# Populating a Tree with Inputs from an RFC Model

## Applies to:

SAP NetWeaver Web Dynpro for Java version NWDS 7.0.13. For more information, visit the User Interface Technology homepage.

## Summary

This article deals with populating a tree Ui element with data from the backend R/3 system using Rfcs.

**Author:** Abhilasha A Dahare.

**Company:** L & T Infotech.

**Created on:** 18 September 2008

## Author Bio

Abhilasha Dahare is working as a Web Dynpro consultant in L&T Infotech Pvt. Ltd. since 9 months ago.

## Table of Contents

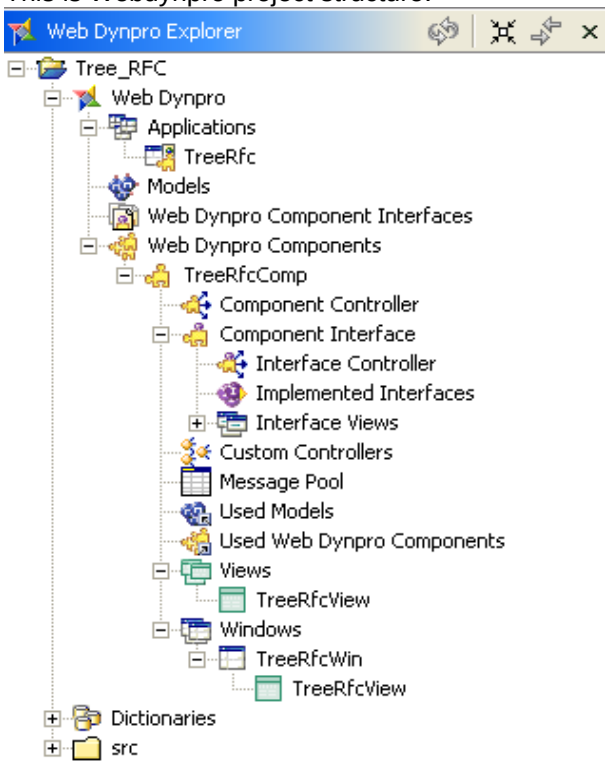
## Steps

### Creating your Web Dynpro Project

Create a new Web Dynpro project called Tree_RFC. In this project create an application called

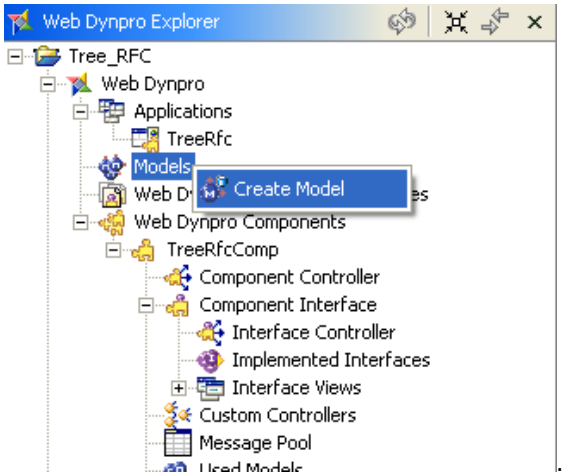
TreeRfc, a component called TreeRfcComp, a window called TreeRfcwin and a view called TreeRfcView. Assign them to a package called com.sap.ltin

This is Webdynpro project structure.

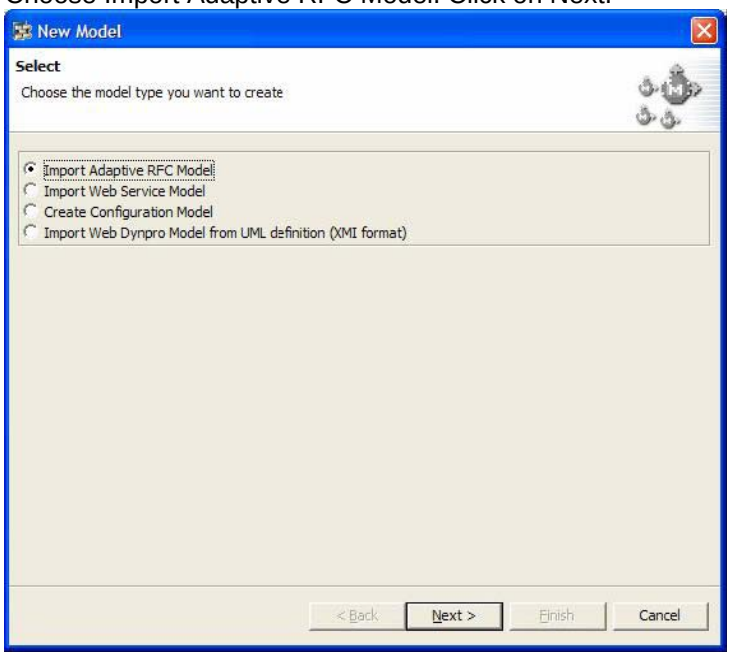


## Creating an RFC Model in Web Dynpro

1. Create an Adaptive RFC Model called FlightModel. To do it, right click on Model and select Create model.

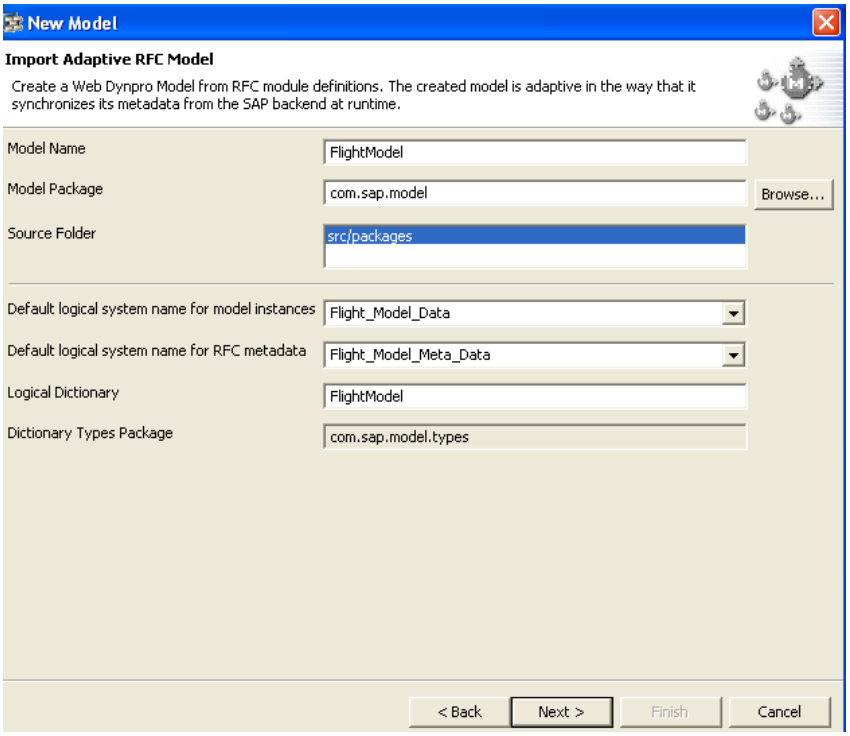



2. Choose Import Adaptive RFC Model. Click on Next.

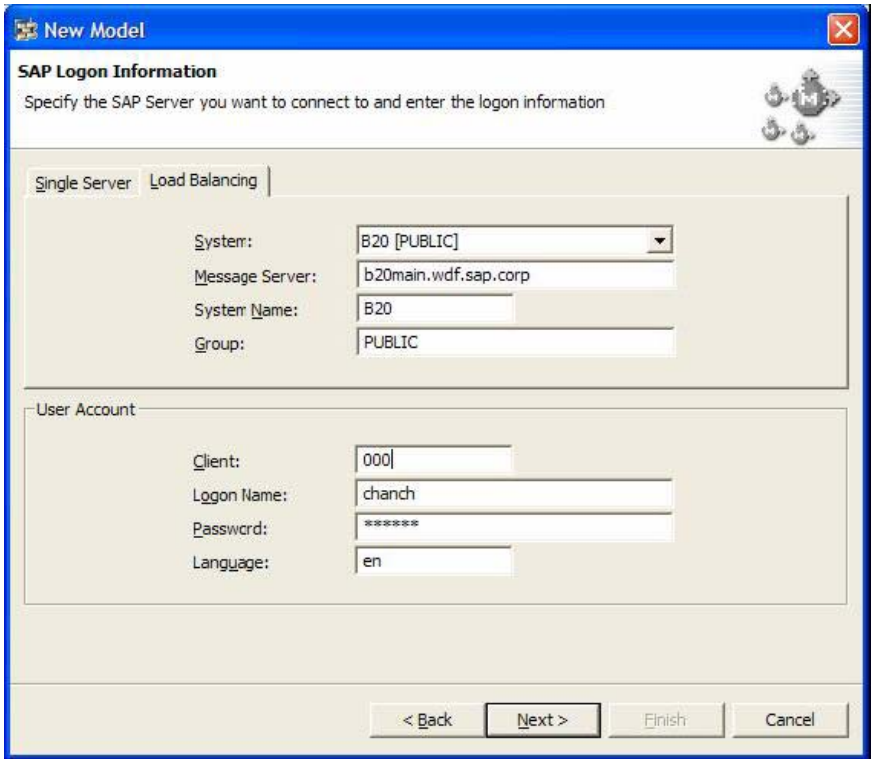



3. Enter the following for the model info:
   - Model Name: FlightModel
   - Model Package: com.sap.model
   - Default logical system name for model instances: Flight_Model_Data
   - Default logical system name for RFC metadata: Flight_Model_Meta_Data and click on next.

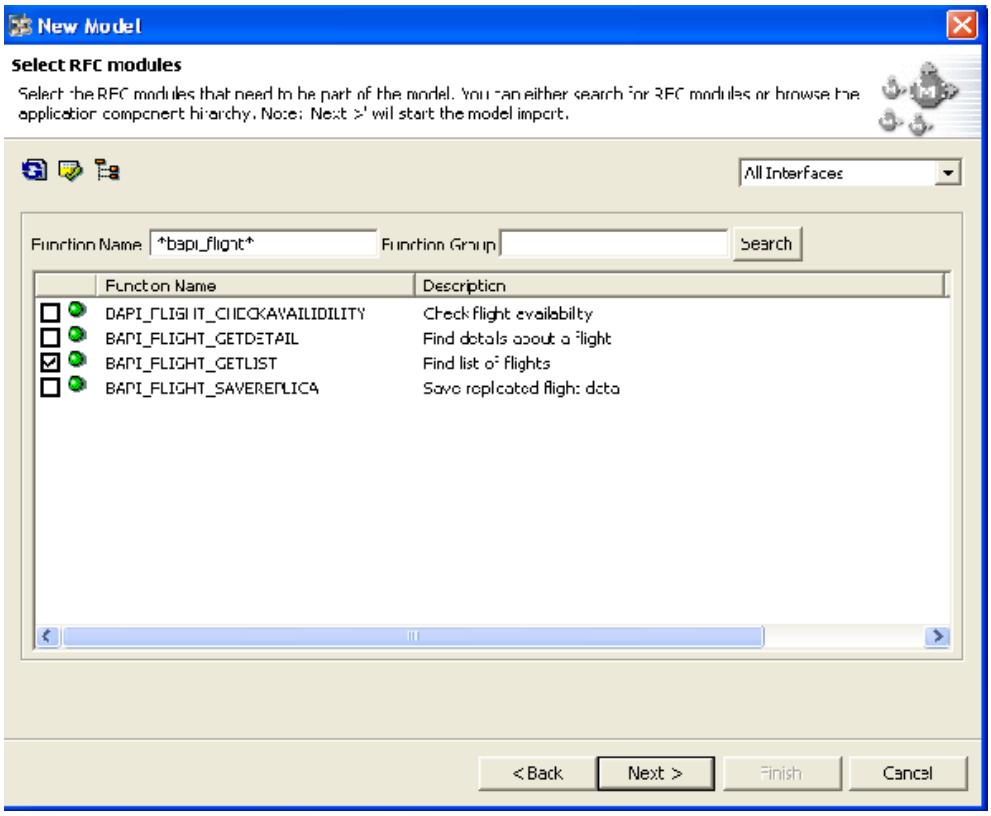
**Note:** Give the jco names as created on the server. The jco names may be different in your case.



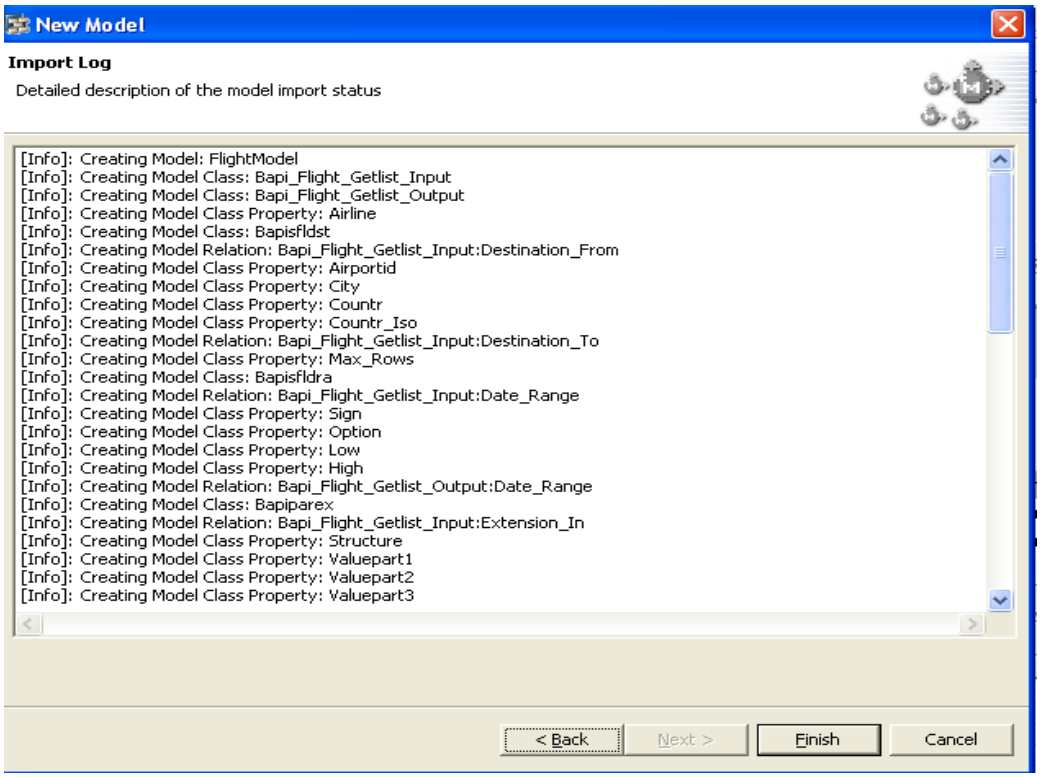

4. Under the Load Balancing tab, enter the logon info of the R/3 system the BAPI is located at (the system show up in the list only if you have it registered in SAP GUI). Click next.



5. Enter BAPI_FLIGHT_GETLIST and click on Search. On the returned function module below, select BAPI_FLIGHT_GETLIST and click on Next.

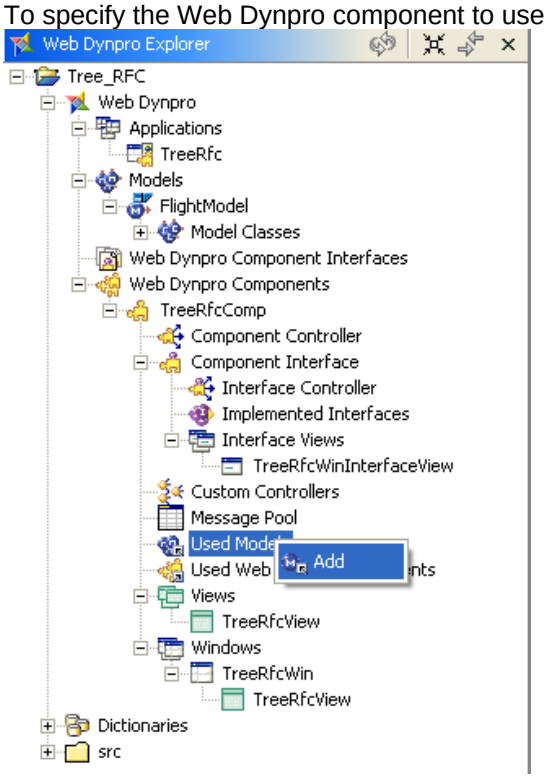



6. The model and the corresponding Java classes will be generated. Click on Finish.

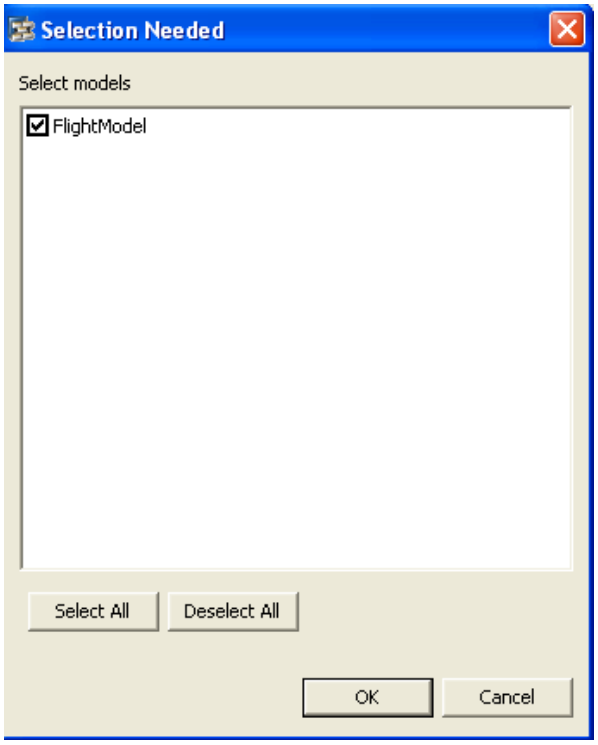


## Calling the RFC model

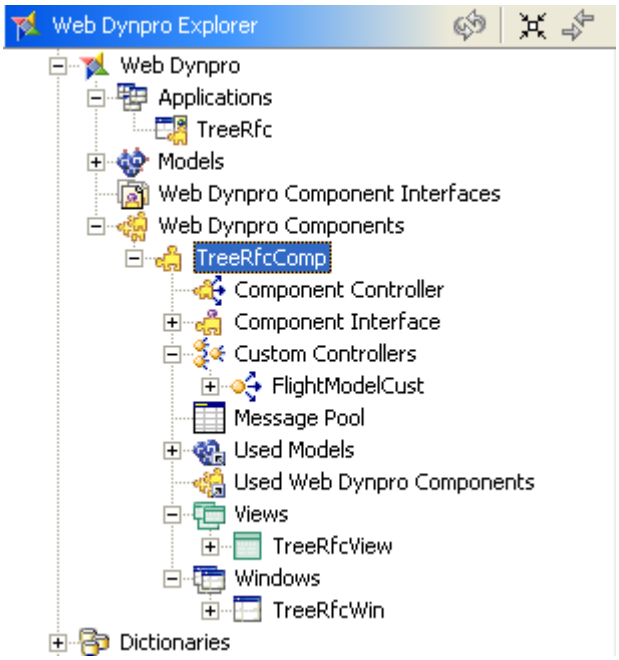
1. To specify the Web Dynpro component to use this model, right click on Used Model and choose Add.



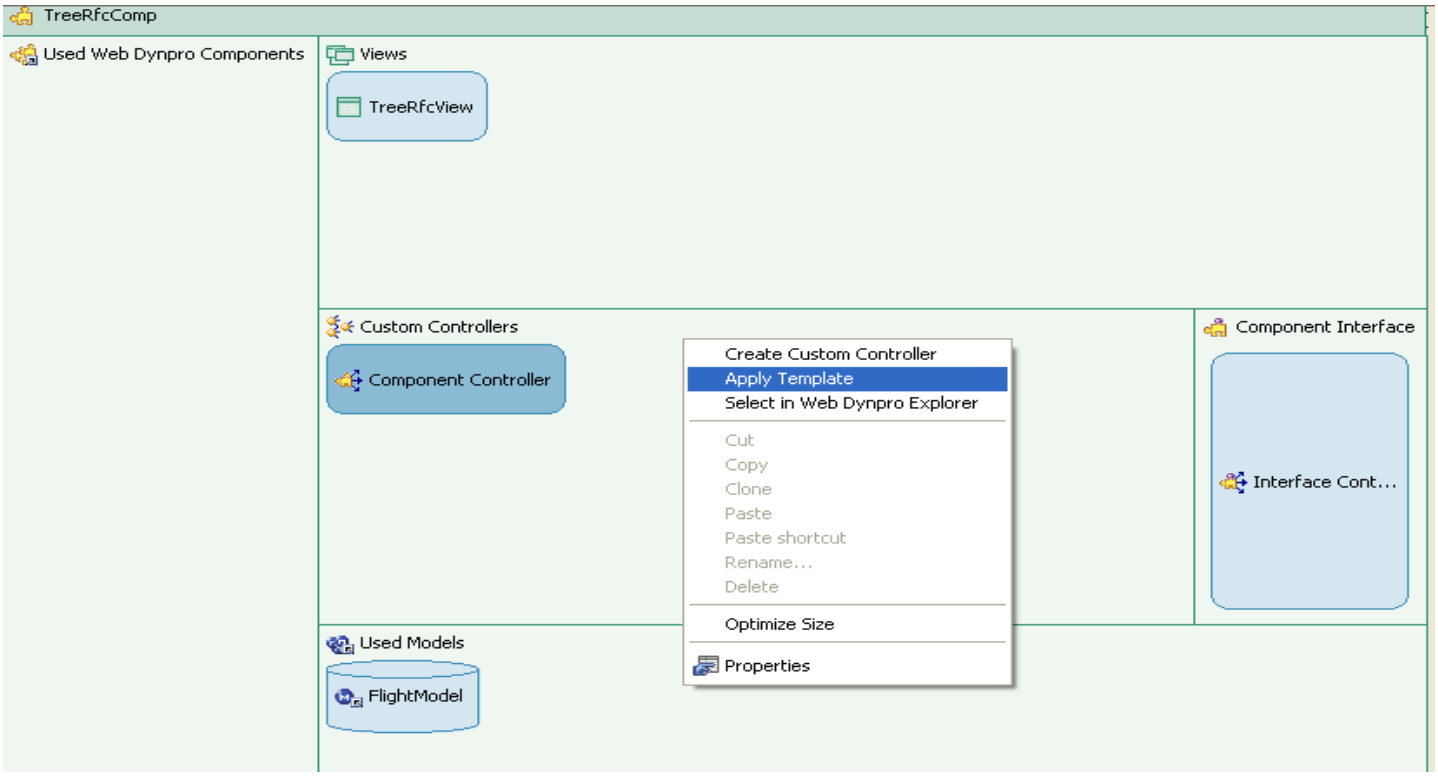

2. Choose FlightModel and click on OK.



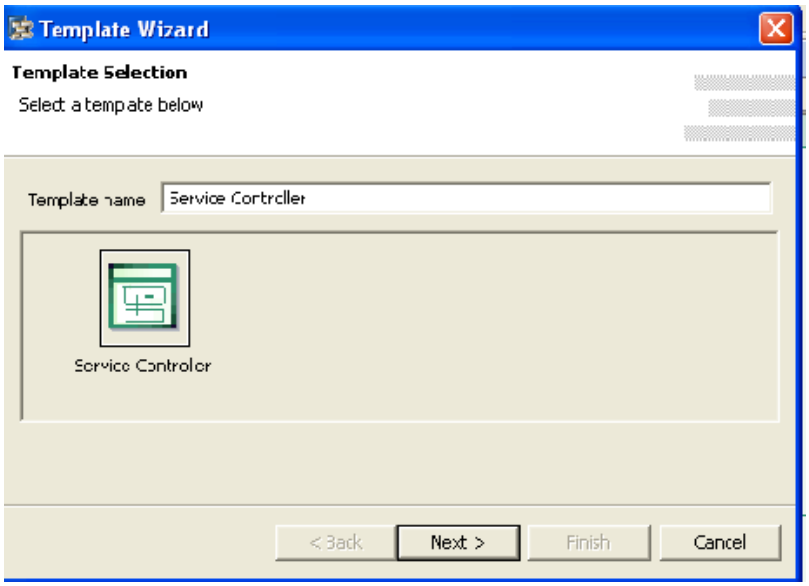
3. We now need to create a controller for executing the model. Open the Data Model of TreeRfcComp by double clicking on it.



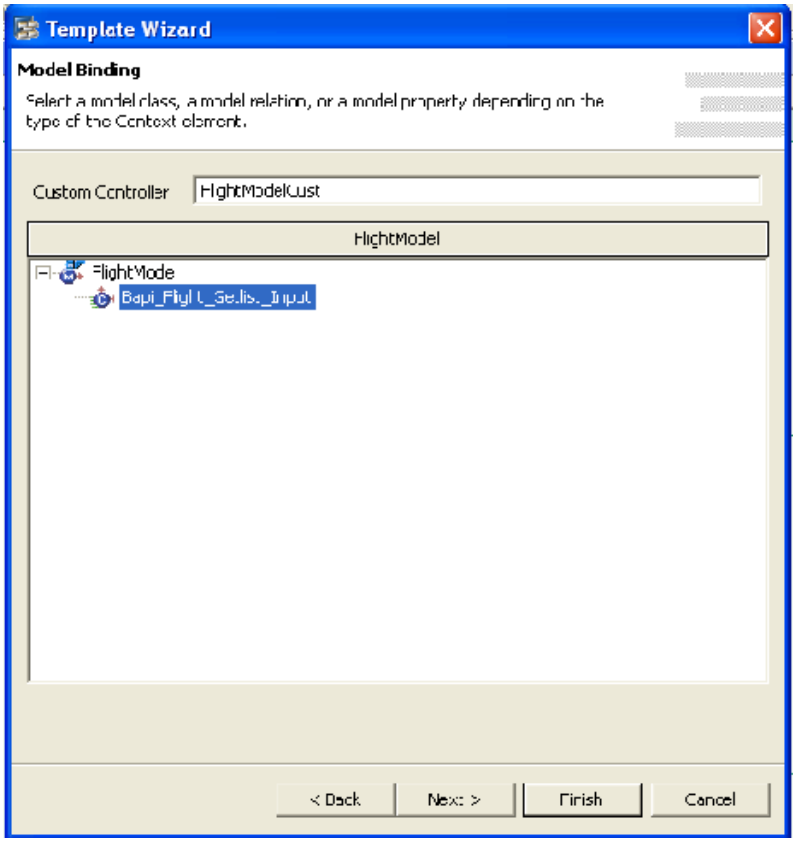

4. Right click on the area of Custom Controller and select Apply Template.



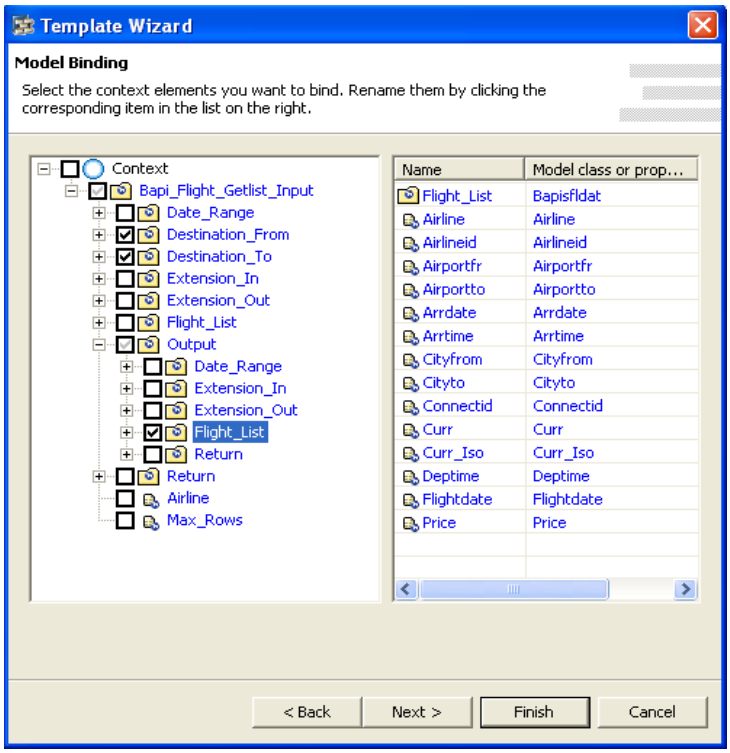
5. Choose Service Controller and click on Next.



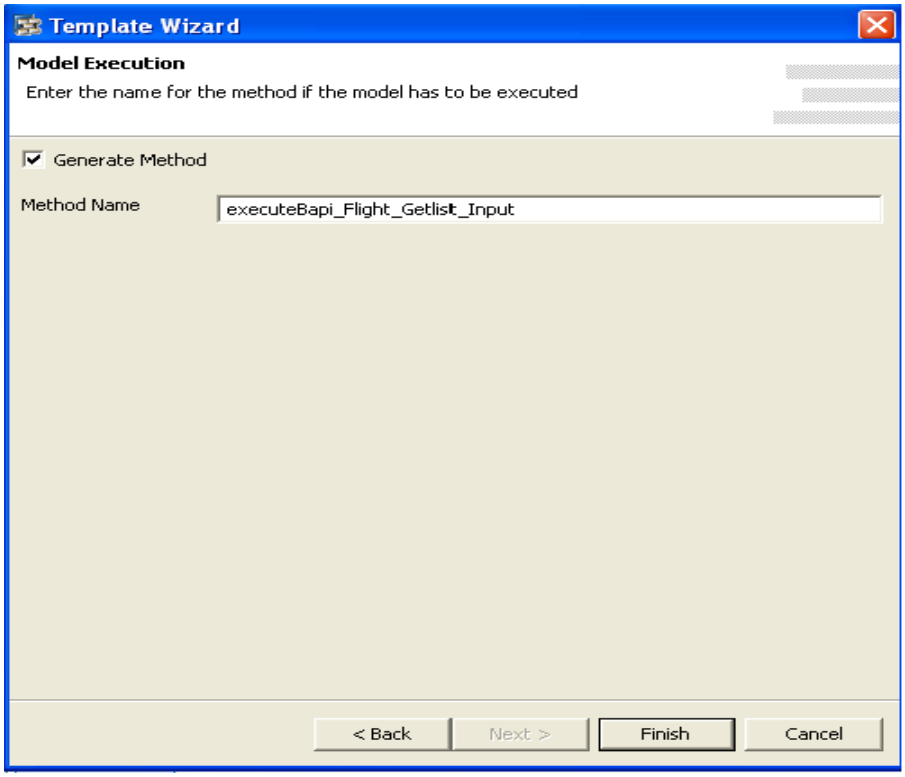

6. Select Bapi_Flight_GetList_Input and click on Next.



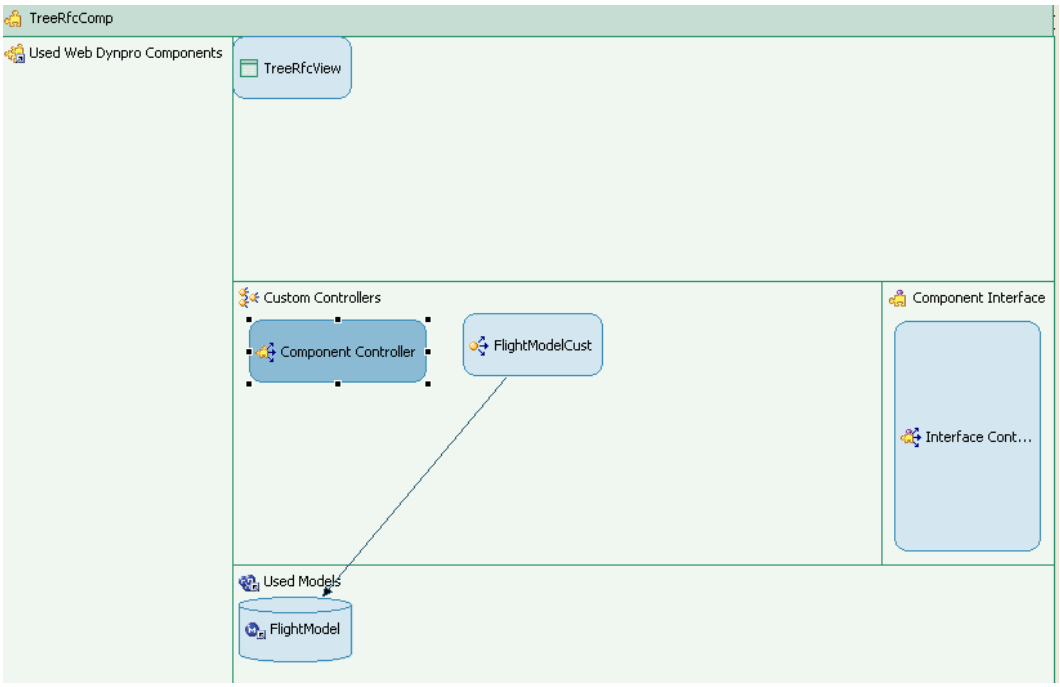
7. Select the following node Destination_From, Destination_To and Under Output Node select Flight_List. Click on Next.



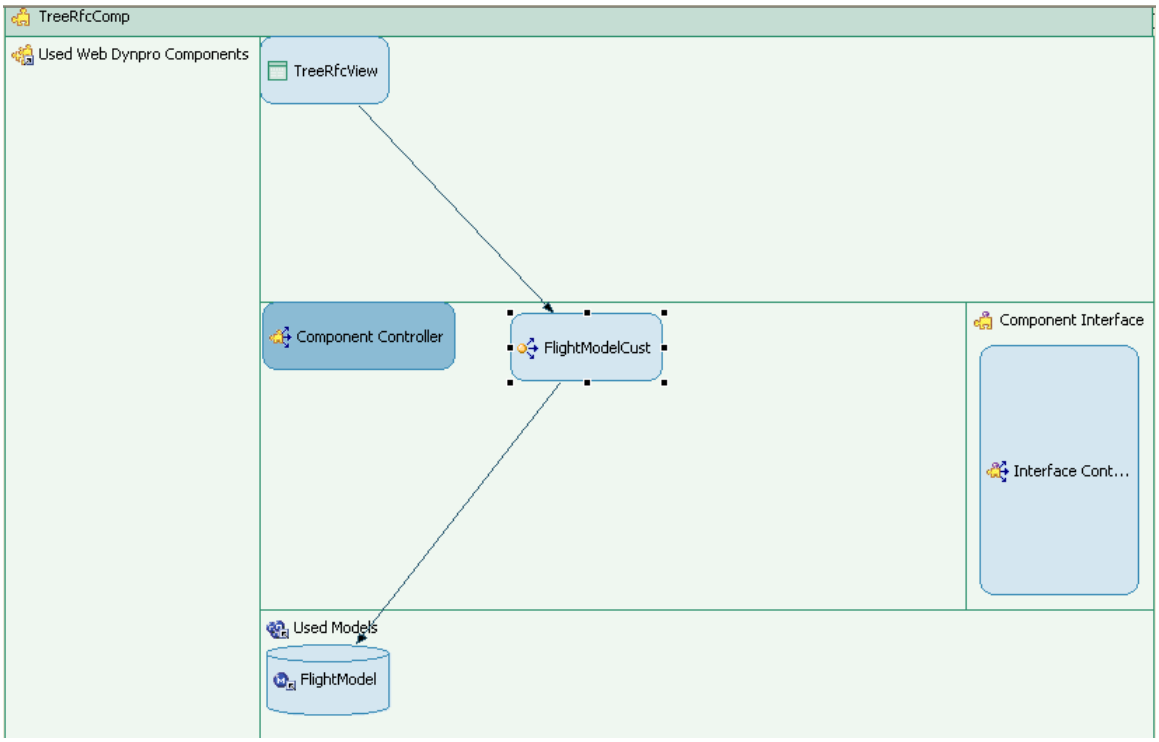

8. Click on Finish on the following screen to generate the method for executing the model.



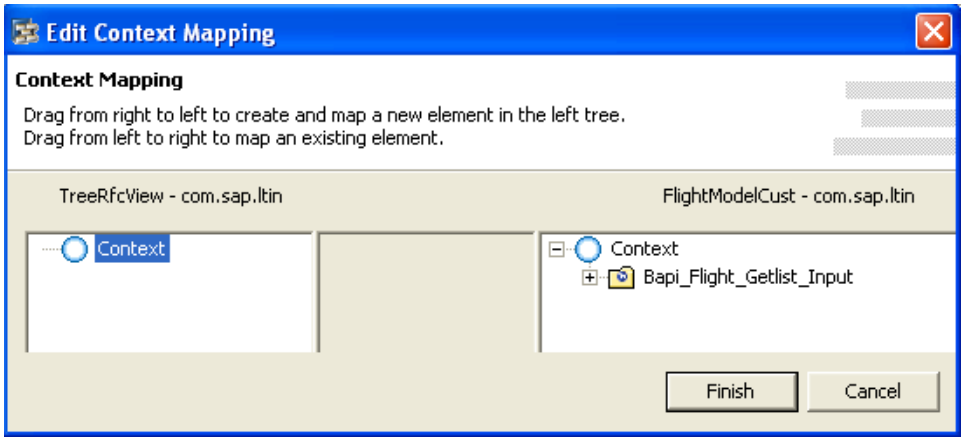
9. This will create the custom controller for executing the mode.



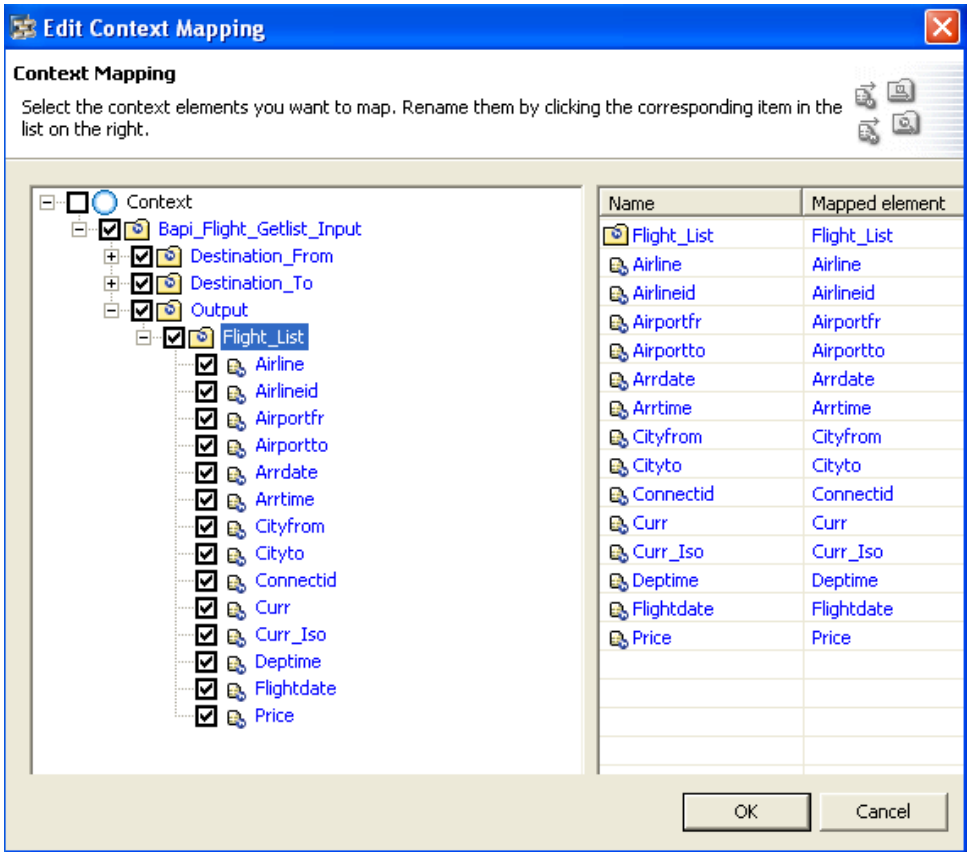

10. We need to specify the View to use this Custom Controller. To do it, draw a Data Link from TreeRfcView to FlightModelCustController.



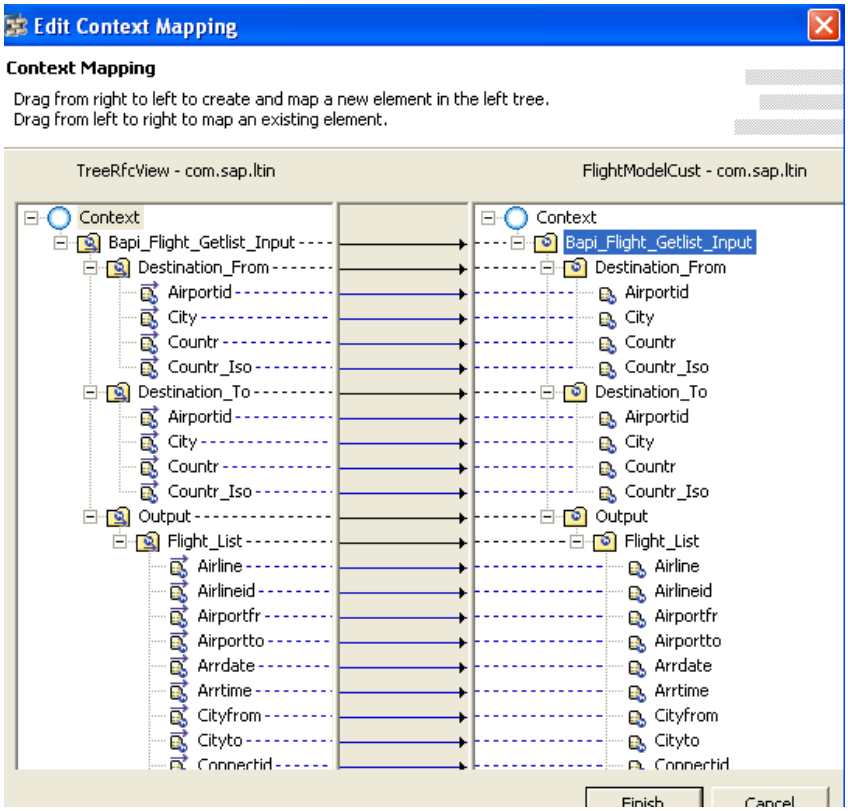
11. A dialog of Edit Context Mapping will pop up. Drag the Bapi_Flight_GetList_Input from the right column to the Context on the left column.



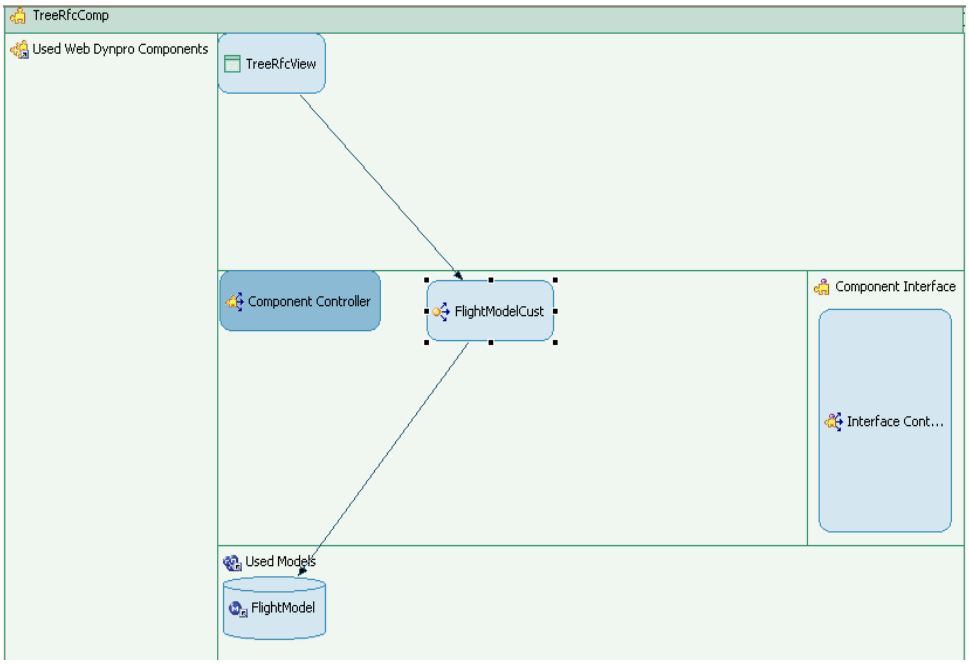

12. On the following dialog box, select all nodes. Click on OK.



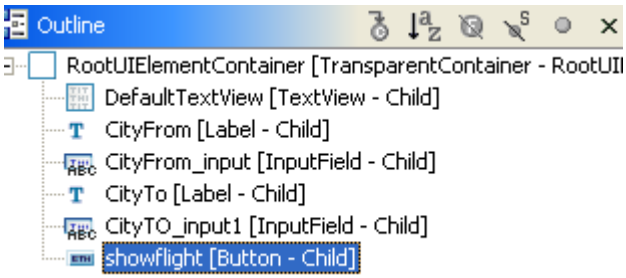
13. You will see the context mapping being created. Click on Finish.




14. The resulting Data Model should look like below.



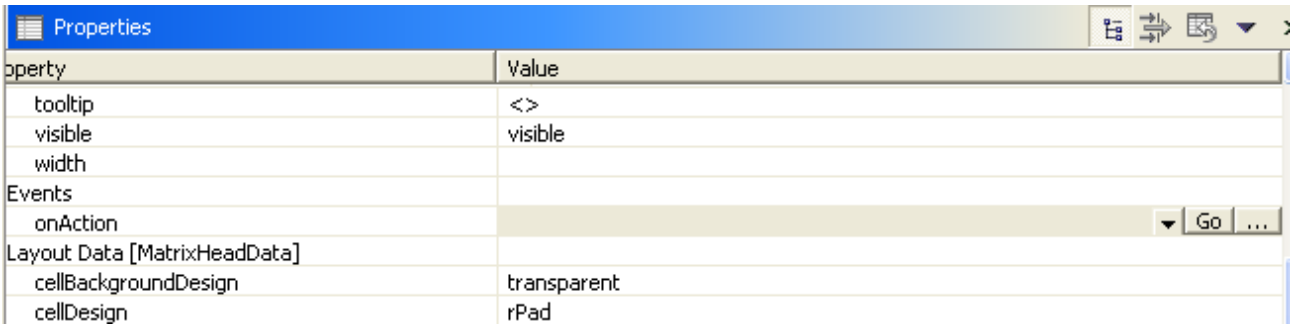
15. Open the TreeRfcView. Insert an label and input field for CityFrom and CityTo. and Insert a button for fetching the data on to the view.



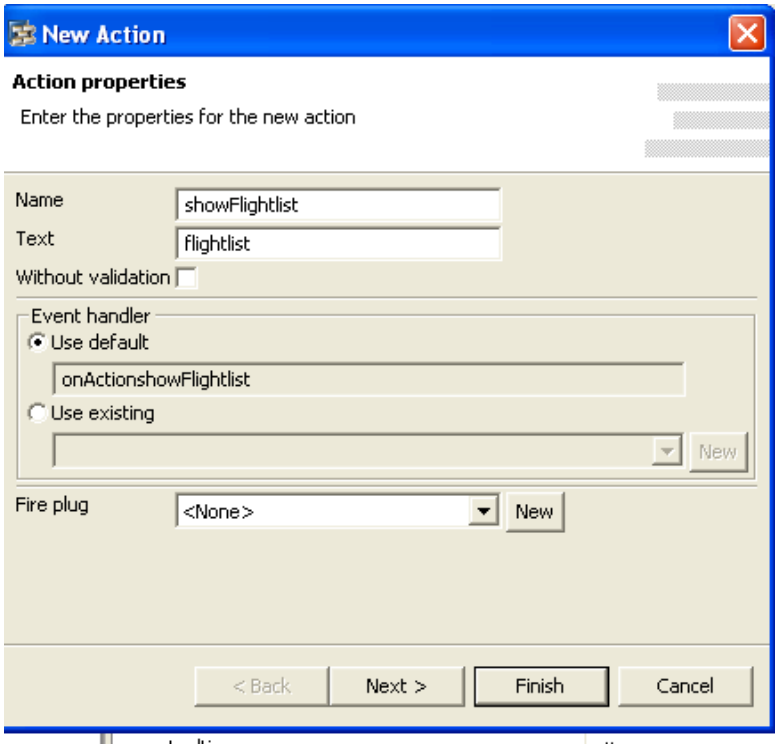

16. **Data Binding**
To display the data in a UI element, the appropriate properties of the UI element must be bound to the context nodes or context attributes.

| **Object** | **Object_Id** | **Object Property** | **value** |
|---|---|---|---|
| Label | CityFrom | Text | CityFrom |
| InputField | CityFrom_input | Value | Bapi_Flight_Getlist_Input.Destination_From.City |
| Label | CityTo | Text | CityTo |
| InputField | CityFrom_input | Value | Bapi_Flight_Getlist_Input.Destination_To.City |
| Button | showflight | OnAction | showFlightlist (here yoy have to create one method showFlightlist) |

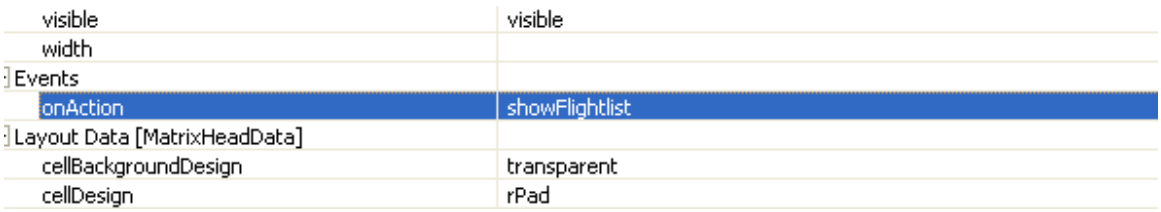
17. Create a method on Button Showflight. Click on Tab



18. Then A Dialog of new action will pop up. Write Name of Action and text and Click on Finish.

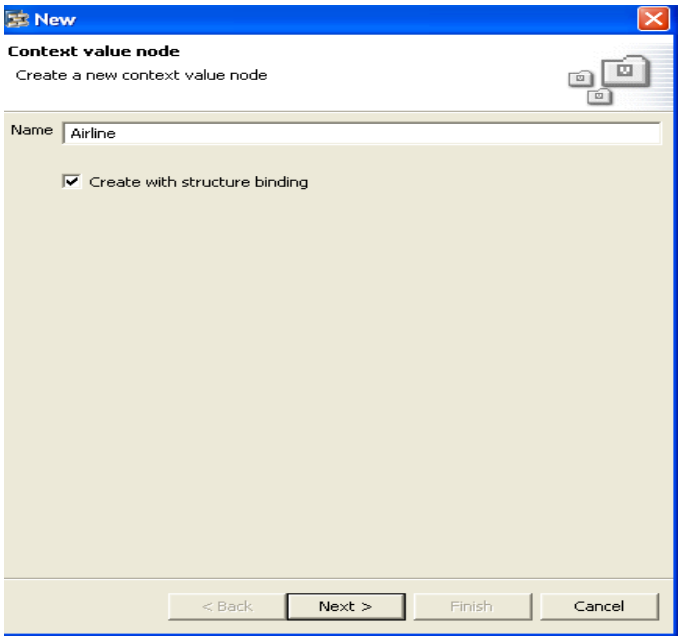



19. Method will be created on Button.

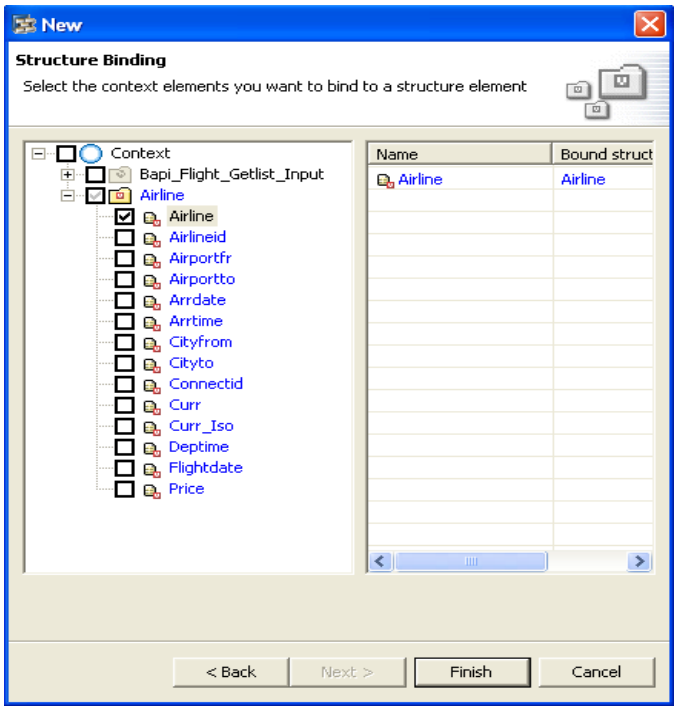


## Populate Data in Tree

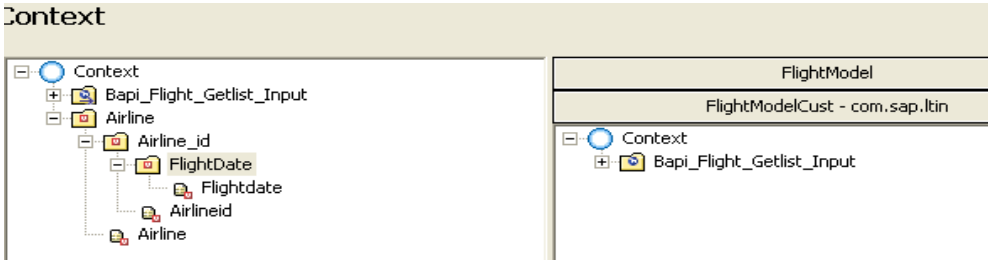
1. Open TreeRfcView Go to the Context tab of TreeRfcView. Create a Value Node name it Airline under the context root with structure binding and Click on Next.




2. A Dialog of Structure Binding will popup ans select Airline.Click on finish.



3. Under Airline Value node create Value node Airline_id and FlightDate with Structure Binding like we created Airline Node with steps 22 and 23. And Repeat this procedure until the context structure looks like the one in the graphic




4. Set the properties of Value node.

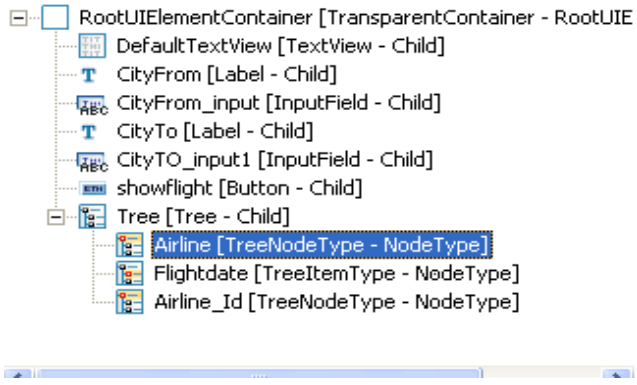
| Value node | Property of Value Node | Set Value |
|---|---|---|
| **Airline** | cardinality | 0..n |
| | selection | 0..1 |
| | singleton | true |

Cardinality and selection of Airline_id and Flightdate node is same as Airline node.

| | | |
|---|---|---|
| **Airline_id** | singleton | False |
| **FlightDate** | singleton | False |

## Tree Creation in the TreeRfcView.

1. Go back to the Layout tab. Insert Tree UI element on to the TreeRfcView by right clicking on default container “RootUIElementContainer”. To create a complete Tree UI element, you add the required subelements of the type *TreeNodeType* and *TreeItemType* to the Tree UI element, which then represent the complete tree. See the graphic below.



2. To display the data in a UI element, the appropriate properties of the UI element must be bound to the context nodes or context attributes.

    a. Navigate to a property and choose the graphic  in the properties window. The  button appears. It enables you to access the Context Viewer dialog box.

    b. Select a context node or the context attribute in the dialog box.

    c. Confirm by choosing OK

    d. The following table lists the main data binding relationships of the Tree example Tree_0. In the same way, the individual associated subelements TreeNodeType and TreeItemType are bound to the corresponding context nodes and context attributes.

| **Object(UI Element)** | **Object_id** | **Properties** | **Value** |
|---|---|---|---|
| **Tree** | Tree | `dataSource` property → value node Airline | Airline |
| Tree | Tree | roottext | FlightList |
| Tree | Tree | title | Flight details |
| TreeNodeType | Airline | `dataSource` property → value node Airline | Airline |
| TreeNodeType | Airline | `text` property → value attribute Airline | Airline.Airline |
| TreeItemType | FlightDate | `dataSource` property → value node FlightDate | Airline.Airline_id.FlightDate |
| TreeItemType | FlightDate | `text` property → value attribute FlightDate | Airline.Airline_id.FlightDate.Flightdate |
| TreeNodeType | Airline_Id | `dataSource` property → value node Airline_Id | Airline.Airline_id |
| TreeNodeType | Airline_Id | `text` property → value attribute Airline_Id | Airline.Airline_id.Airlineid |

### Filling Context with Data

1. The "ShowFlightList" method in the implementation enables you to fill the context of a view with data. Go back to the Implementation of TreeRfcView and add the following highlighted code to the end of the onActionshowFlightlist method.

```
public void onActionshowFlightlist(com.sap.tc.webdynpro.progmodel.api.
IWDCustomEvent wdEvent)
  {
   //@@begin onActionshowFlightlist (ServerEvent)
wdThis.wdGetFlightModelCustController().executeBapi_Flight_Getlist_Input();

for(int i=0;i<wdContext.nodeFlight_List().size();i++)
{
  IPrivateTreeRfcView.IAirlineNode airlinenode = wdContext.nodeAirline();
IPrivateTreeRfcView.IAirlineElement airline= airlinenode.createAirlineElement();
airline.setAirline(wdContext.nodeFlight_List().getFlight_ListElementAt(i).getAirline(
));
 airlinenode.addElement(airline);
airline.setAirline(wdContext.nodeFlight_List().getFlight_ListElementAt(i).getAirline(
));                                                                    cont…


date_ele.setFlightdate(wdContext.nodeFlight_List().getFlight_ListElementAt(i).getFlig
htdate());
                  Flightdatenode.addElement(date_ele);

                       }
   //@@end
  }      {

                   IPrivateTreeRfcView.IAirlineNode airlinenode =
wdContext.nodeAirline();
                   IPrivateTreeRfcView.IAirlineElement airline =
airlinenode.createAirlineElement();




IPrivateTreeRfcView.IAirline_idNode AirlineId = airline.nodeAirline_id();
 IPrivateTreeRfcView.IAirline_idElement Airlineid_ele =
AirlineId.createAirline_idElement();
 Airlineid_ele.setAirlineid("Airline ID: "+wdContext.nodeFlight_List().
 getFlight_ListElementAt(i).getAirlineid());
 AirlineId.addElement(Airlineid_ele);


IPrivateTreeRfcView.IFlightDateNode Flightdatenode = Airlineid_ele.nodeFlightDate();
IPrivateTreeRfcView.IFlightDateElement date_ele =
Flightdatenode.createFlightDateElement();
date_ele.setFlightdate(wdContext.nodeFlight_List().getFlight_ListElementAt(i).getFlig
htdate());
```

```
Flightdatenode.addElement(date_ele);

   }
   //@@end
  }
```

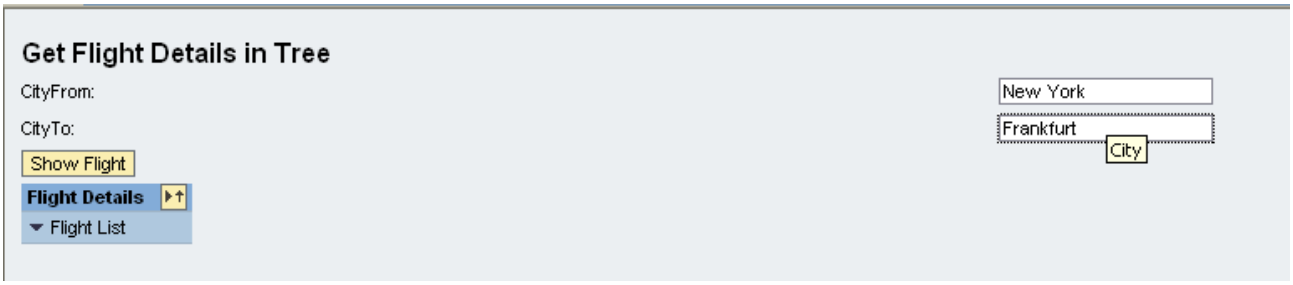
2. Go to the Implementation of custom controller FlightModelCust.Replace the generated code with the following code in `wdDoInit` method in the `FlightModelCust`controller implementation .

```
public void wdDoInit()
  {
    //@@begin wdDoInit()
    //$$begin Service Controller(-1438839805)
   Bapi_Flight_Getlist_Input input = new Bapi_Flight_Getlist_Input();
   wdContext.nodeBapi_Flight_Getlist_Input().bind(input);
   input.setDestination_From(new Bapisfldst());
   input.setDestination_To(new Bapisfldst());
    //$$end
    //@@end
  }
```

3. Save All Meta-data.

## Deploy the Application TreeRfc

1. Deploy and Run TreeRfc. Enter some valid data for flight and click on Show Flight. Check if the detail of this flight can be shown.




2. The output would look like this.

## Related Content :

How To execute an RFC model with inputs from Interactive Forms.

Tree API

Tree Demo Application

For more information, visit the User Interface Technology 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.