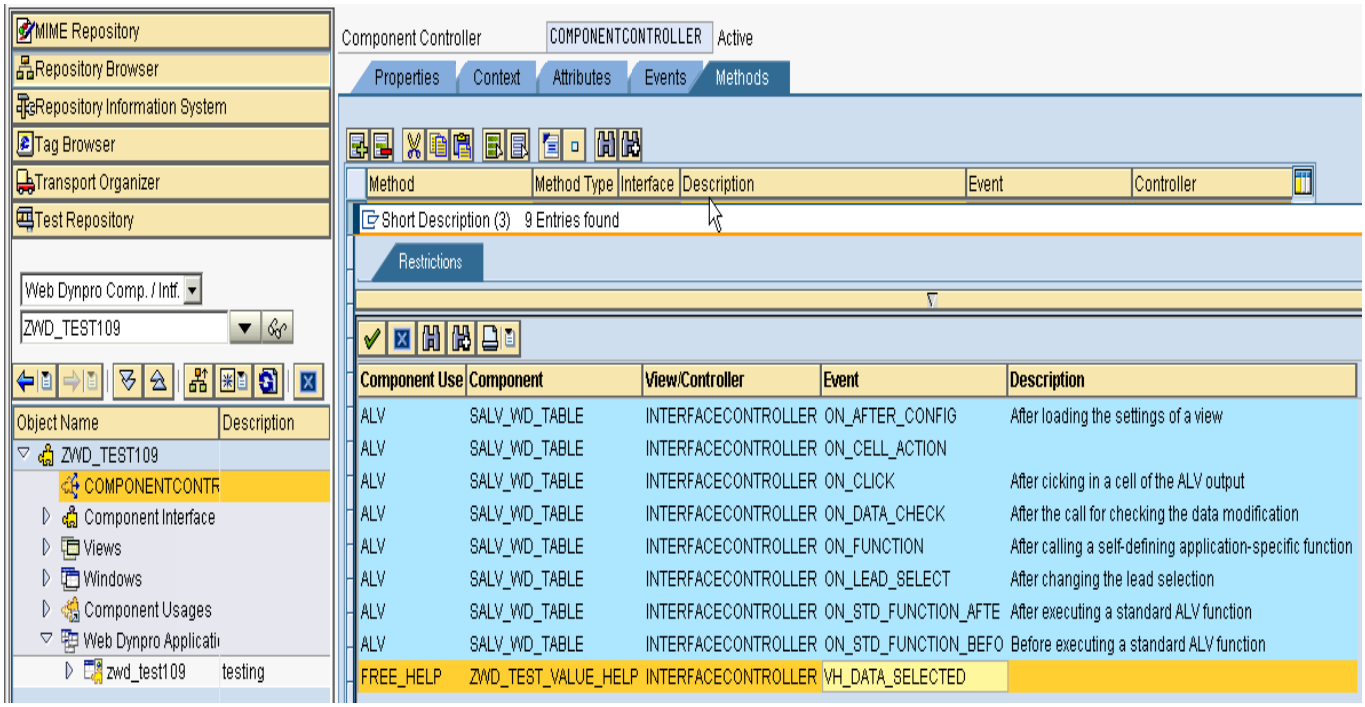
Step 12:

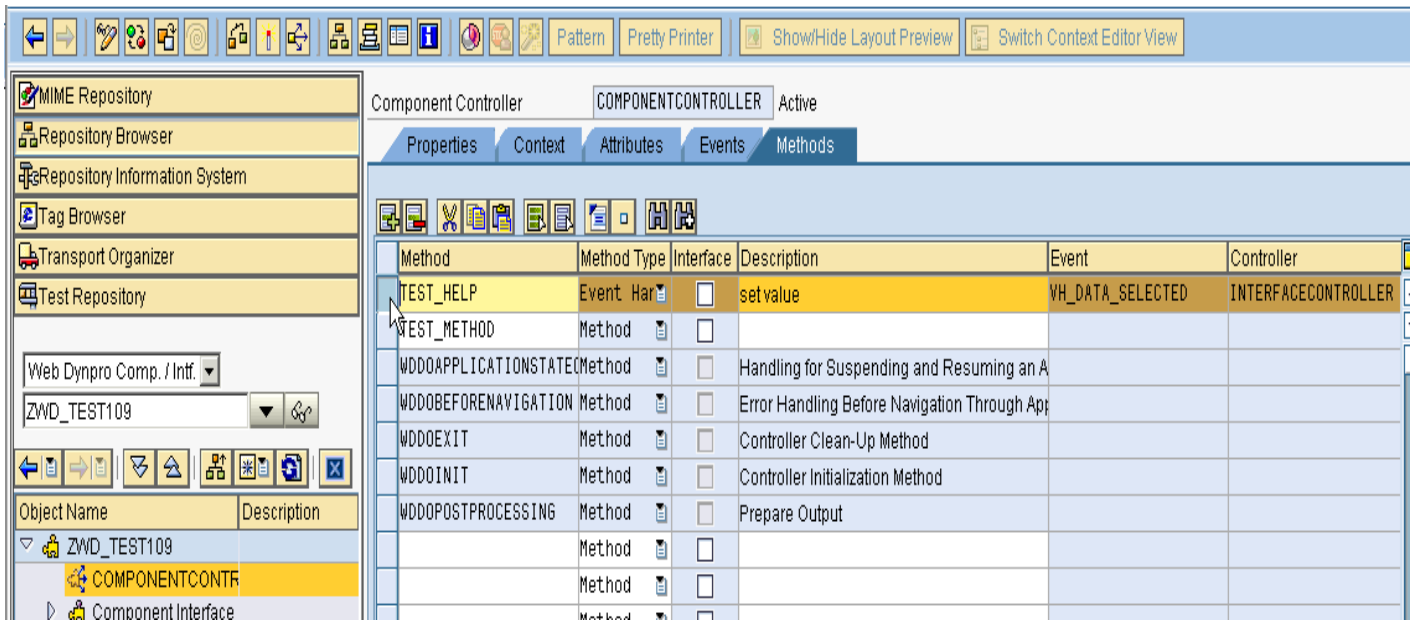
In the *Context* tab of the Component Controller you can see those nodes from the Free Help component that were declared as a interface nodes. Drag and Drop those nodes into this Component Controller context to facilitate the data transfer between the two WD components.



Step 13:

In the methods tab of the component controller create a suitable *event handler* for the interface event that we created in the Free Help component in the STEP 6.

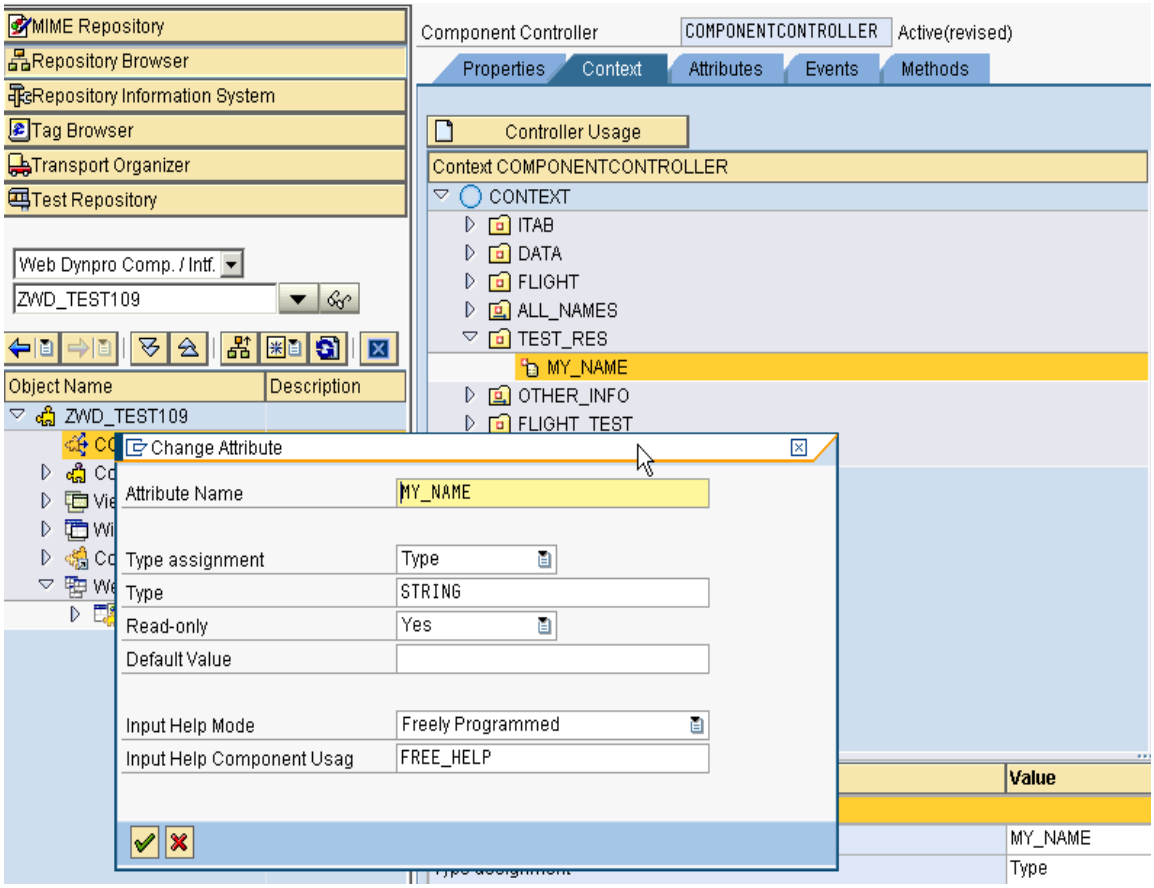




Make use of F4 help to select the suitable event.

Step 14:

In the component controller context, where you have modeled the data, attach this Free Help component to the attribute that requires the Freely Programmed Help.



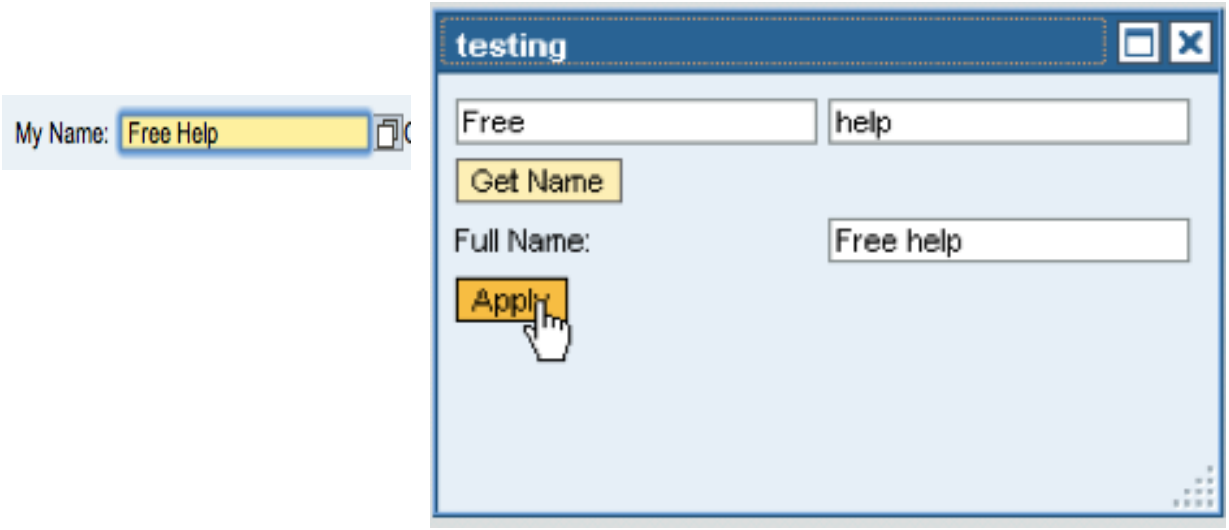
Select the Input Help Mode as Freely Programmed. Do F4 to select your Free Help Component.

Step 15:

Create *a view layout* for this parent component. In the layout, bind the *UI input field* element to the attribute that has Free Help component attached. Now you can see the F4 getting enabled for the UI element on the screen.

Create an application for this Parent Component and run the URL on your browser. You can see the Input Field with the F4 button enabled. On F4, the Free Help component appears as a popup.

Perform your business logic in the Free Help component. Once the data is available, click APPLY, this will close the F4 window and returns the desired value from the Free Help component to the Parent component.



Here, the “Full Name” value flows to the *My Name* field in the Parent Component.

Related Content

http://wiki.sdn.sap.com/wiki/display/WDABAP/Freely%2bprogrammed%2binput%2bhelp

For more information, visit the Web Dynpro ABAP homepage.

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.