Java Web Dynpro:
Create Sales Order Using Bapi

Author: Monica Radytia Chrismawati
Created on: 15 June 2013

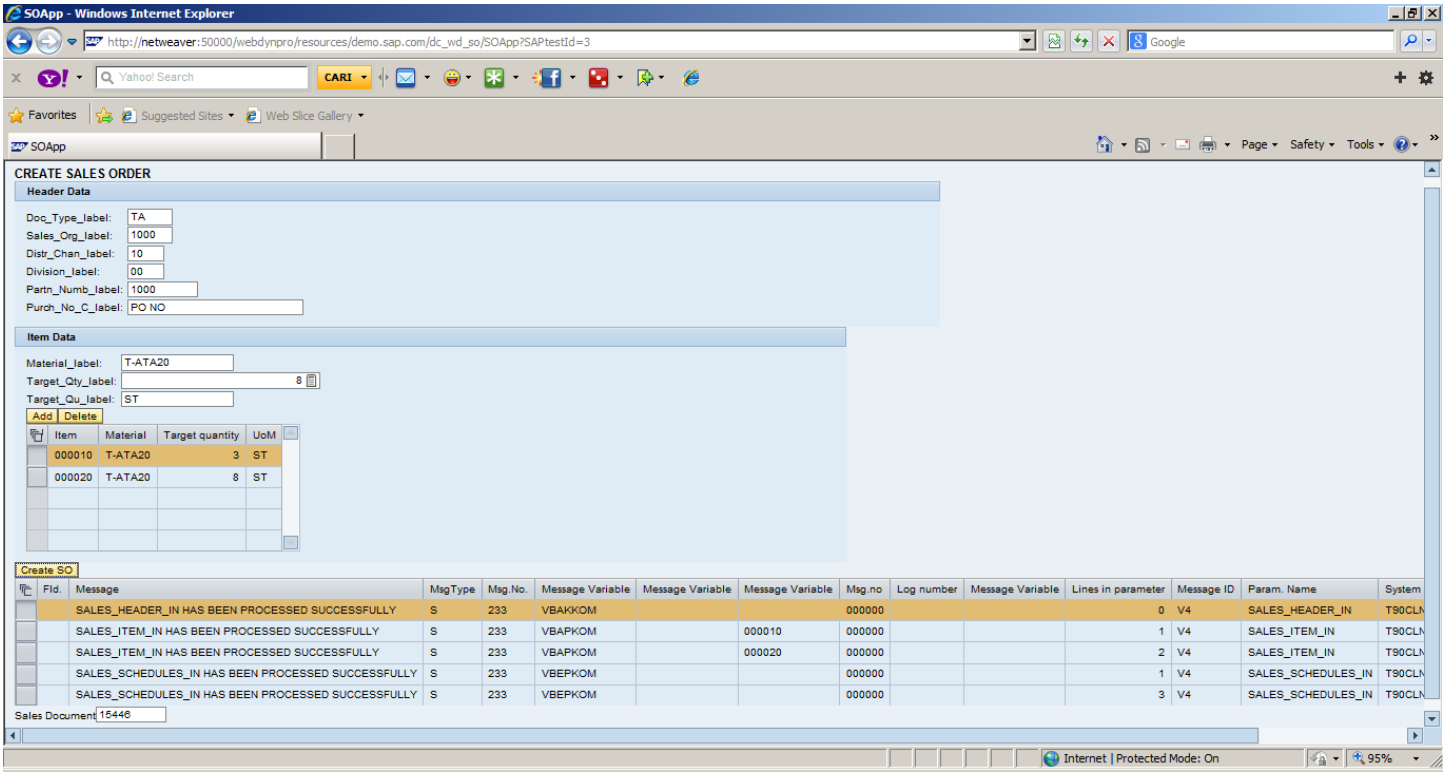
Applies to:

SAP Netweaver Composition Environment 7.3

Summary:

In this tutorial, we will try to create sales order using BAPI in Java Web Dynpro. The final result will be like the screenshot below. We will try to bind data that comes from user input into BAPI. This tutorial will use **BAPI_SALES_CREATEFROMDAT2, BAPI_TRANSACTION_COMMIT,** and **BAPI_TRANSACTION_ROLLBACK**. The main concept about the use of BAPI in Java Web Dynpro is BAPI will be imported as model data that will be converted as java class which will be used in developing java code.

As your basic knowledge in SAP SD, sales order is formed from three big section: header level, item level, and schedule lines level. We will provide a form that will be filled up with header data, multiple items data that can be added and deleted, then will show the return log after user press the button to create sales order.



Tools:

- SAP NetWeaver Developr Studio SAP Netweaver 7.3 SP00 PAT0000.

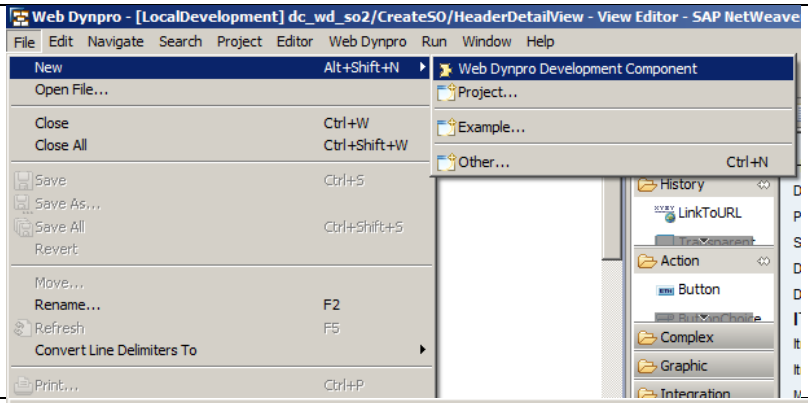
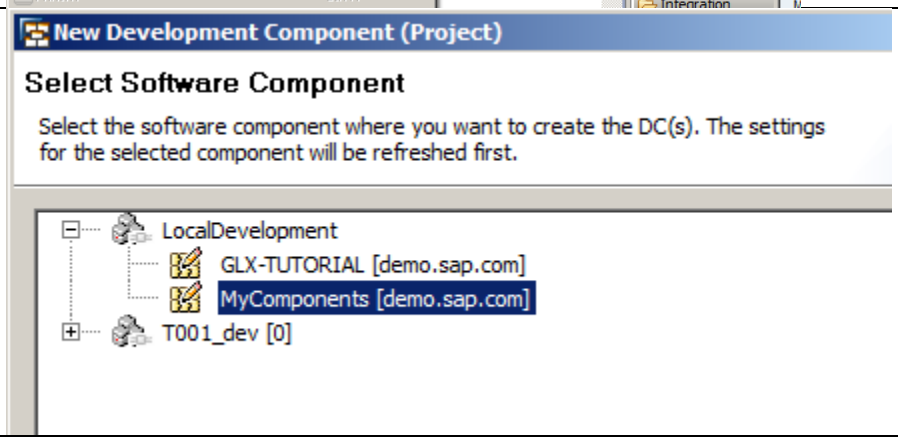
Prerequisite:

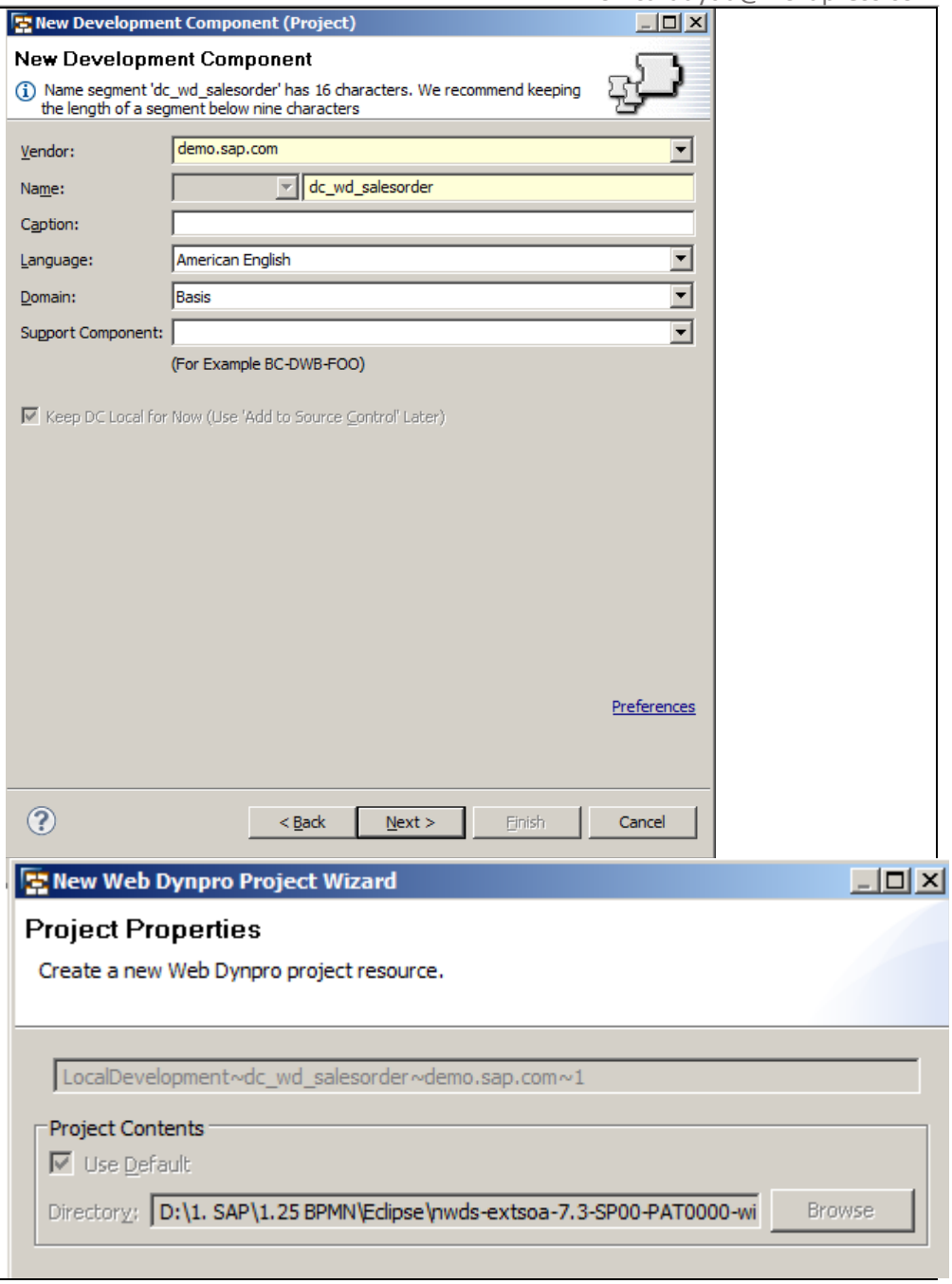
- ❒ SAP NetWeaver Developr Studio is installed in your computer
- ❒ SAP J2EE Engine is up.
- ❒ SAP As Java in NWDS already configured.
- ❒ R/3 Configuration is already configured.
- ❒ You have access to a remote SAP back-end system (SAP R/3).
- ❒ The SAP System Landscape Directory (SLD) and the SLD bridge are configured and running.

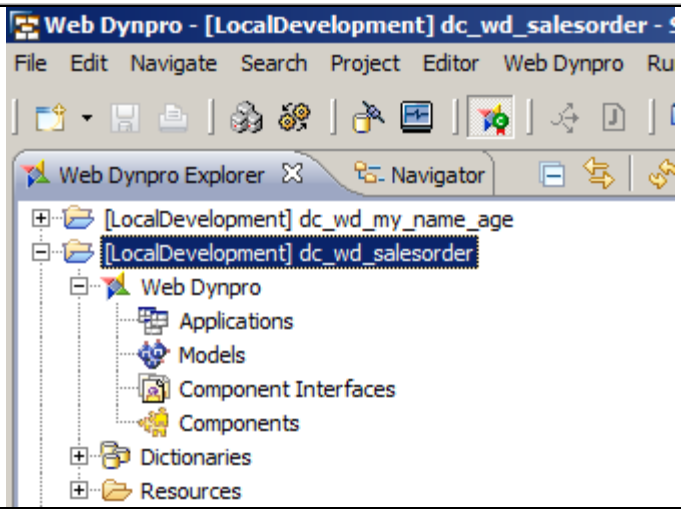
Knowledge:

- ❒ You have some experience with Java Web Dynpro applications.
- ❒ You have experience in Java programming.
- ❒ You are familiar with the use of Function Modules and BAPIs.

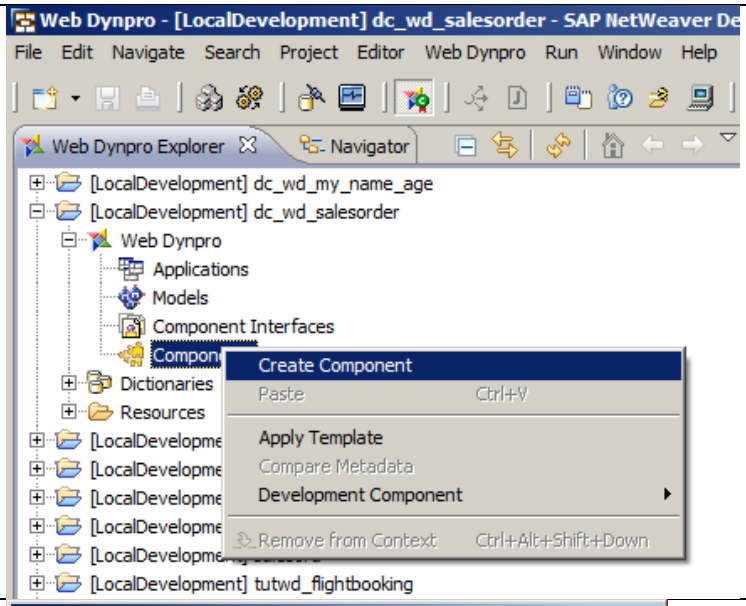
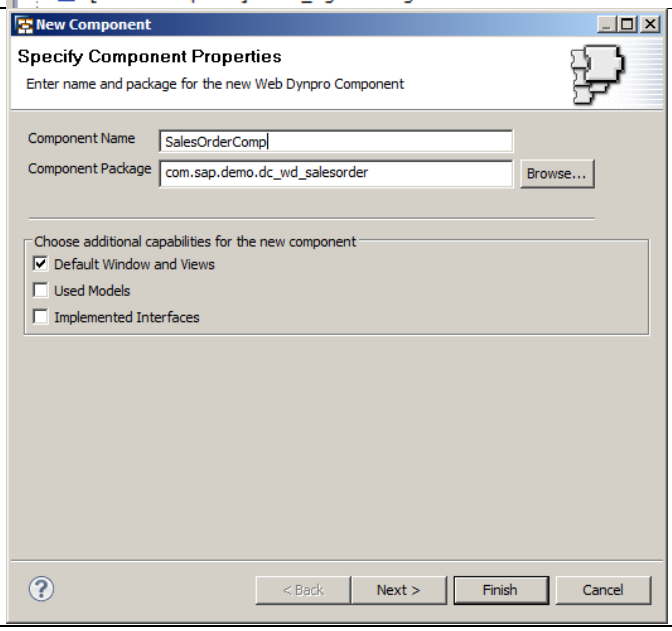
STEP 1. CREATE A NEW DEVELOPMENT COMPONENT

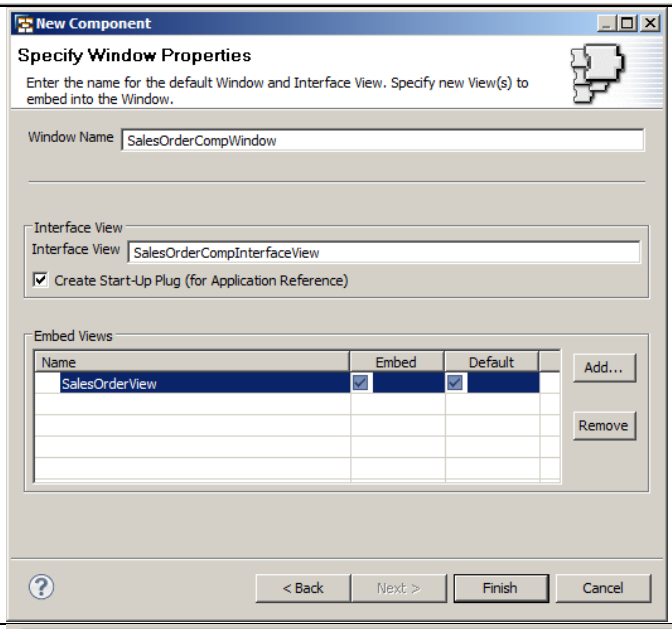
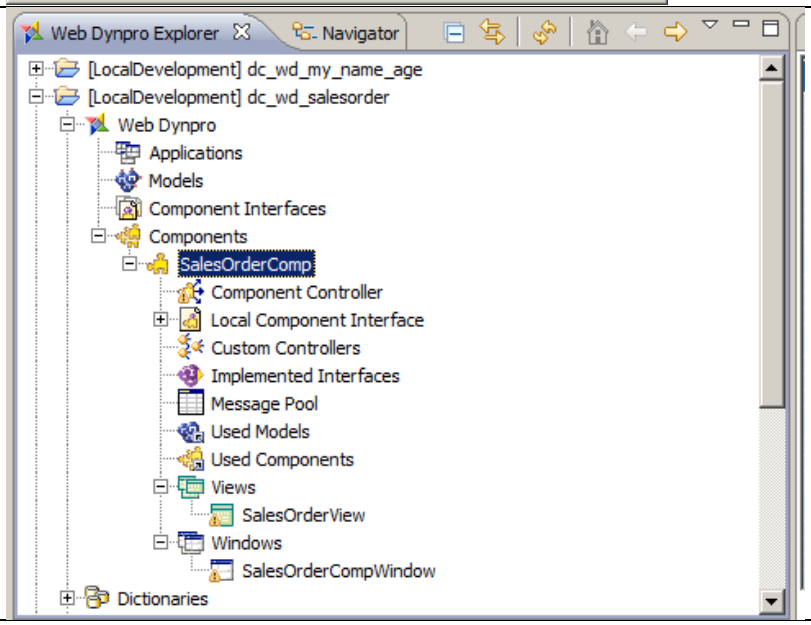
1	Start SAP NetWeaver Developer Studio (NWDS). *File -> New -> Web Dynpro Development Component.*	
2.	Select the appropriate *Software Component* then click *Next* button.	

3.	A new window to create *Development Component* appear. Type "dc_wd_salesorder" as name of the new development component. Click *Next.*	

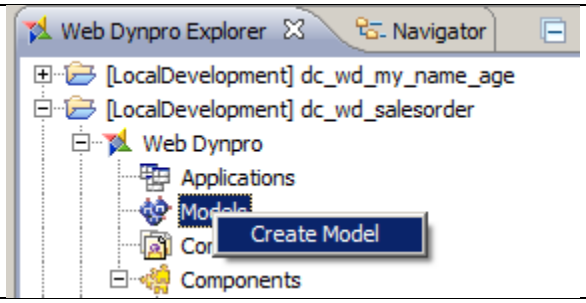
4.	A new web dynpro development component is created. An empty structure of this dc will be shown like screenshot beside.	

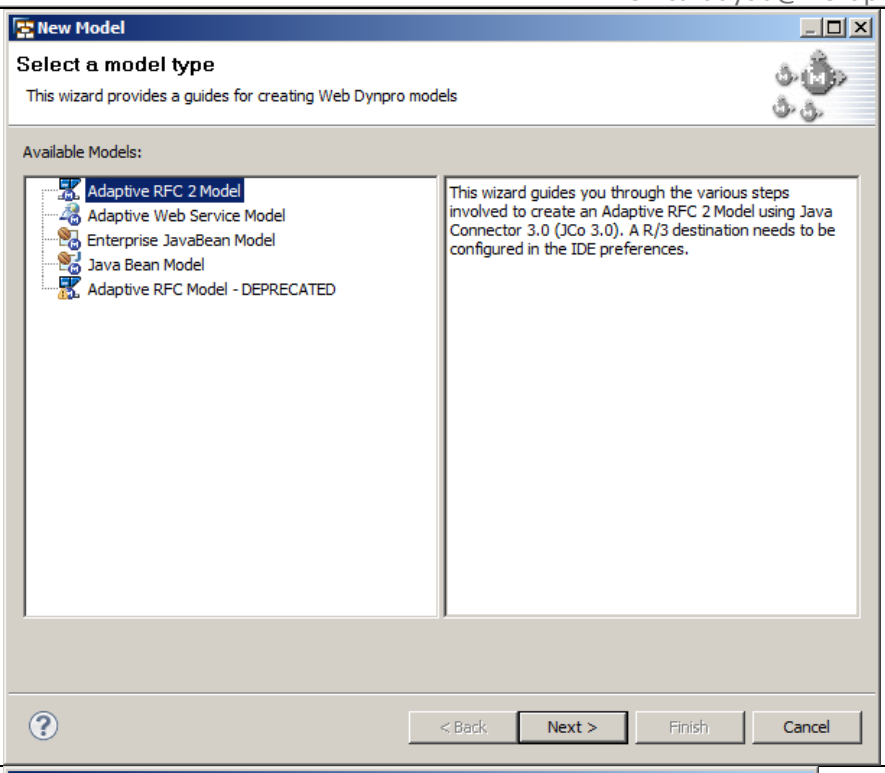
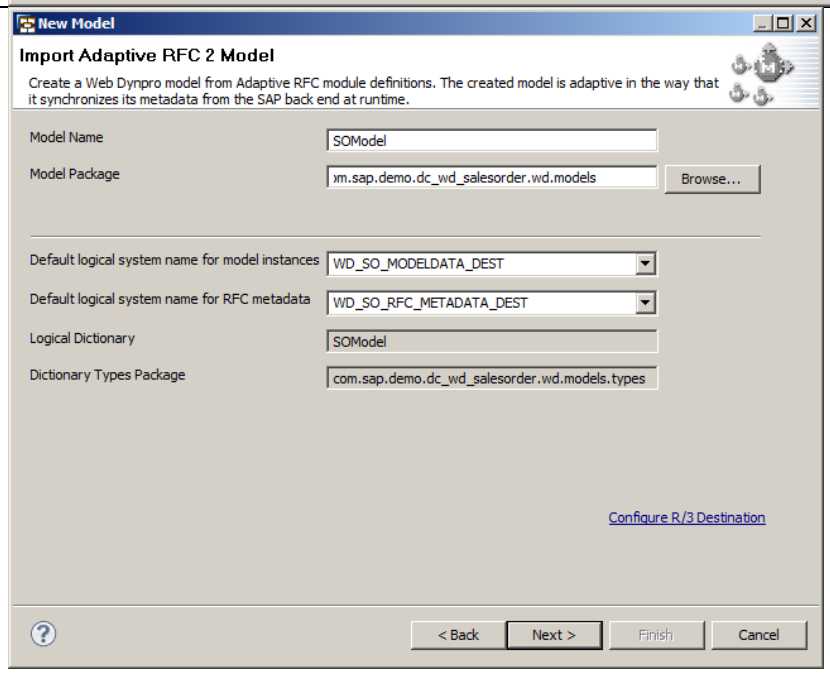
STEP 2. CREATE COMPONENT

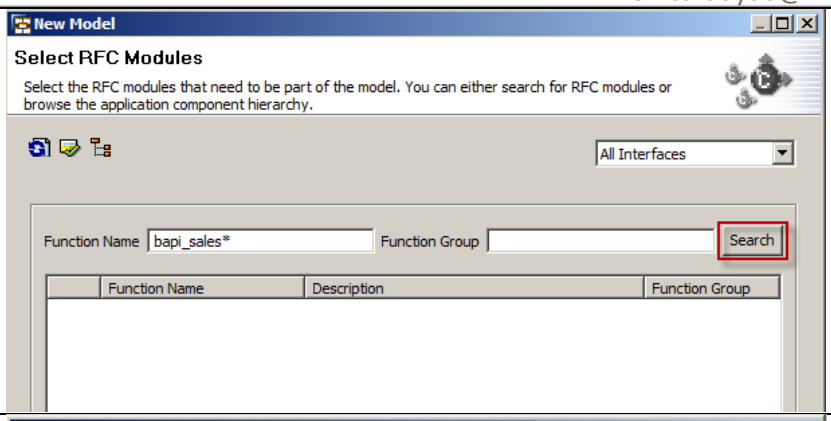
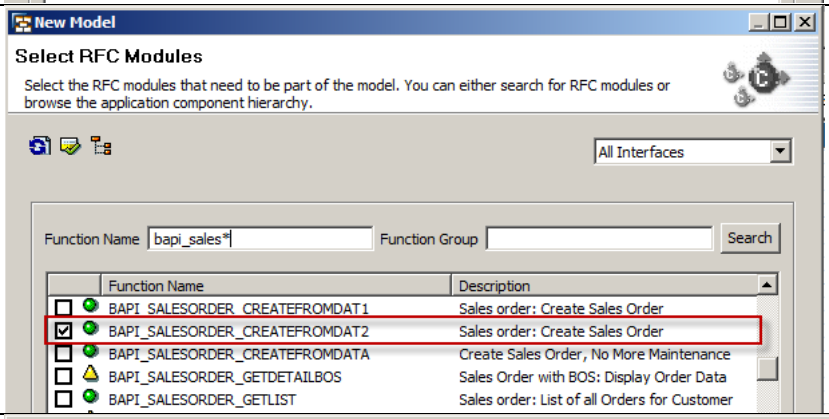
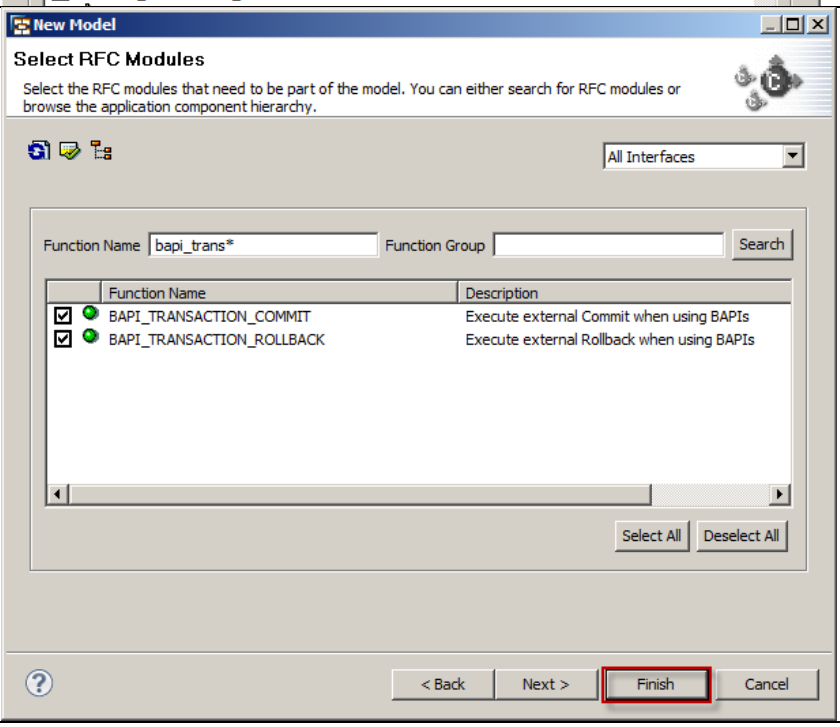
5.	Click on node *Component* then right click on it. Choose *Create Component.*	
6.	Type "SalesOrderComp" in *Component Name* field. Then choose the appropriate *Component Package.* Thick on *Default Windows and Views* only, since we will used model later.	

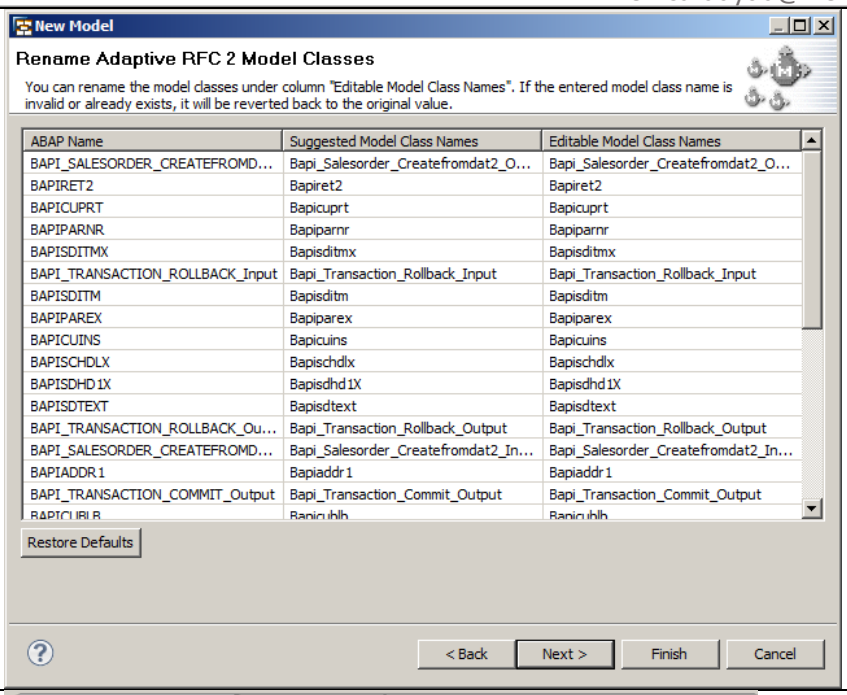
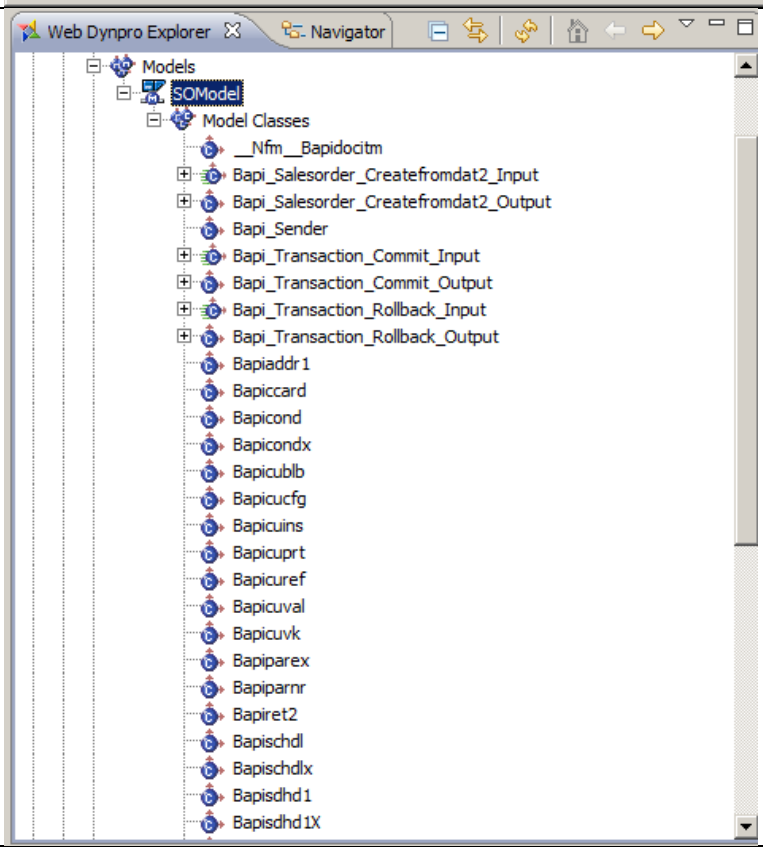
7.	A new window appear. If there is no revise from suggested name of window, interface, and view, continue by click *Finish.*	
8.	The structure of our dc after created a component will be look like this screenshot.	

STEP 3. IMPORTING BAPI AS MODEL

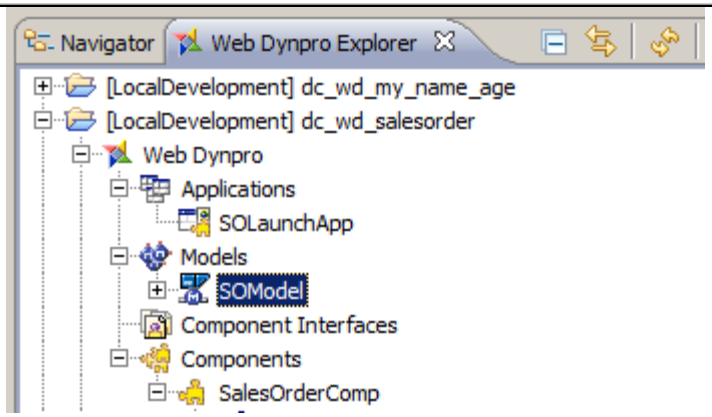
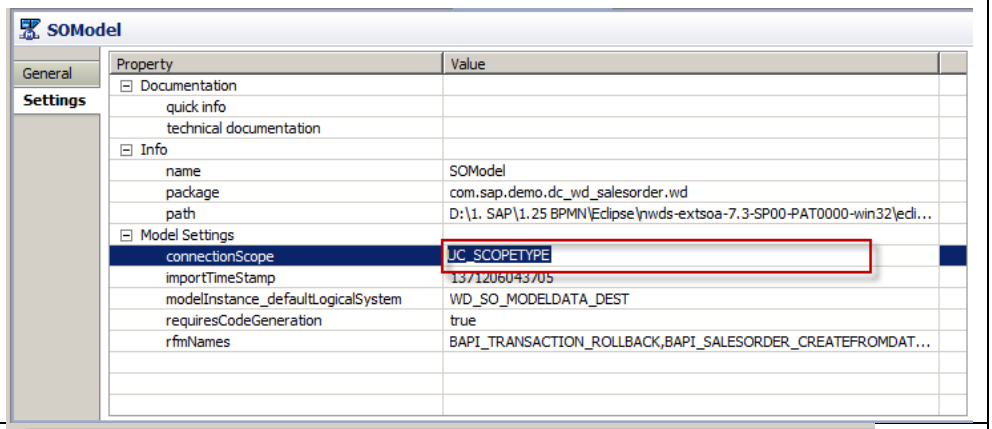
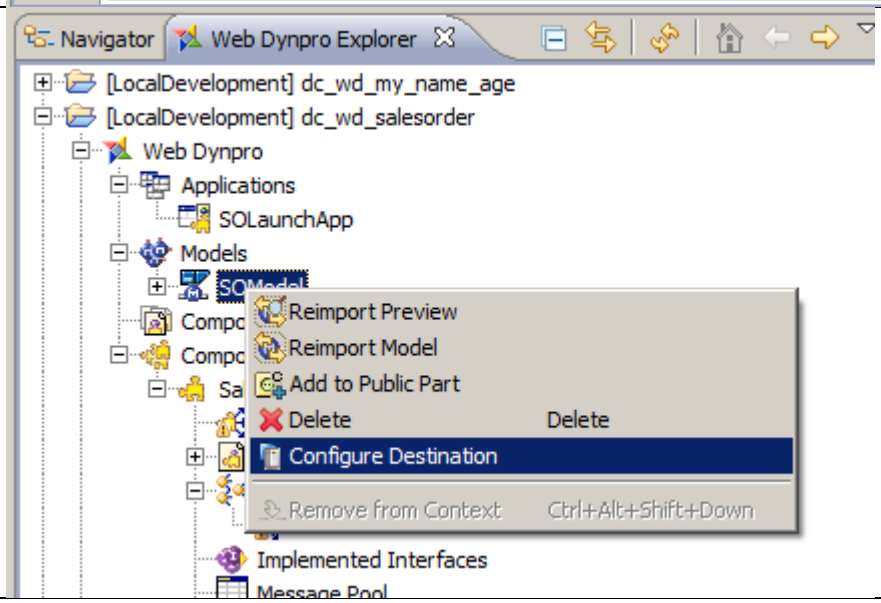
9.	Right click on *Models* > *Create Model*.	

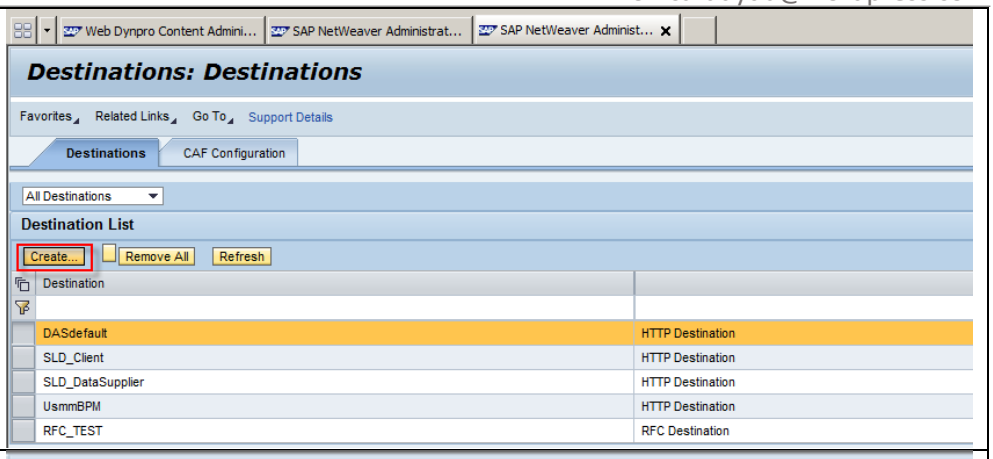
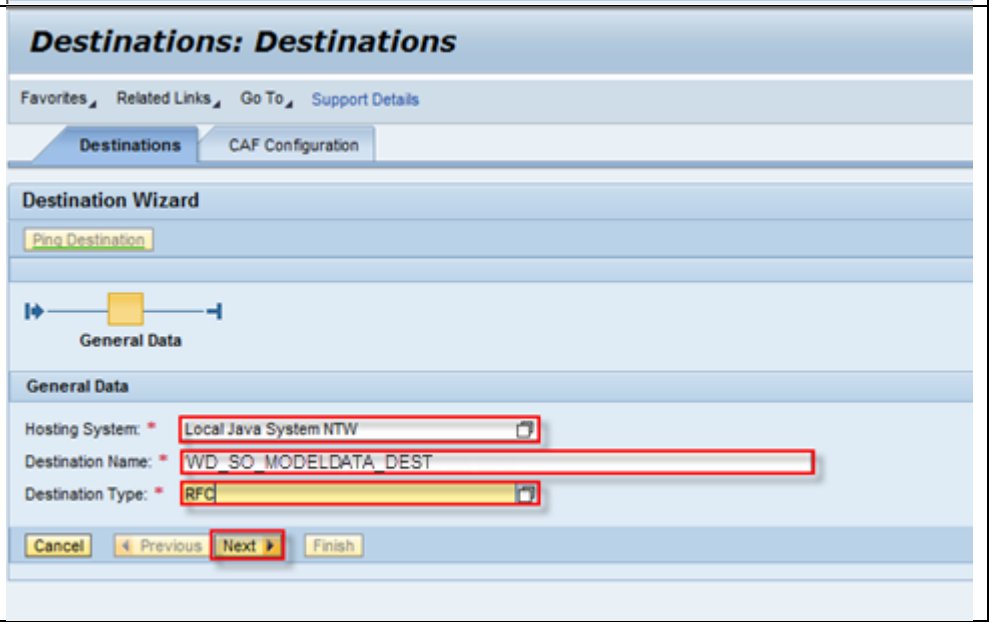
10.	Click on *Adaptive RFC 2 Model* > *Next.*	
11.	Type "SOModel" in Model Name. Browse appropriate model package, or rename the suggested name. Rename the Default logical name for model instance: "WD_SO_MODELDATA_DEST" and Default logical system name for RFC metadata: "WD_SO_RFC_METADATA_DEST" Click Next.	

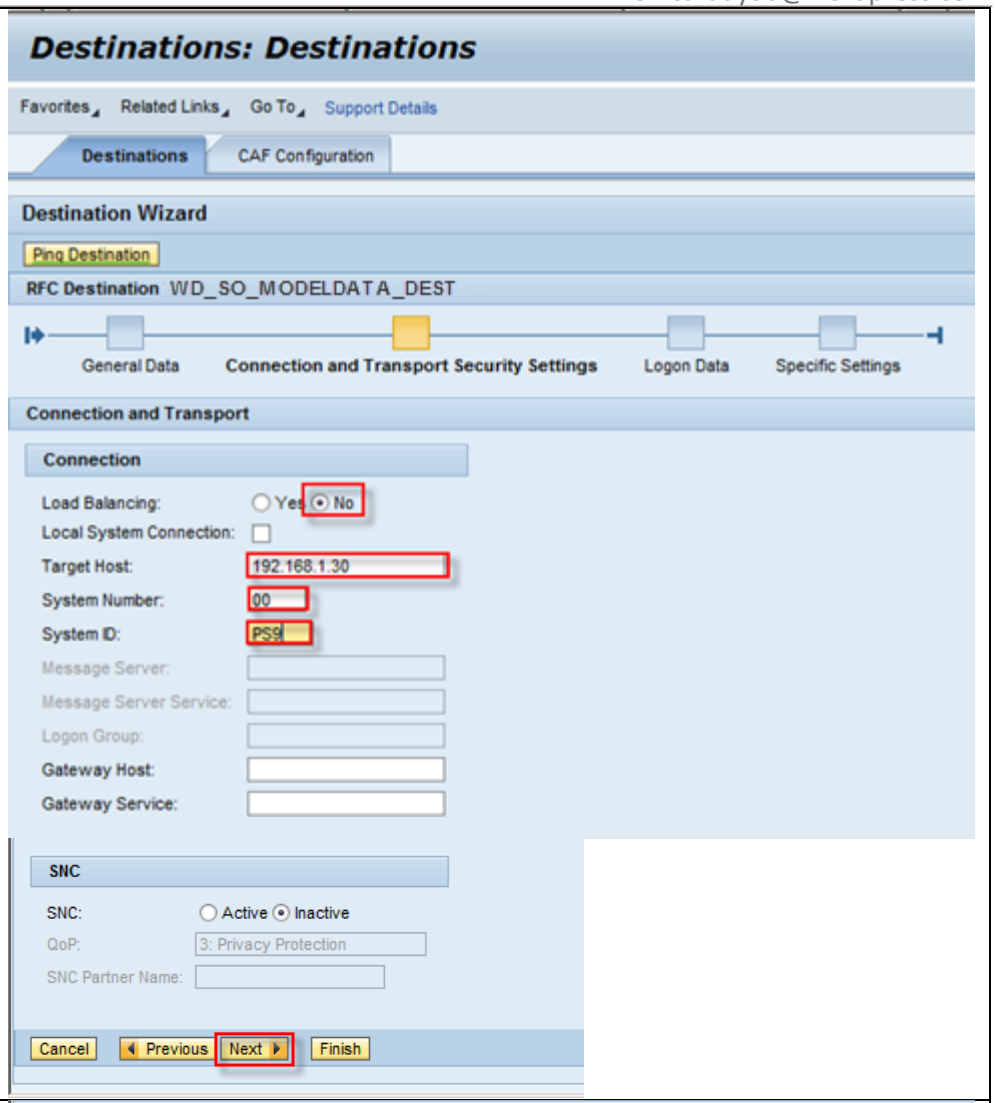
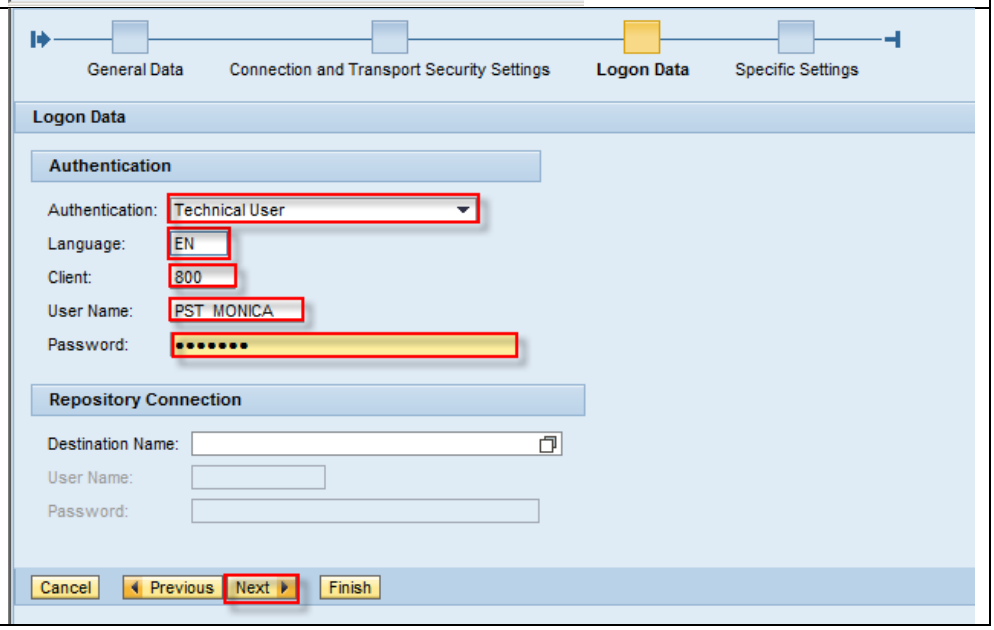
12.	In new window appear, type in Function Name for the 3 BAPIs that will be used in this tutorial: - BAPI_SALES_CREATEFROMDAT2 - BAPI_TRANSACTION_COMMIT - BAPI_TRANSACTION_ROLLBACK	
13.	Thick on three BAPIs that will we use.	
14.	Click Finish.	

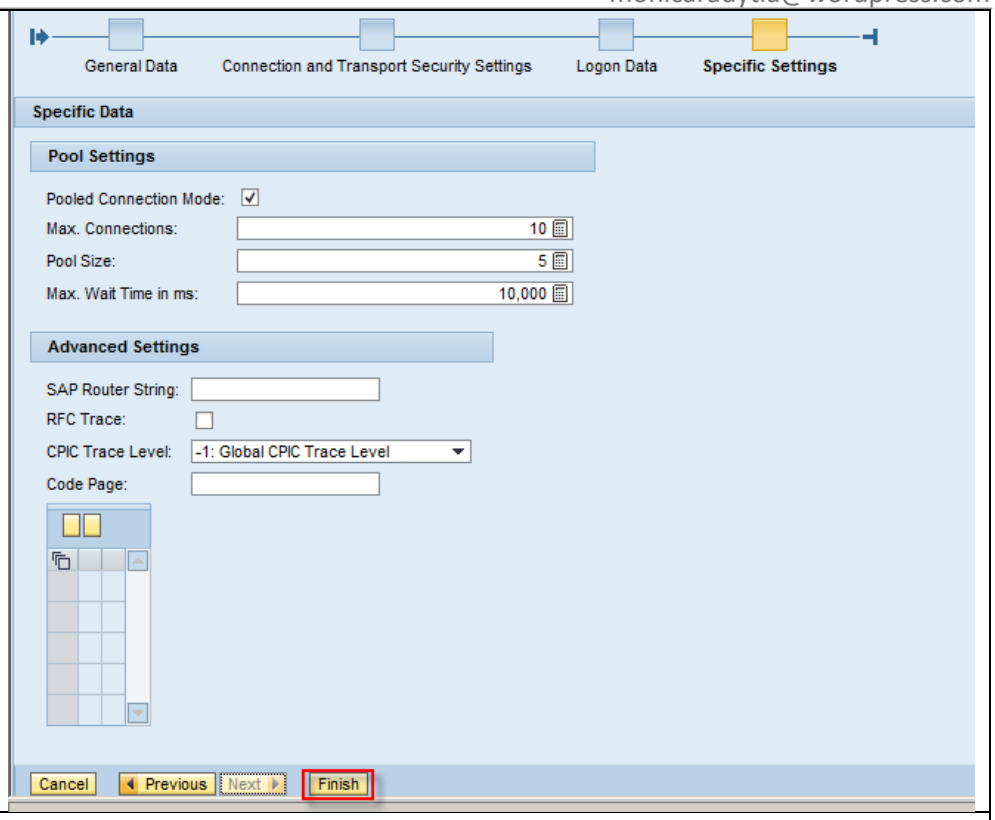
15.	If you need rename, on next screen appear you can do that, but in this tutorial, we will remain the name of model as default. Click Next.	
16.	If success, then tree structure of SOModel will appear as screenshot beside.	

STEP 4. CONFIGURE MODEL DESTINATION

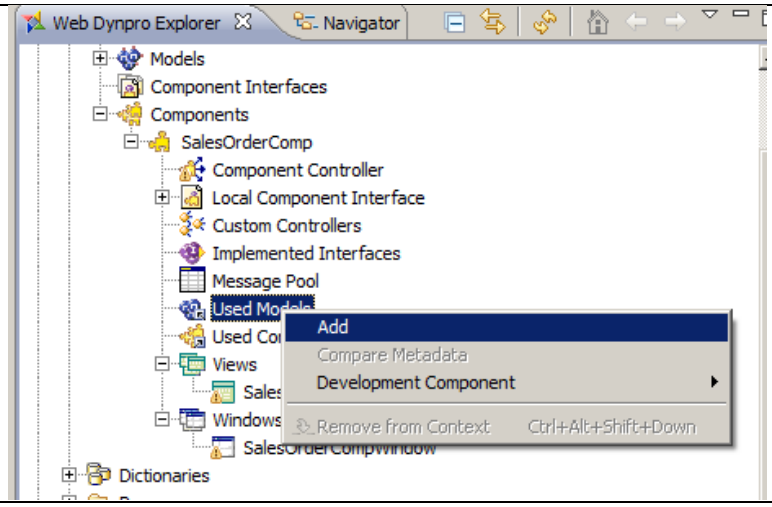
17.	Make sure your model's connection type is "UC_SCOPETYPE". Double click on model you have.	 
18.	Right click on model, *SOModel > Configure Destination*.	

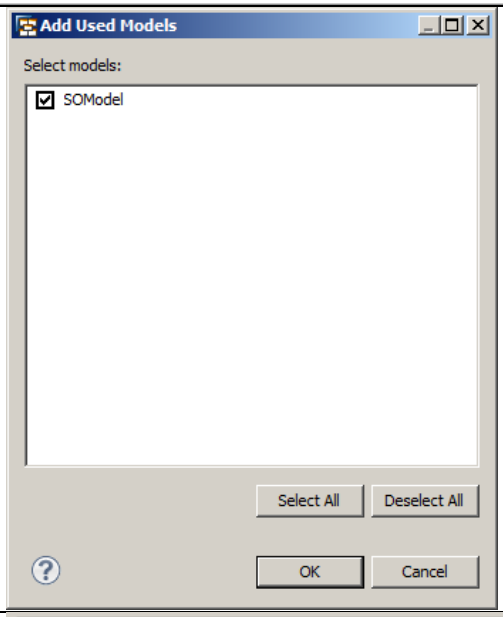
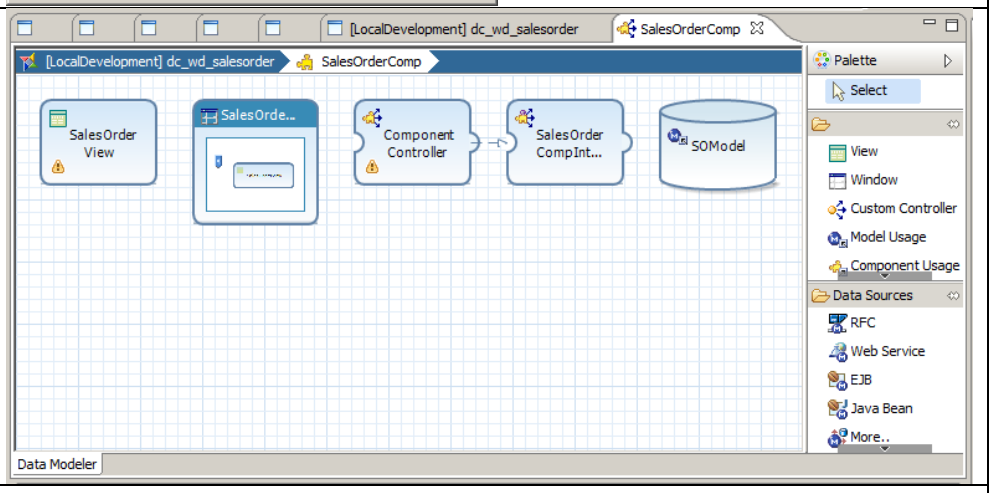
19.	Click Create.	
20.	Fill the Destination with your RFC Application Data First. Click Next.	

21.	Fill up the form with target host of your SAP Backend. Click Next.	
22.	Fill with your username that will you use to login in SAP Backend system. Click Next.	

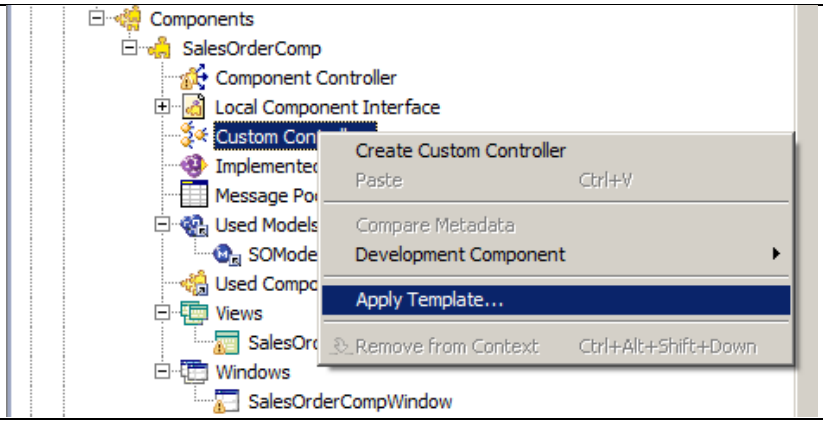
23.	Leave default. Click Finish.	

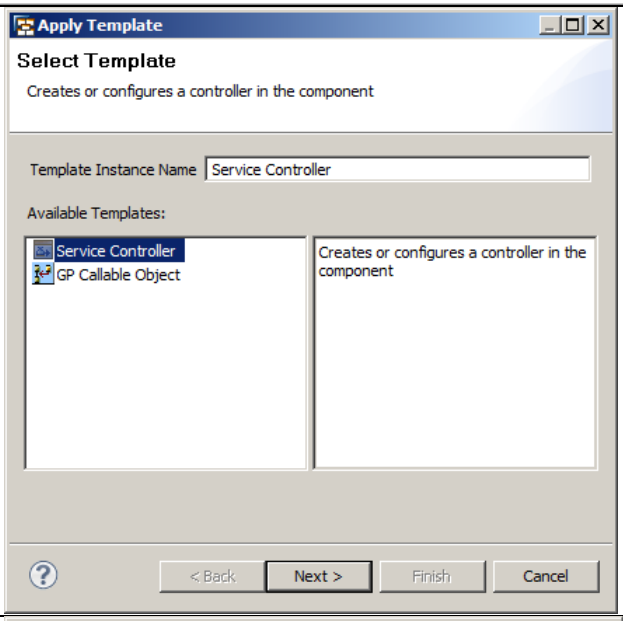
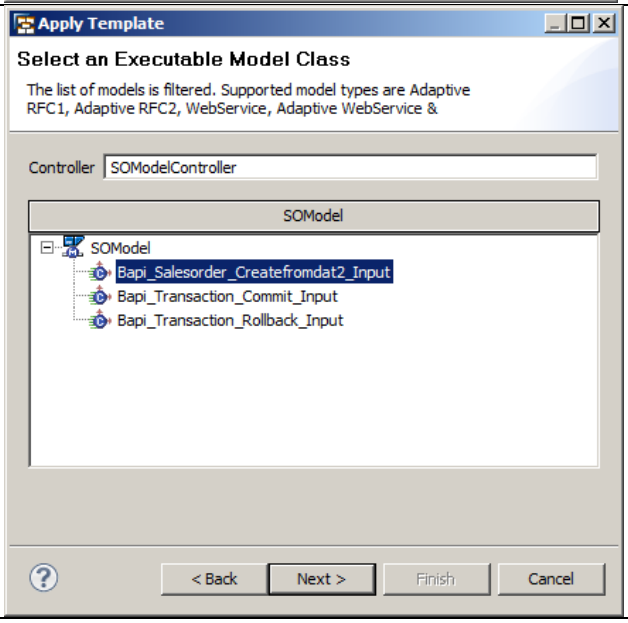
STEP 5. USED MODEL IN COMPONENTS

24.	Expand the structure of the tree Components. Right click on *Used Models > Add.*	

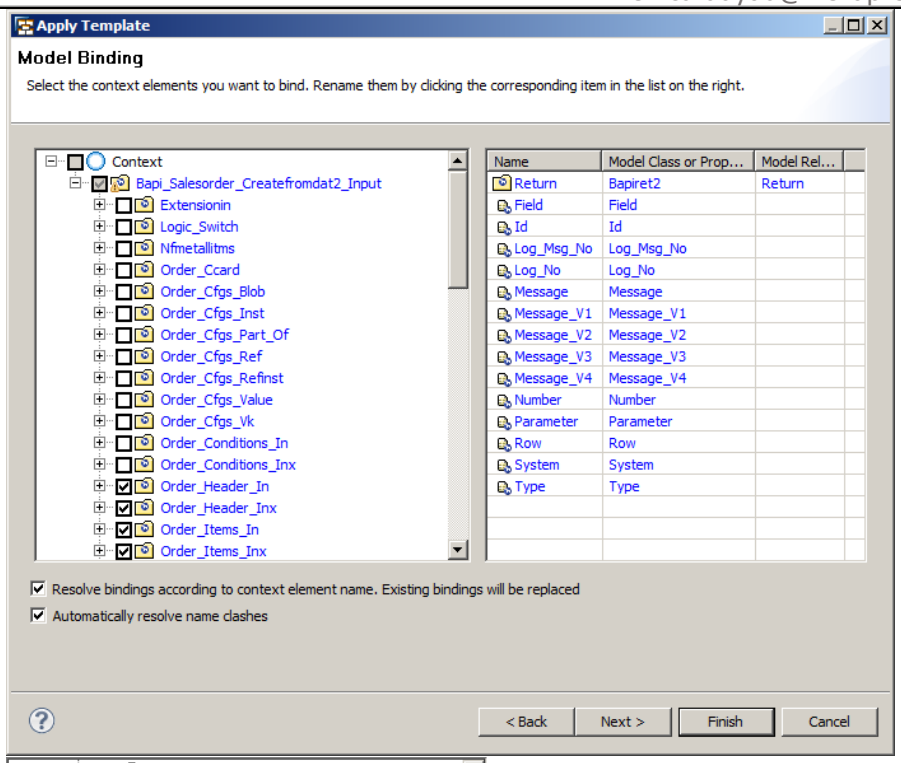
25.	On pop up window appear, select models. Click OK.	
26.	Open the Data Modeler of the components. Here we can see that model already used in our SalesOrderComp.	

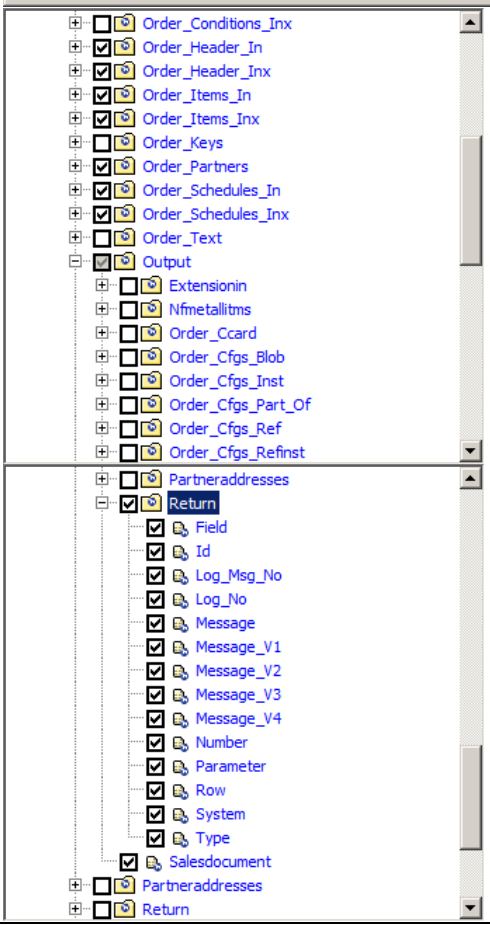
STEP 6. CREATE CUSTOM CONTROLLER

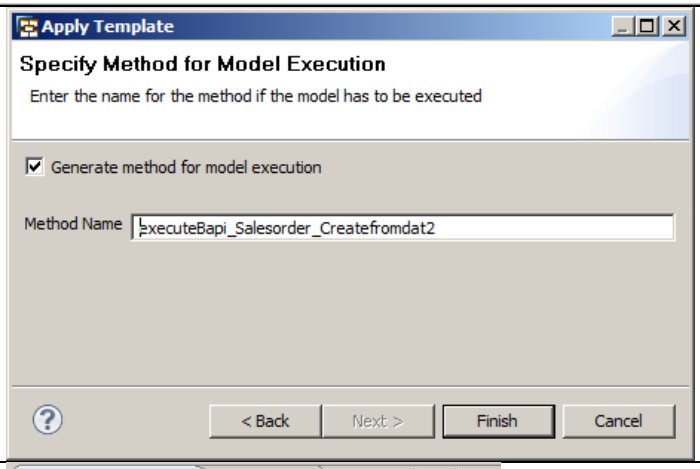
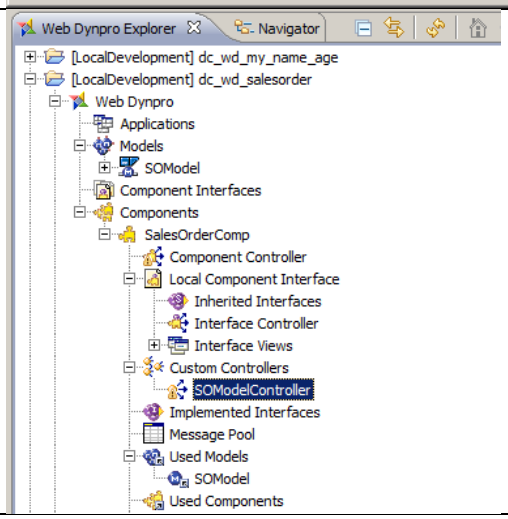
27.	Right click on *Custom Controller > Apply Template.*	

28.	Choose Service Controller. Click Next.		
29.	Choose our first model that will we use. Click Next.		

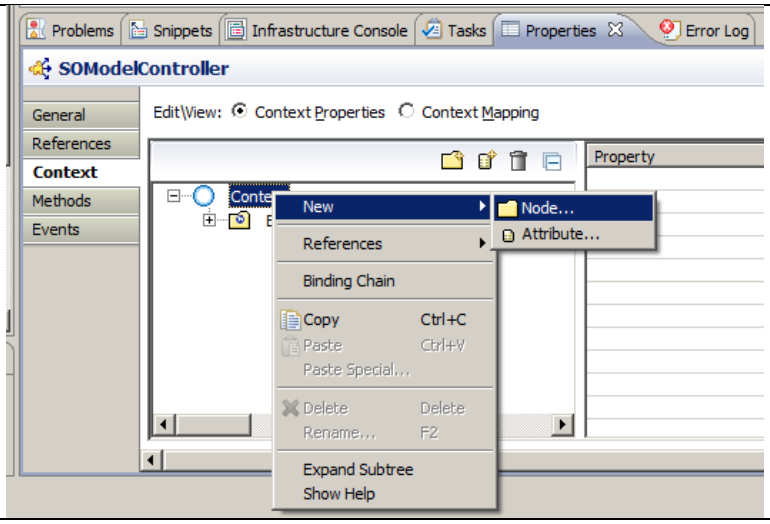
30. Structure of BAPI_SALES_CREATE_FROMDAT2 appear. Here we only use: **Order_Header_In, Order_Header_Inx, Order_Items_In, Order_Items_Inx, Order_Partners, Order_Schedules_In, Order_Schedules_Inx, Output.** Click Finish.

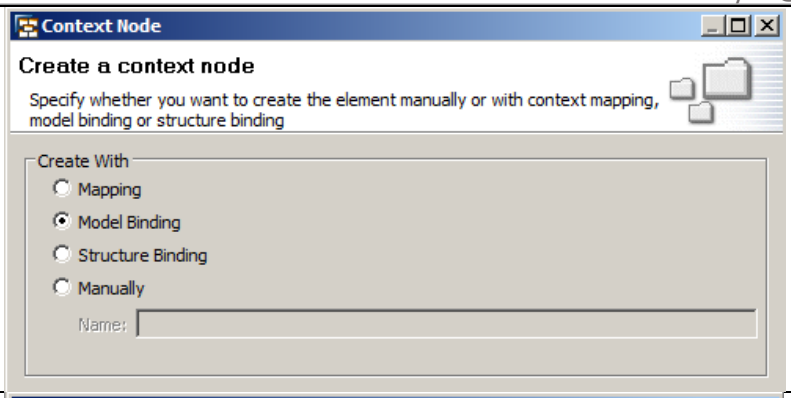
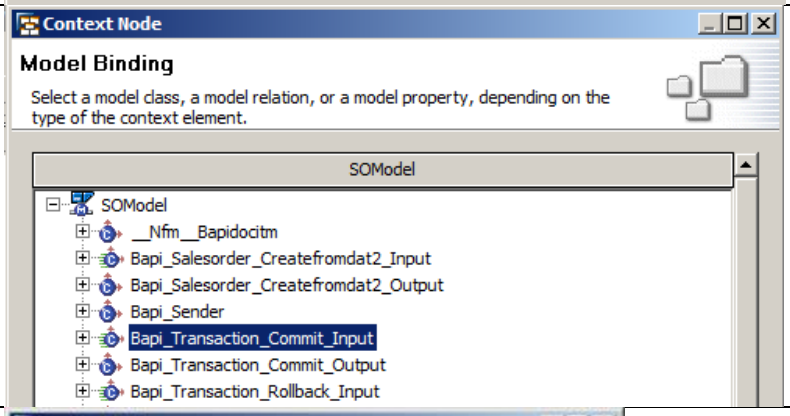
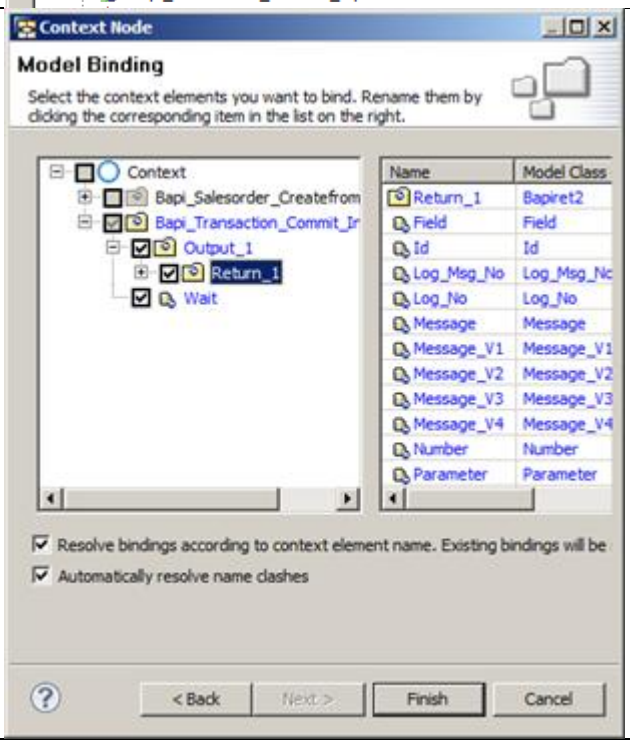




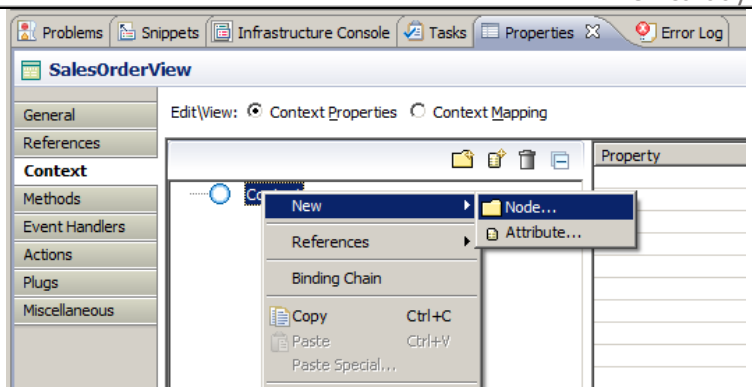
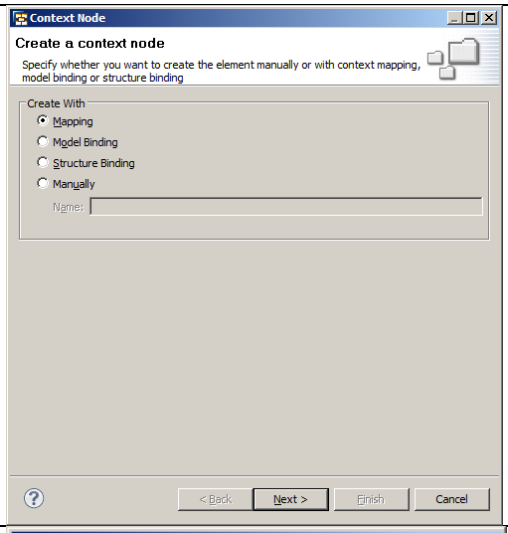
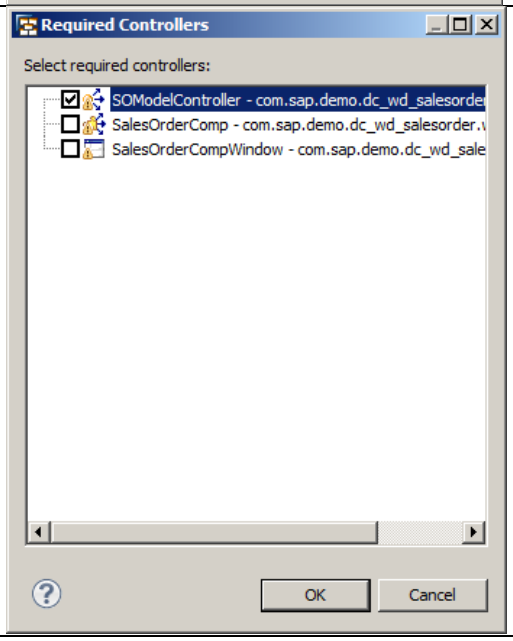
31.	On Method Name rename the method if you will. If not leave default. Click Finish.	
32.	Custom Controller *SOModelController* is created. The model **Bapi_Salesorder_Createfromdat2_Input** also automatically bind into custom controller's context.	

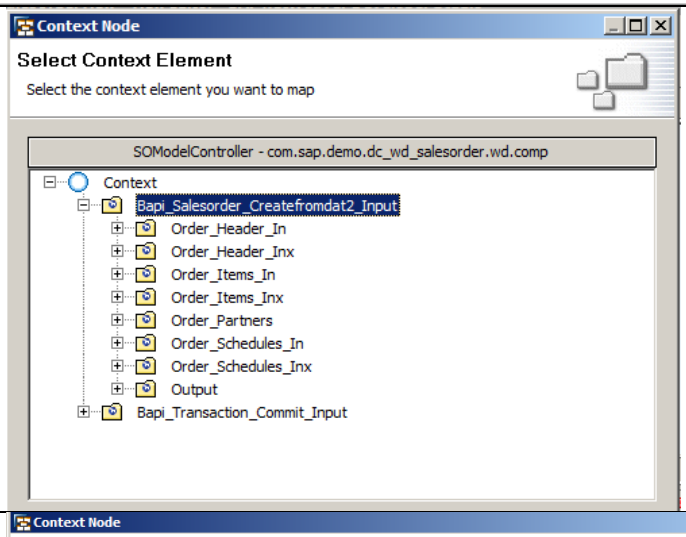
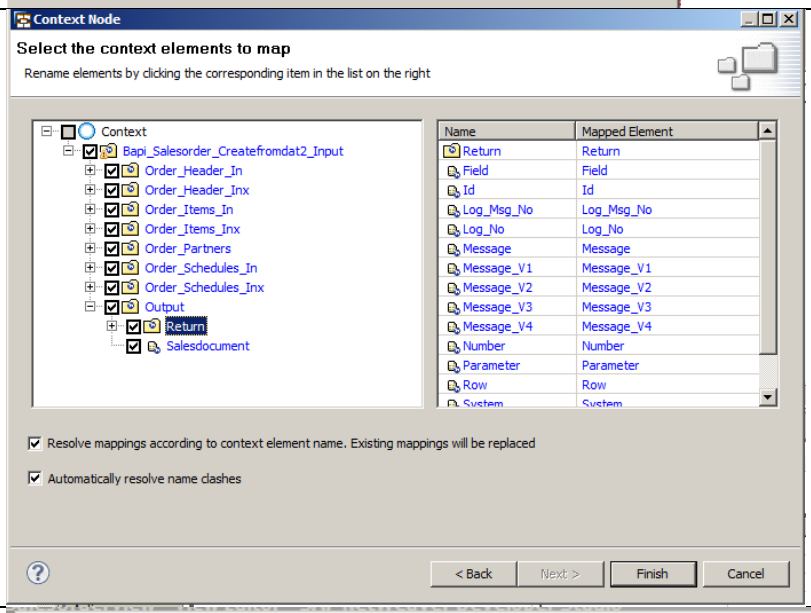
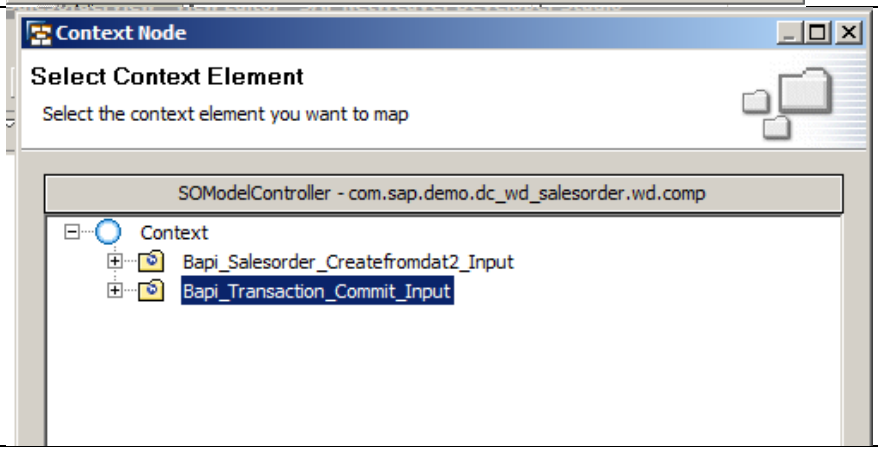
STEP 7. ADD CONTEXT USING MODEL BINDING IN CUSTOM CONTROLLER

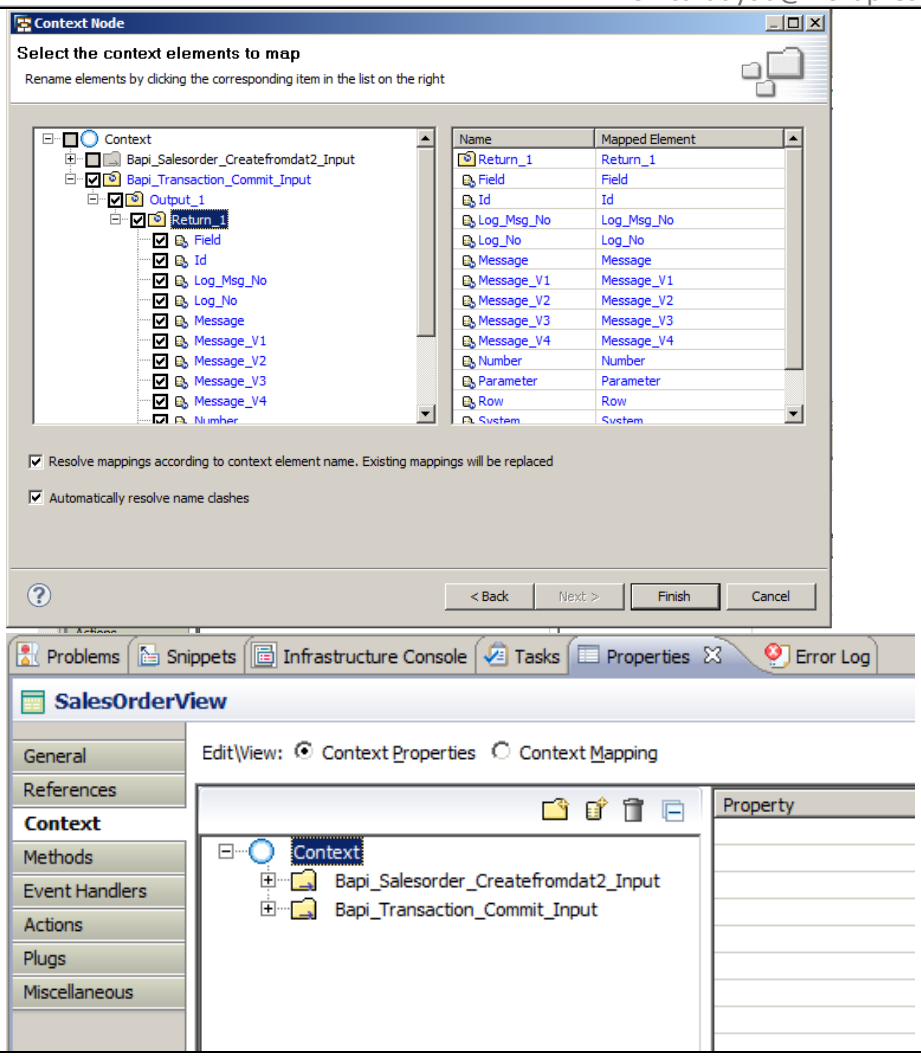
33.	Add another model of **Bapi_Transaction_Commit_Input.** Double click in custom controller. Open tab Context. Right click on *Context > New > Node.*	

34.	Choose Model Binding. Click Next.	
35.	Choose **Bapi_Transaction_Commit_Input.**	
36.	Choose the appropriate node. Finish.	

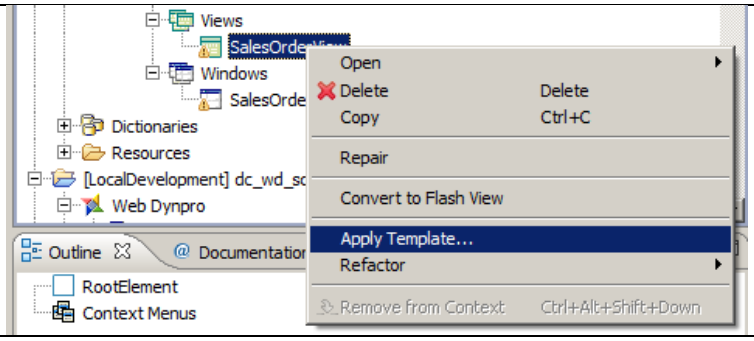
STEP 8. ADD CONTEXT USING MODEL BINDING IN SALES ORDER VIEW

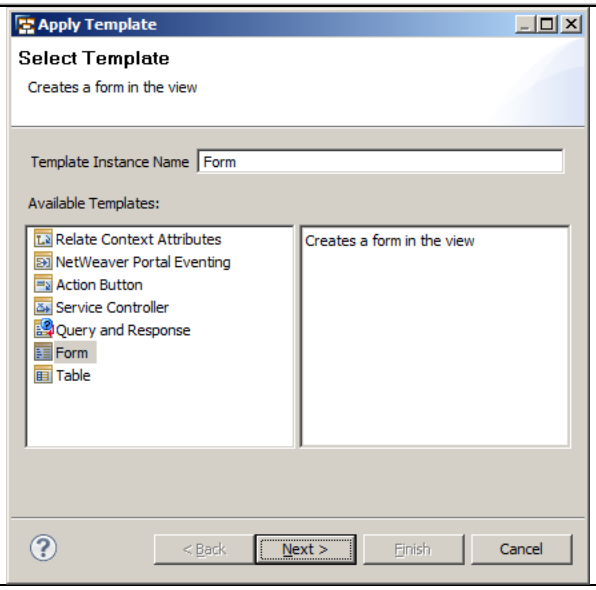
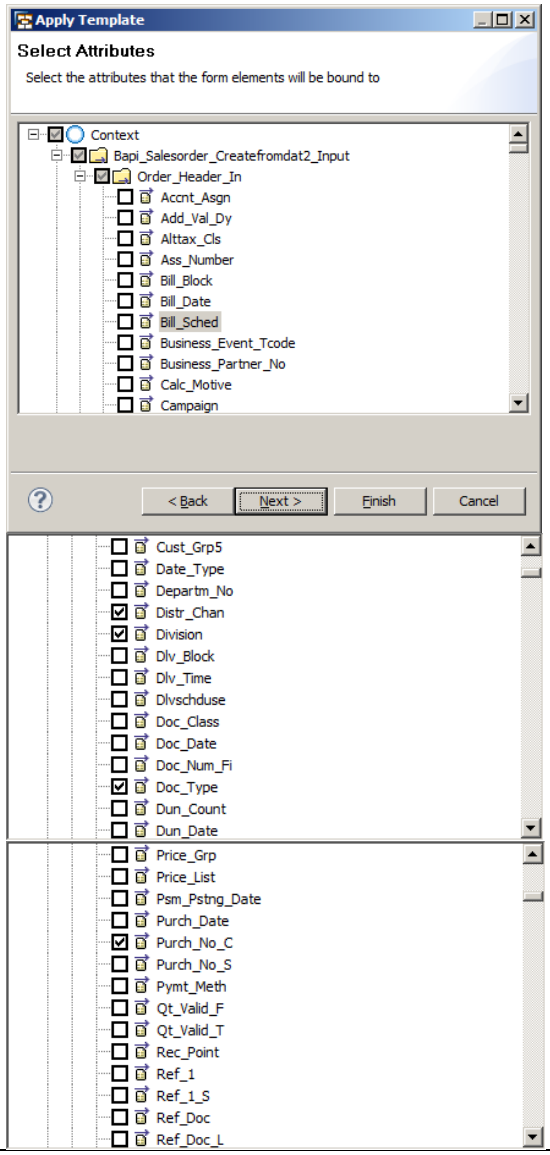
37.	Open Context tab of Sales Order View. Right click on Context, *New > Node.*	
38.	Choose Mapping. Click Next.	
39.	Choose *SOModelController* as required controller. Click OK.	

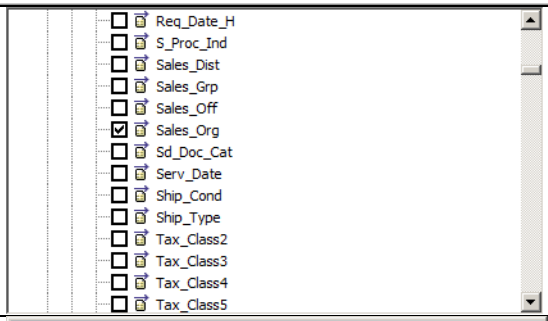
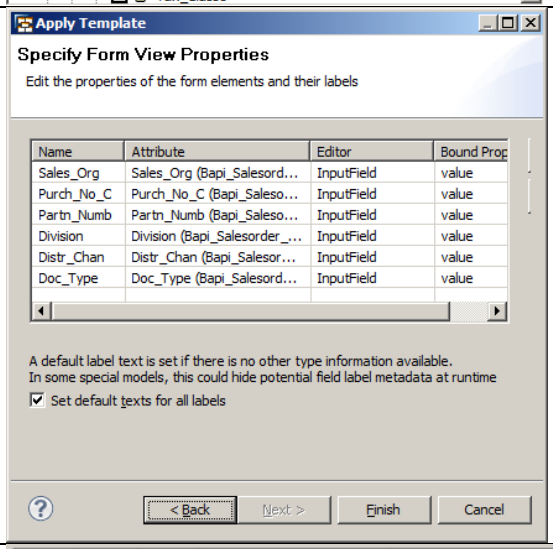
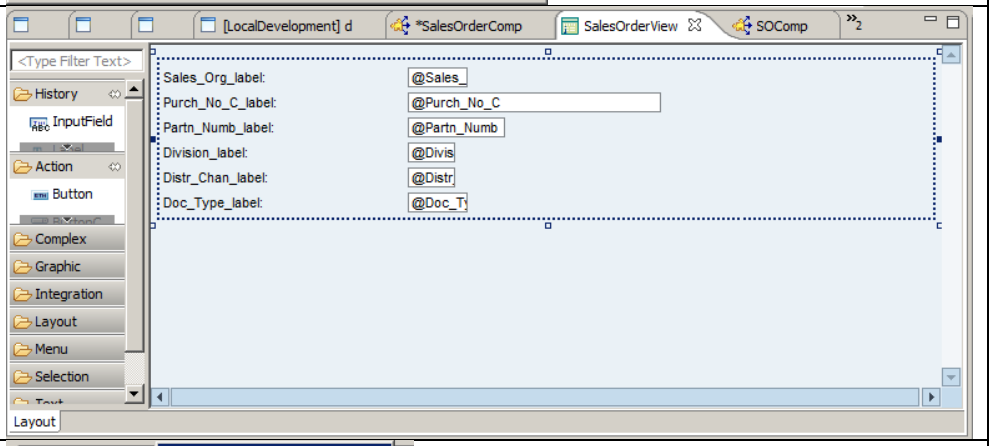
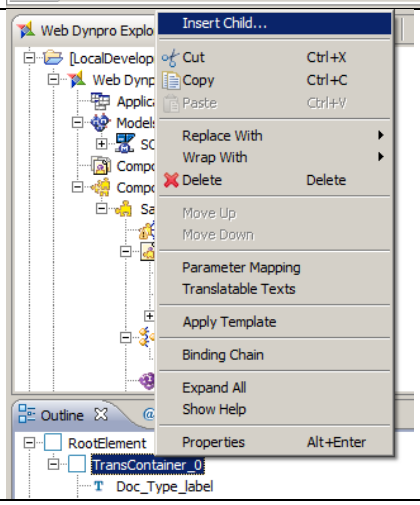
40.	Choose **Bapi_Salesorder_Createfromdat2_Input.**	
41.	Select the context elements to map. Click Finish.	
42.	Repeat the process for **Bapi_Transaction_Commit_Input.**	

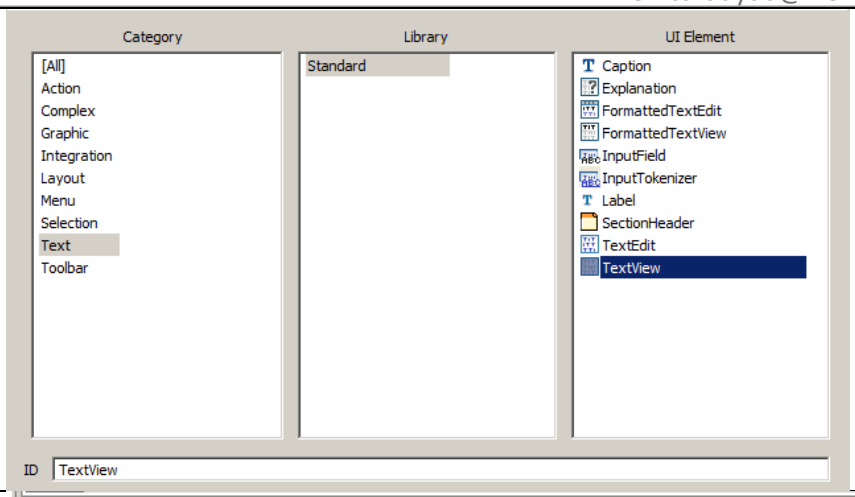
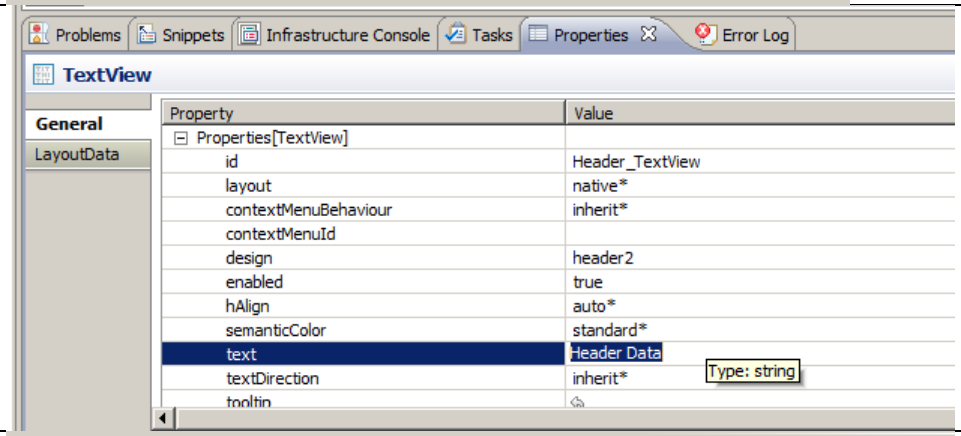
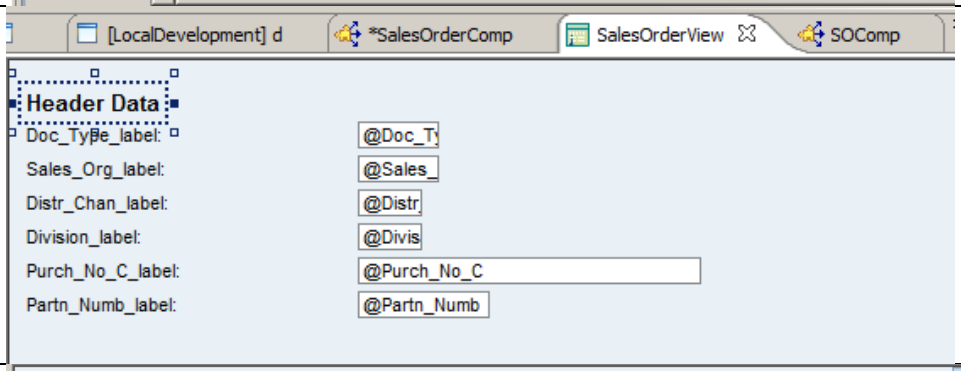
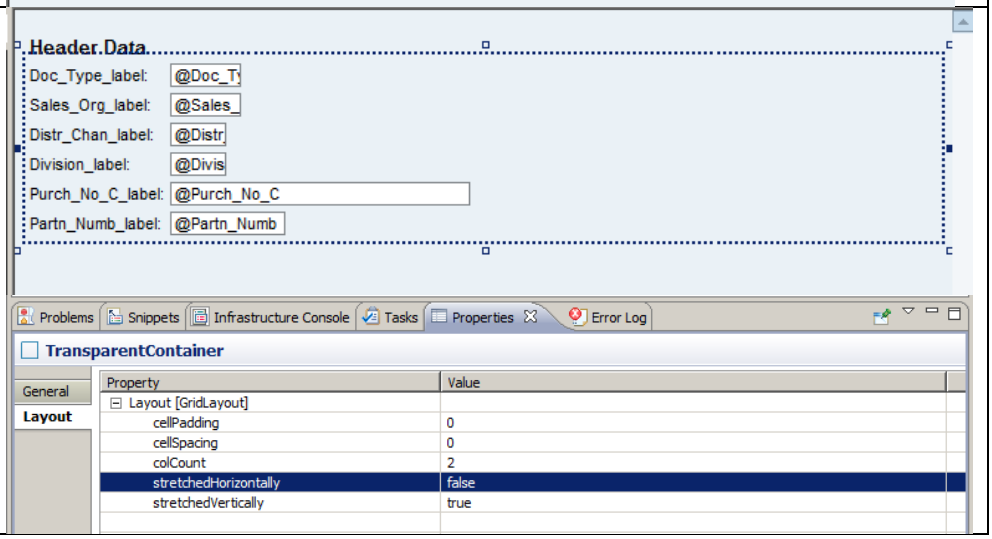
43.	Choose the node. As the result, context of SalesOrderView will contains two new nodes from context mapping to model.	

STEP 9. CREATE UI WITH APPLY TEMPLATE

44.	Right click on *SalesOrderView* > *Apply* Template.	

45.	Choose Form. Click Next.	
46.	Here we will create UI for header data sales order: - **Document Type** - **Sales Organization** - **Distribution Channel** - **Division** - **Purchase Order No.** Choose from Bapi_Salesorder_Createfromdat2_Input > Order_Header_In. Thick on the appropriate element from the context appear. Click Next.	

		
47.	Click Finish.	
48.	Form with labels and text fields will automatically create based on context. The value of text fields also bound into context.	
49.	We need little additional efforts to complete the UI. Here we will insert text field for title. On the Outline section of SalesOrderView right click on the header data's container > Insert Child.	

50.	Choose *Text* > *Text View.*	
51.	A new child will be generated. Rename the id property with "Header_TextView and text property to "Header Data".	
52.	The result will show like this.	
53.	We can make change to the attribute of the property. For the example I change the Layout property to not to be able to stretch horizontally. Thus the property is set to "false".	

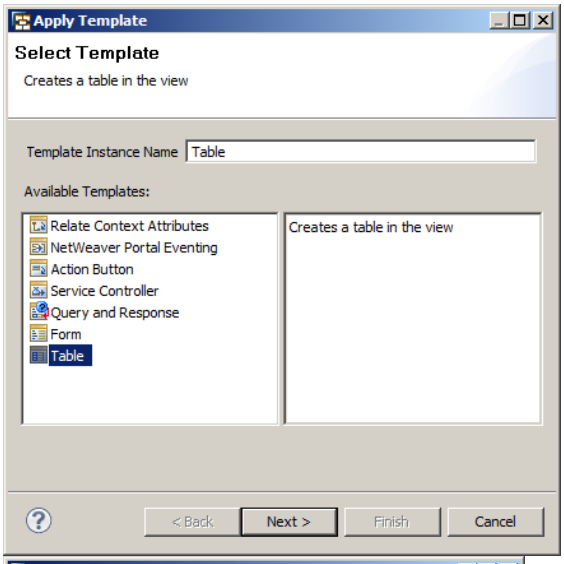
54. Continue to create another UI, table for store item data. Now we choose Order_Items_In to be applied into template.

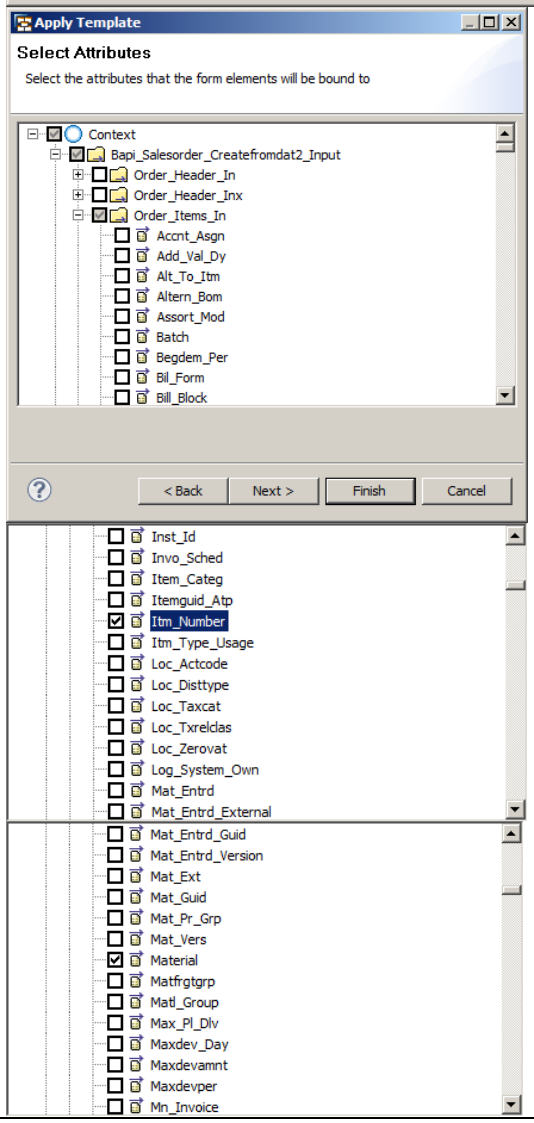
Right click on *SalesOrderView > Apply Template.*

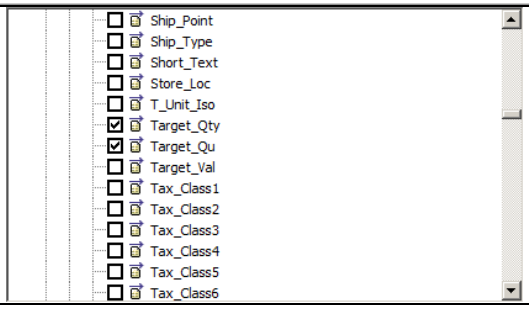
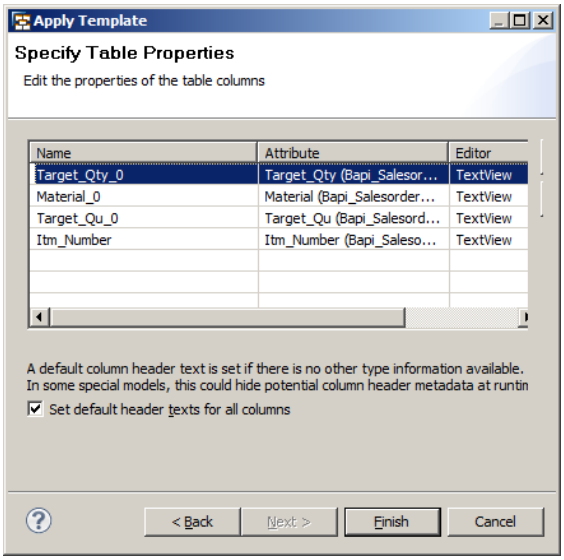
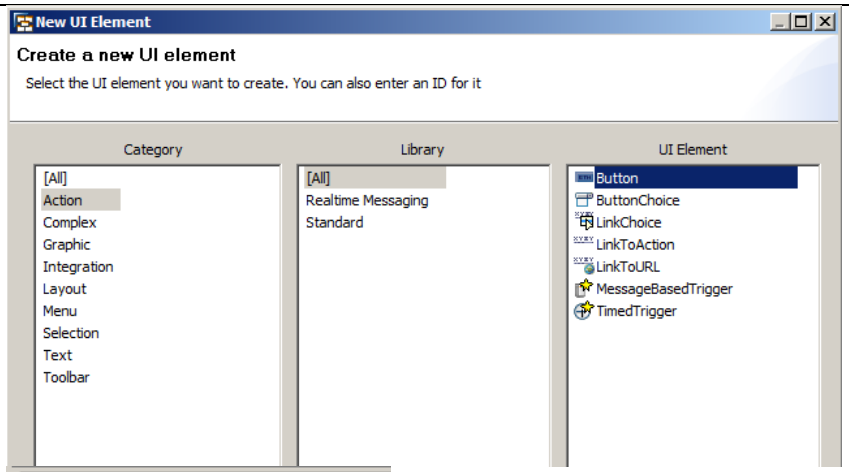
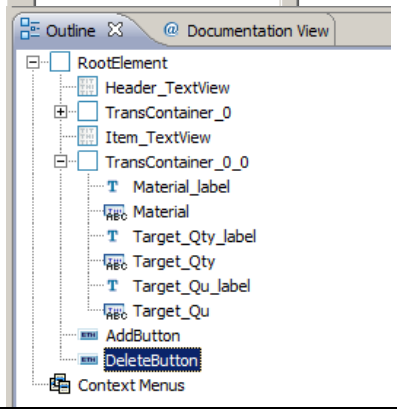
Choose Table. Click Next.

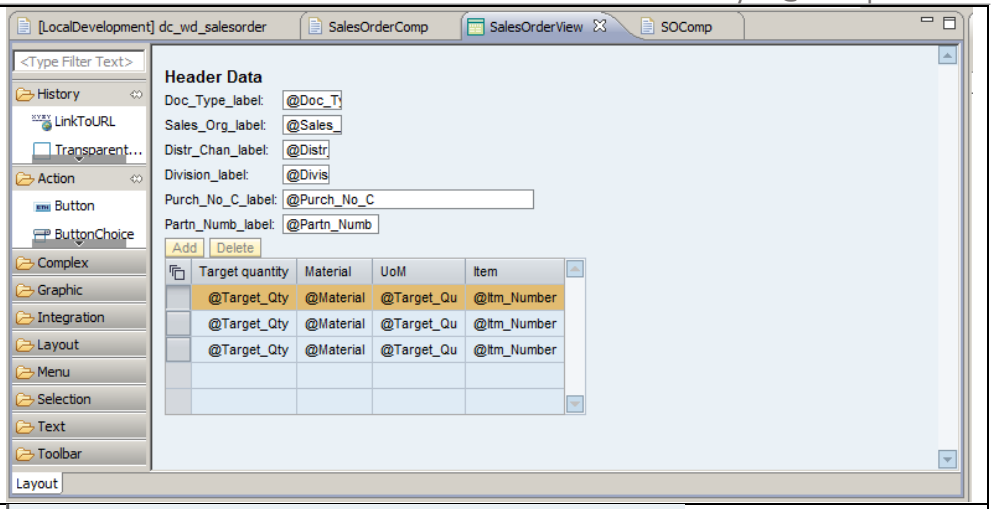
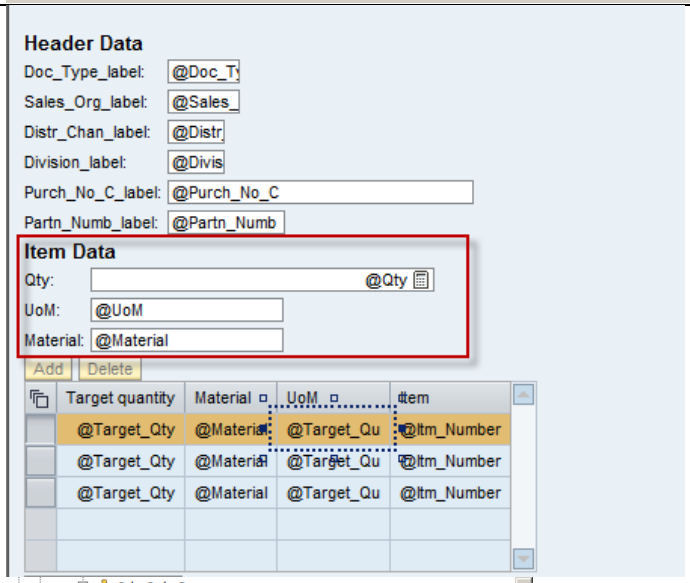
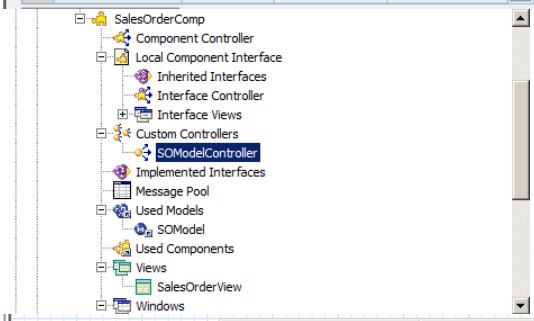
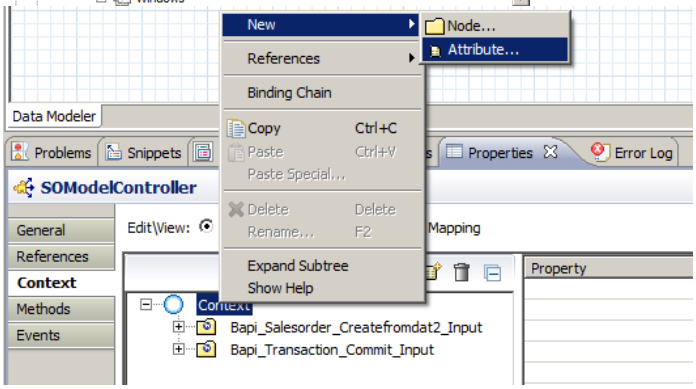
Choose Order_Items_In:
- Itm_Number
- Material
- Target_Qty
- Target_Qu

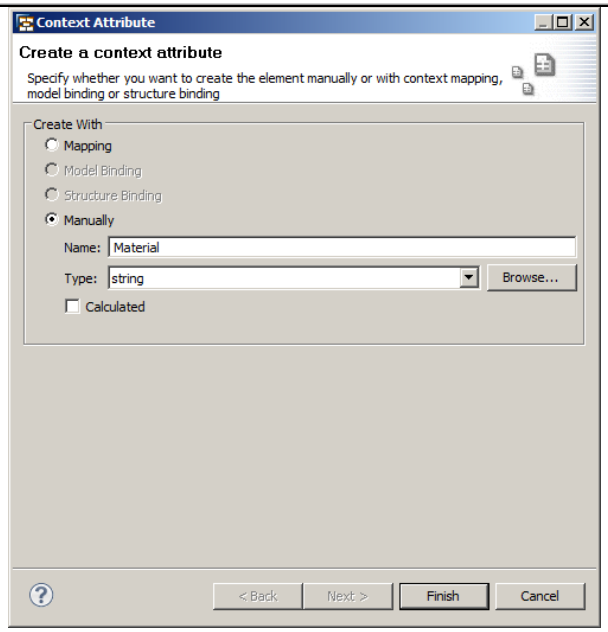
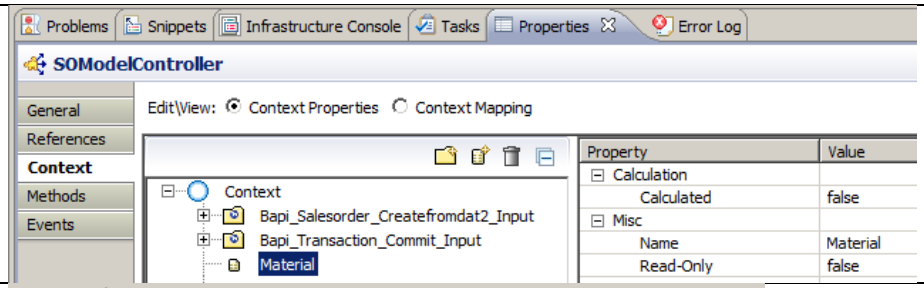
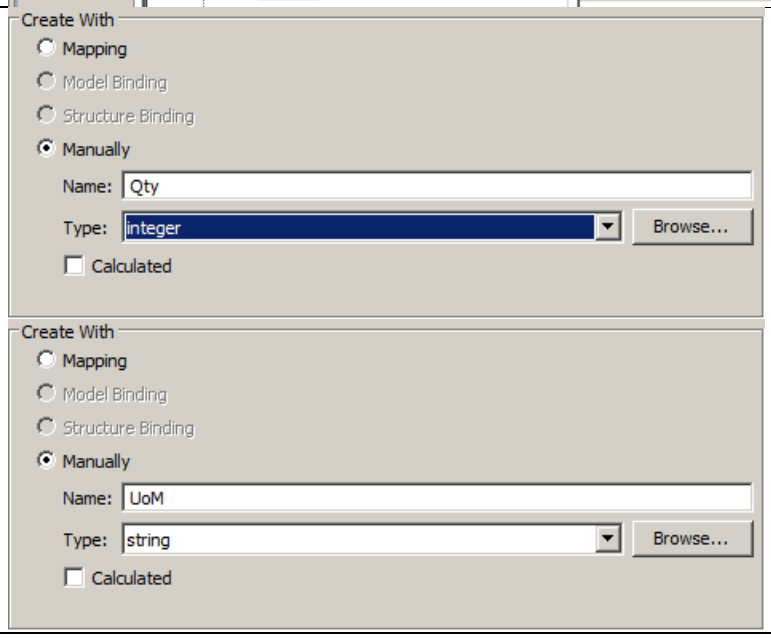
Click Finish.

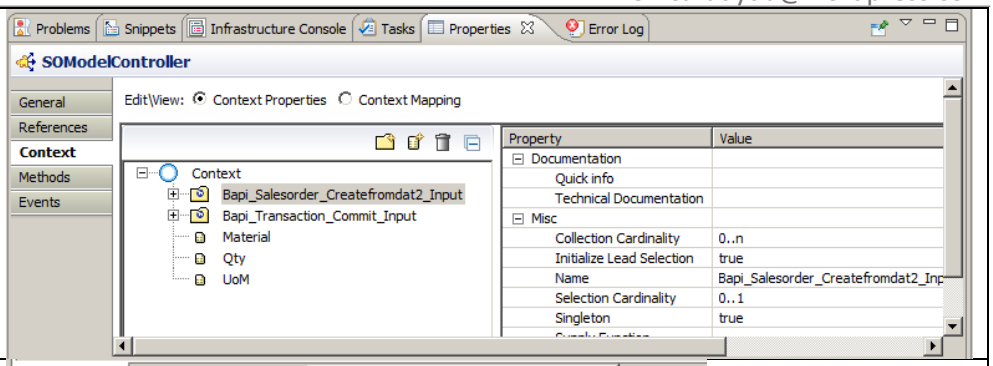
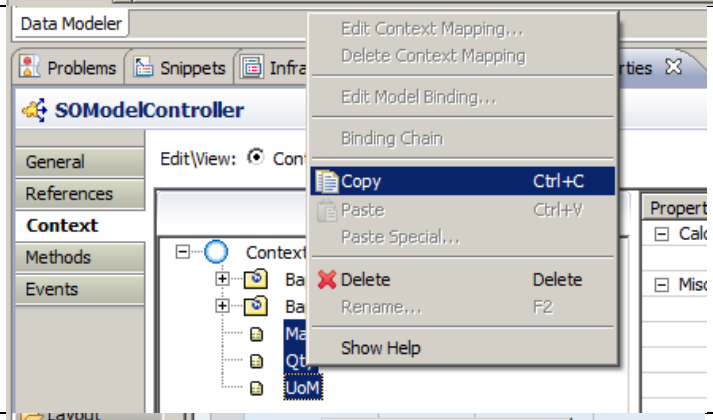
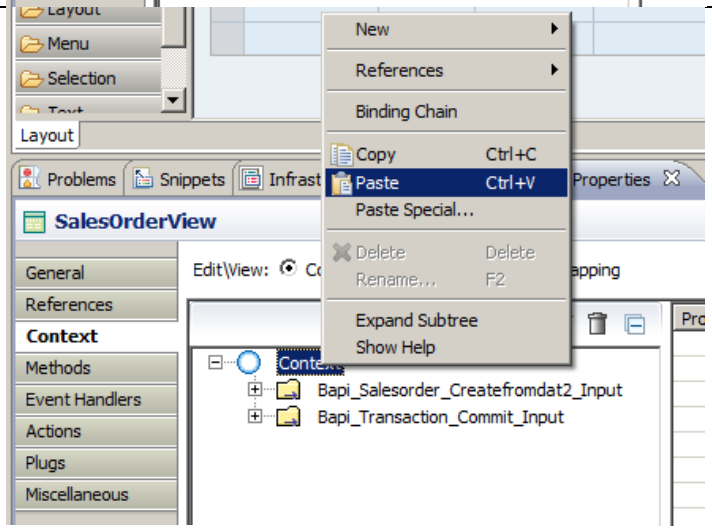
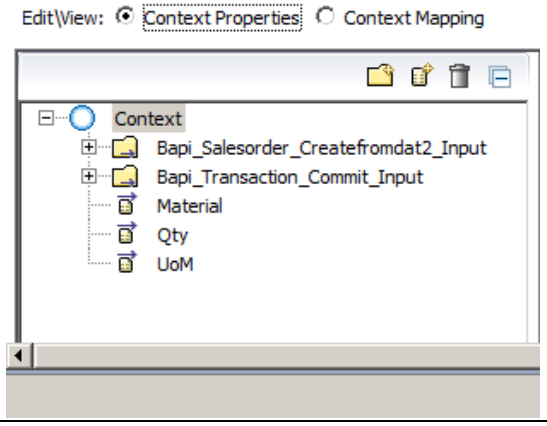


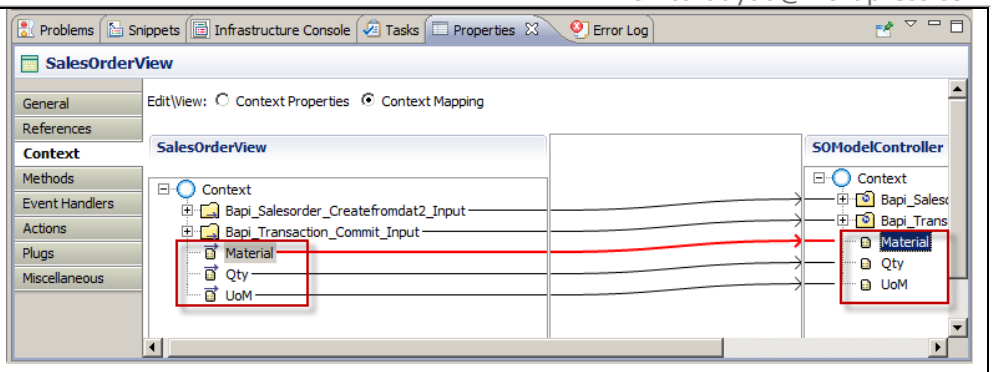
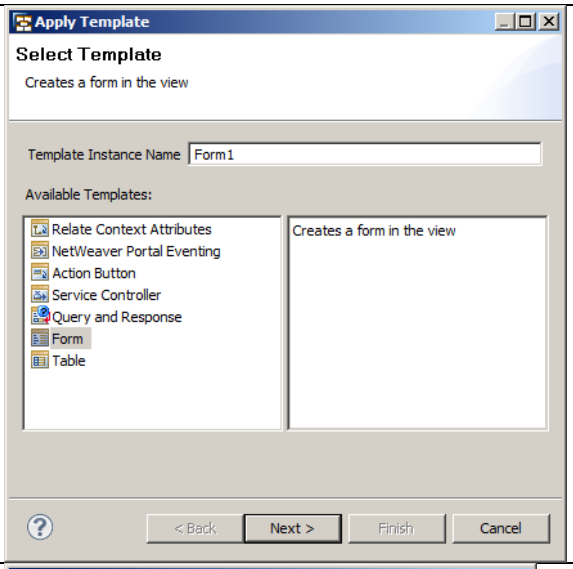
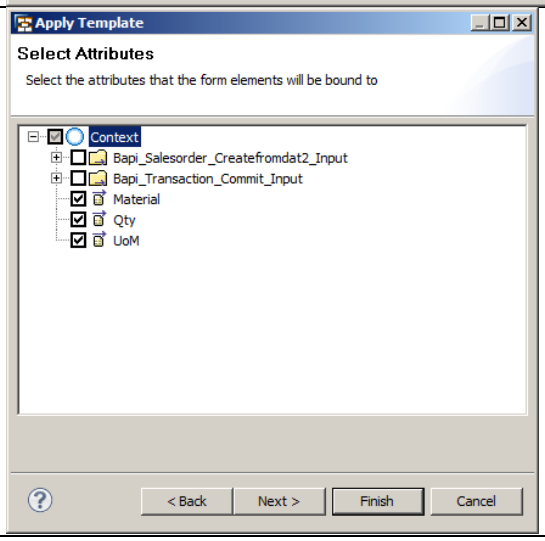


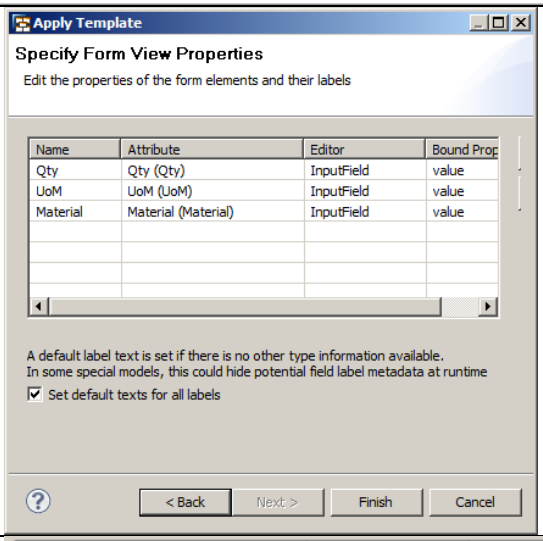
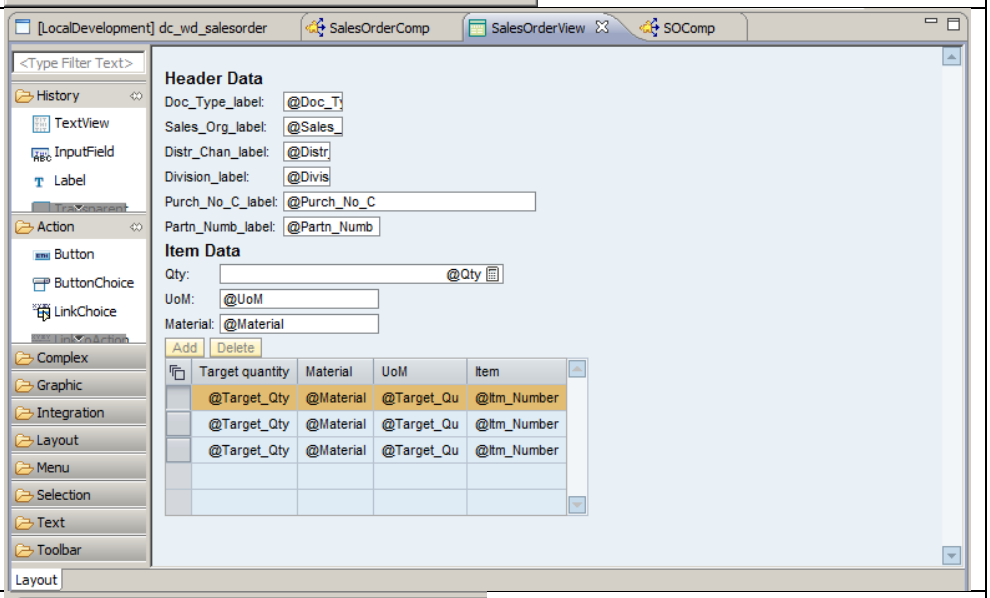
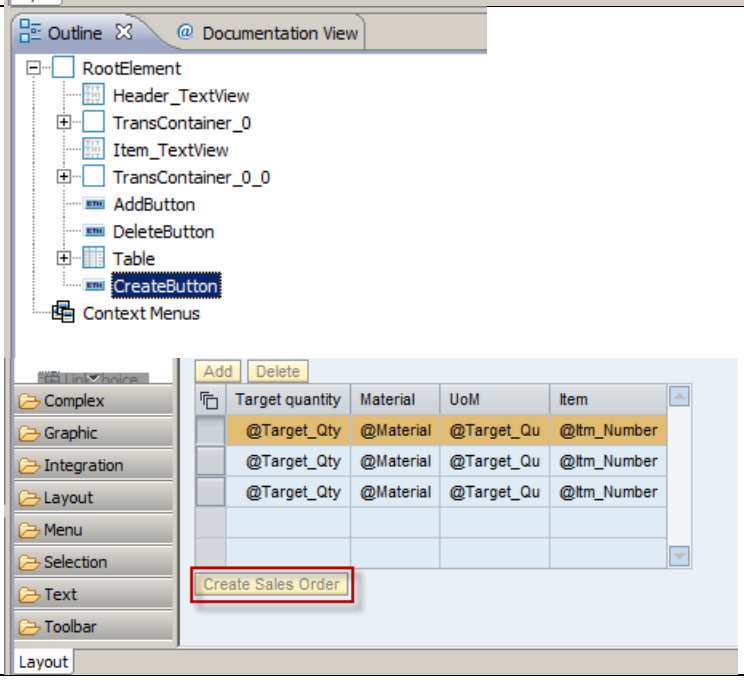
		
55.	Click Finish.	
56.	Continue with create two additional button: “Add” and “Delete”. On Outline, right click on item’s container > *Insert New Child.* Choose *Action* > *Button.* Rename the default name identifier with “Add”. Then copy this button to be pasted and renamed it to “Delete”.	 

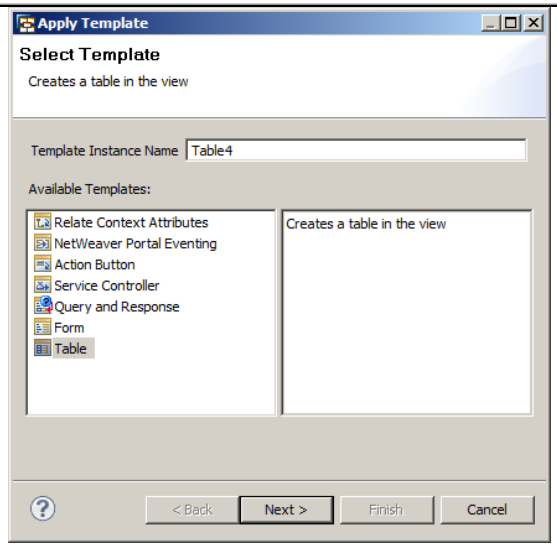
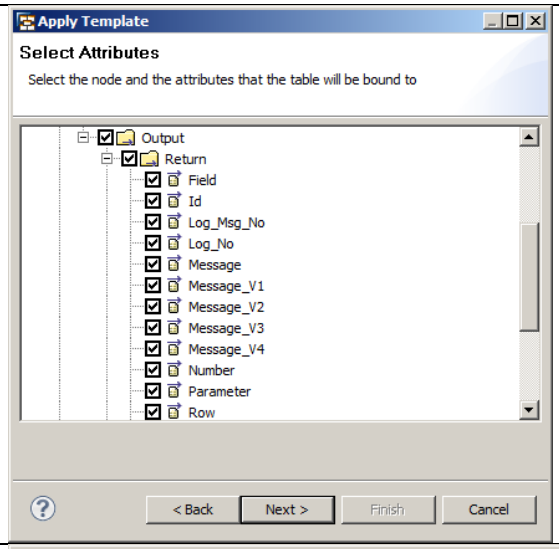
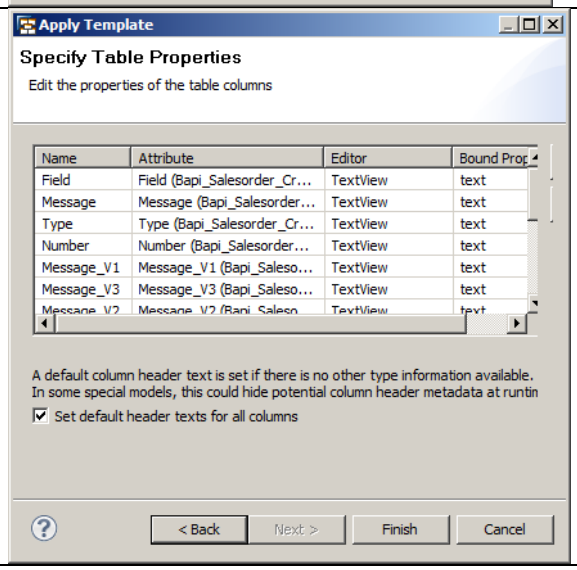
57.	Arrange the order of the layout as appropriate thus the result will come like screenshot beside.	
58.	We need to create multiple items in item data that will display in table first before bound into context and send to BAPI. Thus, we need to create additional contexts to temporary stores the data before send into BAPI. Double click on SOModelController. On Context tab, right click on *Context > New > Attribute.*	  

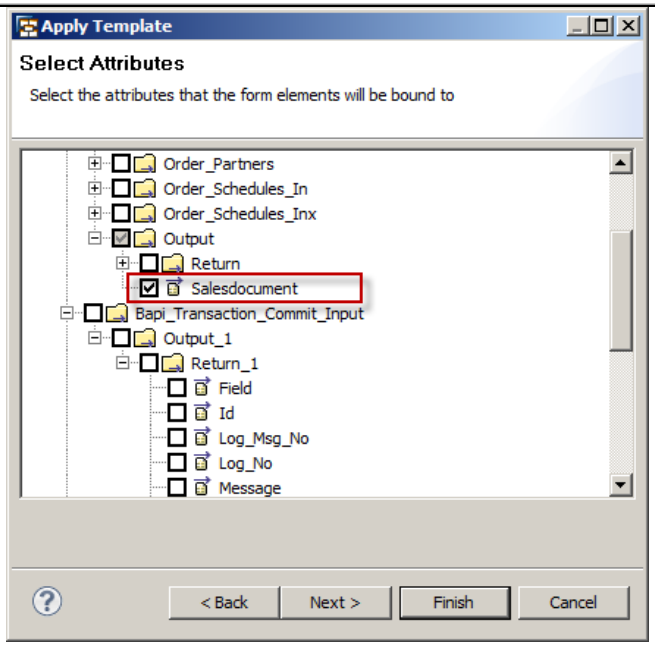
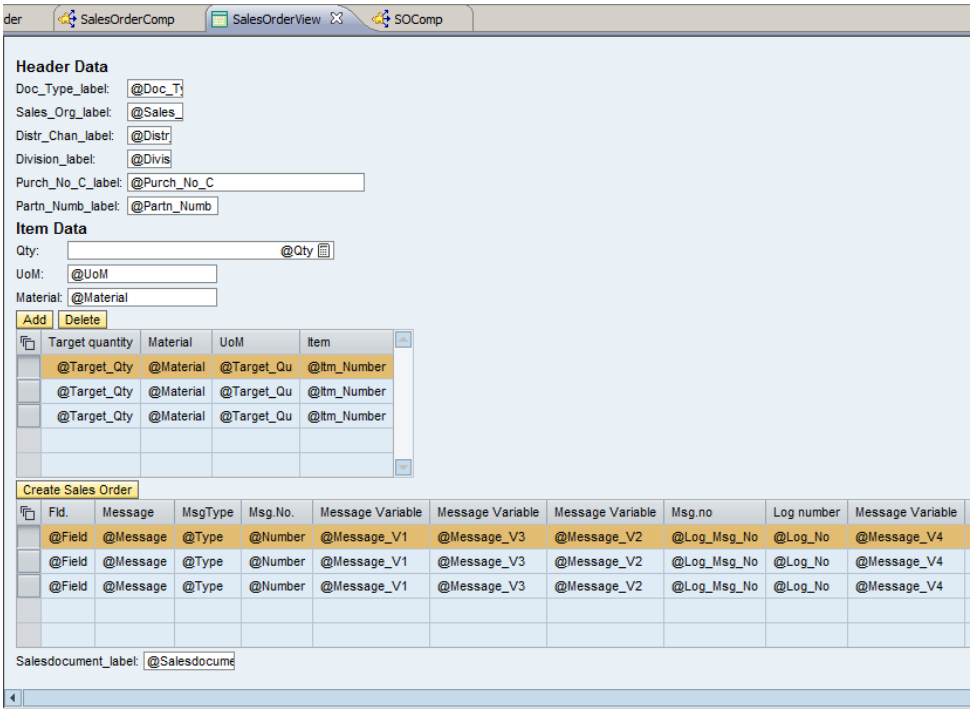
59.	Chooose Manually. Enter: Name = Material. Type = string. Click Finish.	
60.	New attribute will appear in context.	
61.	Repeat the process to create manually attribute to create two another attributes for quantity and UoM. Attributes: Name = Qty Type = integer Name = UoM Type = string	

		
62.	Choose the three attributes by left click on each attribute by pressing shift keyboard. Then right click > Copy.	
63.	Open the properties of SalesOrderView, and paste this three attributes into context. Thus, the context will have three new attributes copy from custom controller.	 

64.	Create context mapping. Open Context tab, choose Context Mapping. Drag each the new attribute from SalesOrderView to SOModelController to create context mapping.	
65.	After finished mapping the context to custom controller, then you can create another UI for item data. Right click on *SalesOrder View > Apply Template.* Choose Form. Click Next.	
66.	Choose Material, Qty, and UoM. Click Next.	

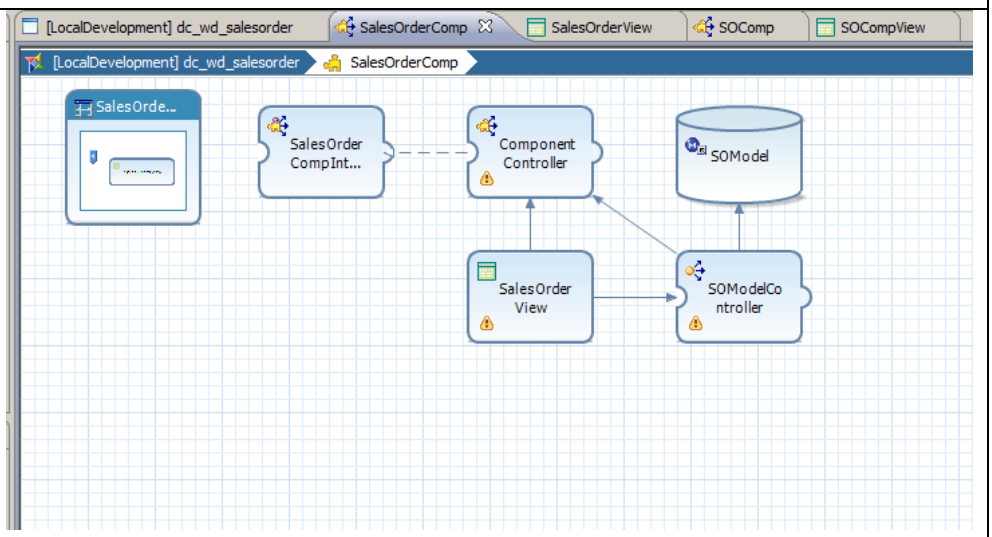
67.	Click Finish.	
68.	Do some order of the arrangement of UIs thus the final result will show like this.	
69.	To trigger action to call BAPI_SALES_CREATEFROMDAT2 we need a button that will generate event. Create a button and rename it to “Create Sales Order”. Position the button below the Item Data table.	

70.	Last, create a table for display log of creation of sales order. Right click on SalesOrderView > Apply Template. Click Next.	
71.	Choose from node **Bapi_Salesorder_Createfromdat2 > Output > Return.** Click Next.	
72.	Click Finish. SAVE.	

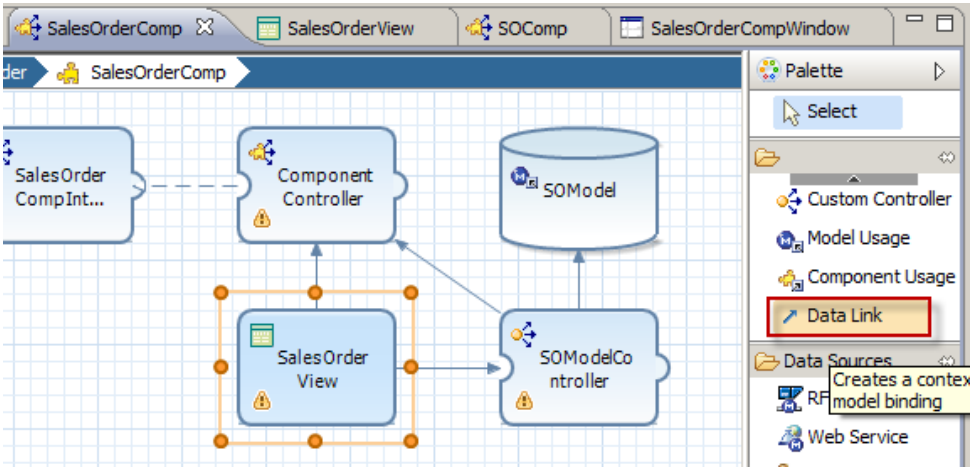
73.	Last, add a label and an input field to display *Sales Document Number*. Use apply template and bound from node Output > Salesdocument. The final result of our layout.	 

STEP 10. CREATE LINK TO COMPONENT CONTROLLER

74. In this step we will create link of our custom controller and view to component. Thus SOModelController will have data link to Component Controller and SalesOrderView too.



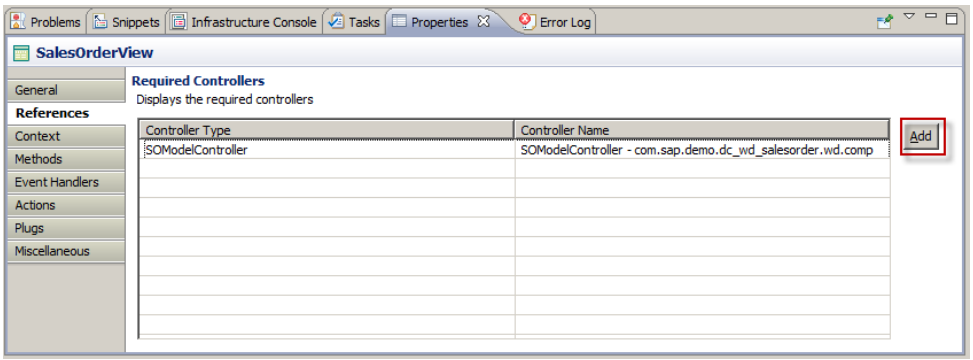
The first way is by drag and drop data link. Click on SalesOrderView, then from Palette appear choose Data Link. If cursor already change Click SalesOrderView to ComponentController.

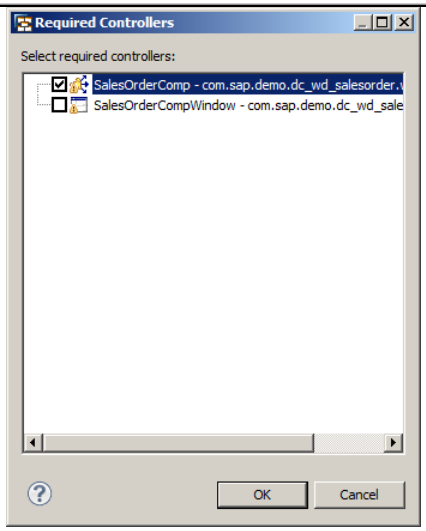
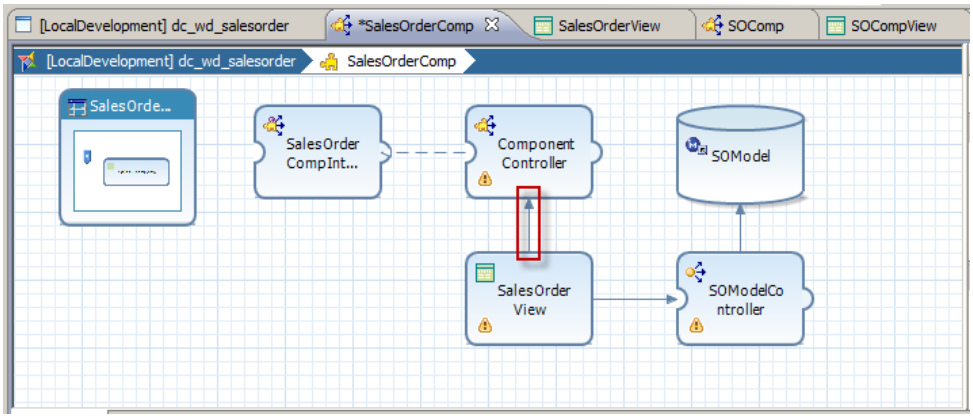
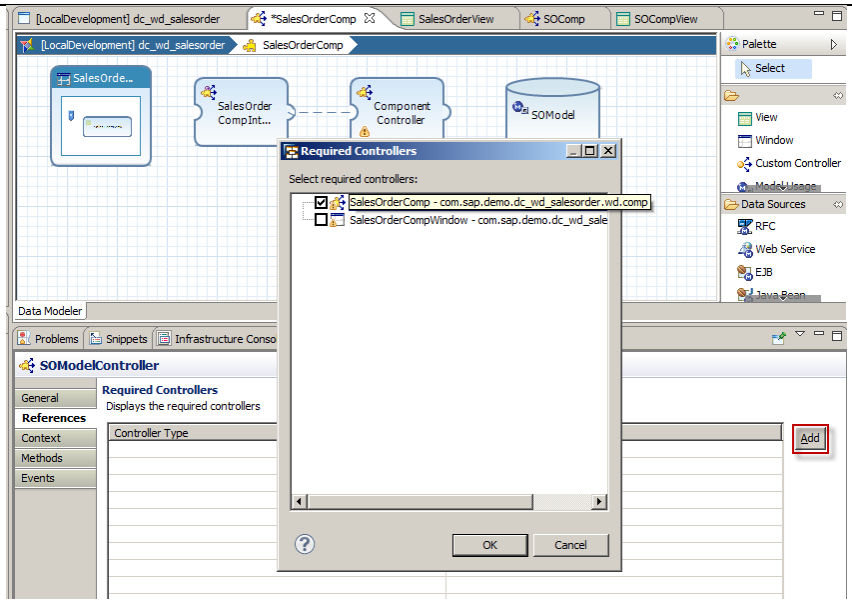


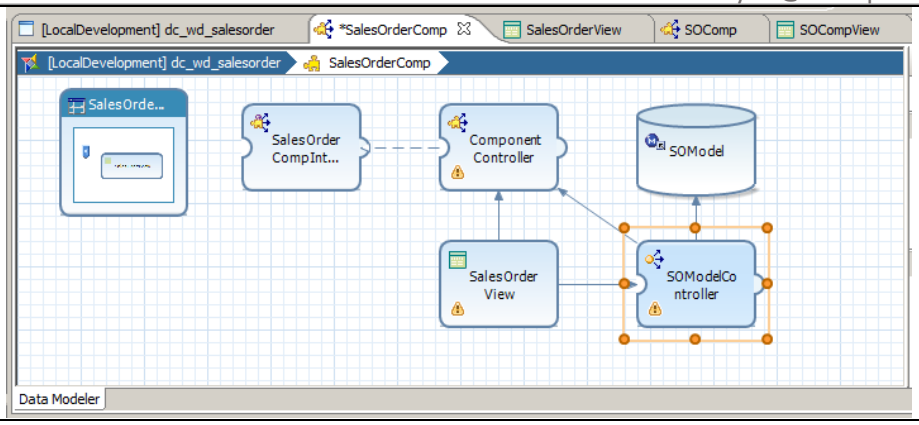
Or

The another method by accessing through *References* in SalesOrderView's properties.

Click Add button.



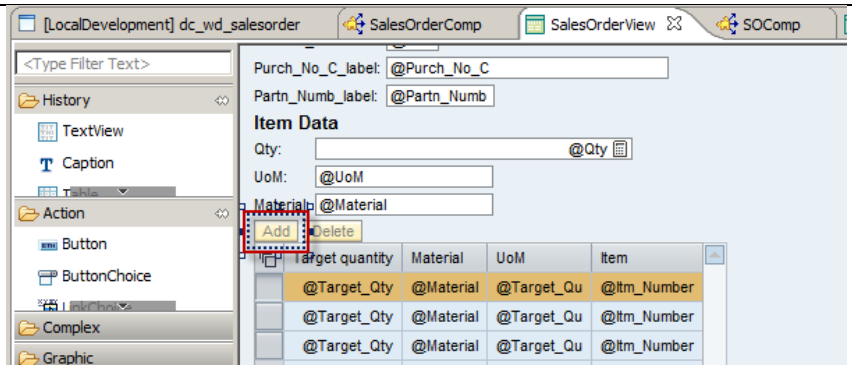
	On pop up window appear, choose SalesOrderComp. Click OK.	 Result: 
75.	Repeat the same way to SOModelController. SAVE.	



STEP 11. ADDING EVENT TO BUTTONS

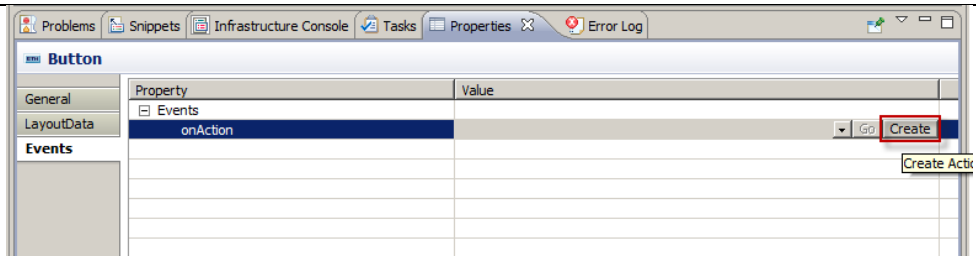
76. From the step before we only created layout and the buttons still not be able to work. Here we will add some java code to make these button do their job.

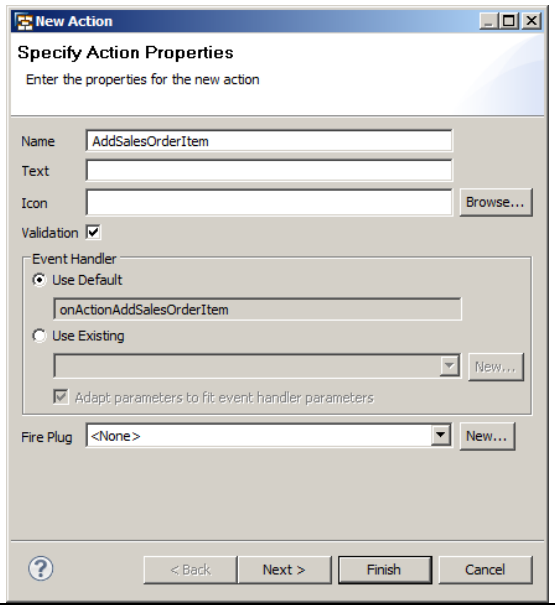
First, open the layout we have, click on "Add" button.

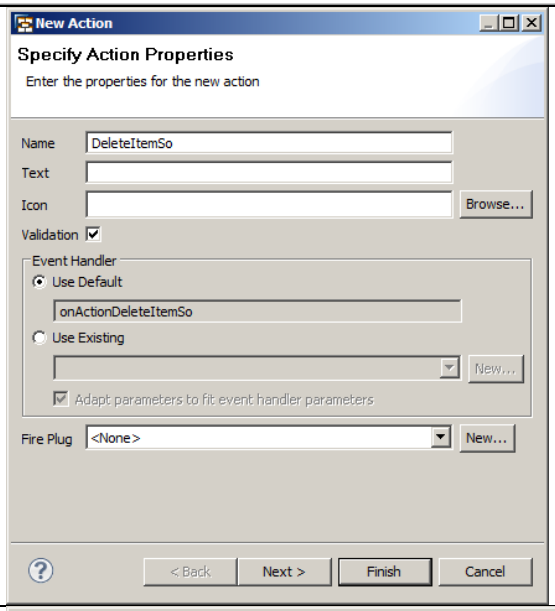
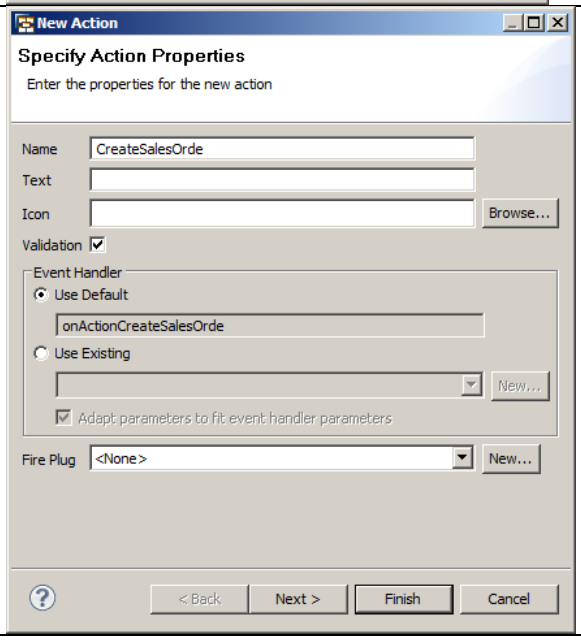


77. After properties window of Add button appeared, click on Events tab and create new action.

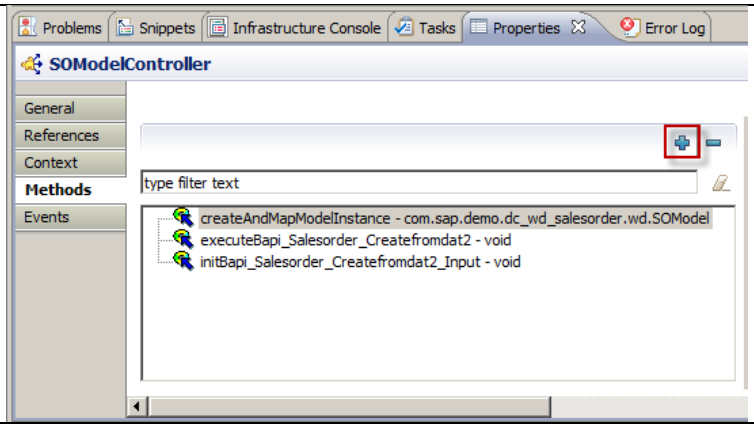
Type "AddSalesOrderItem" on field Name. It automatically create method that will we use later.
Click Finish.

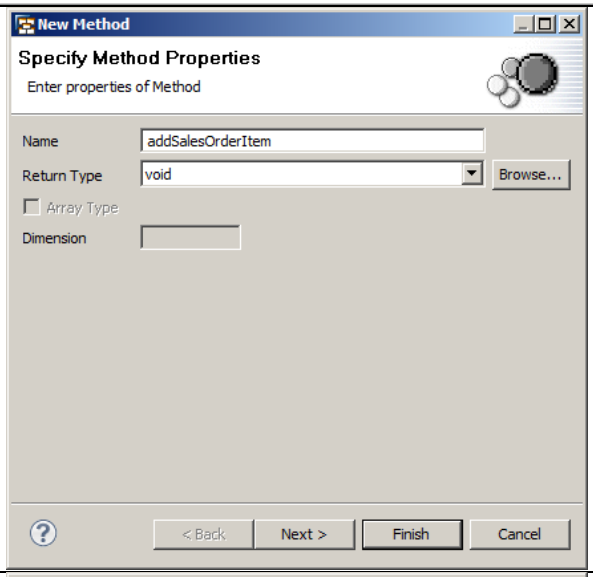
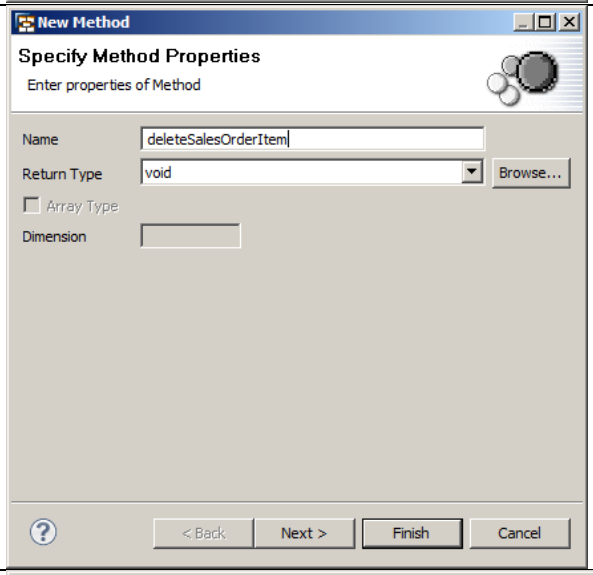
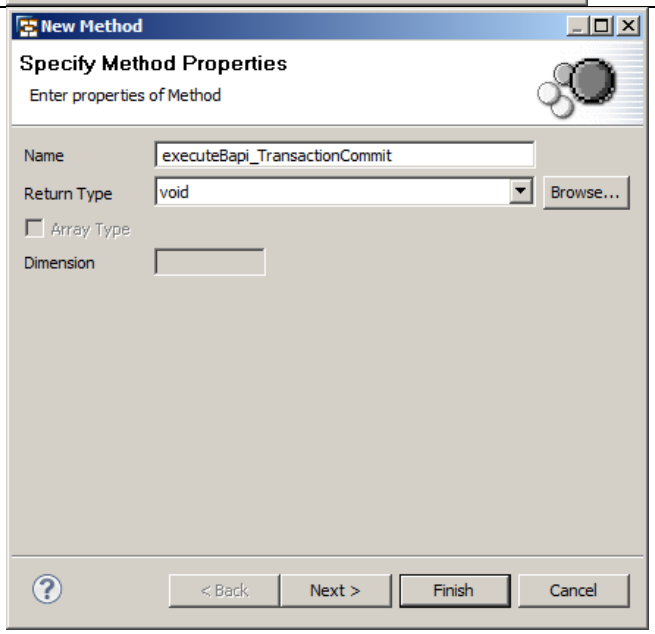


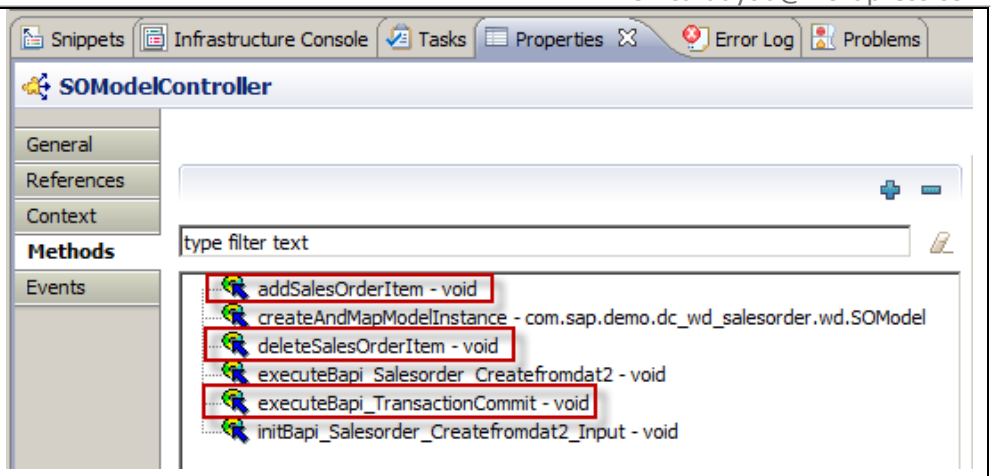


78.	Continue the same process for “Delete” button.	
79.	Last, for “Create Sales Order” button too.	

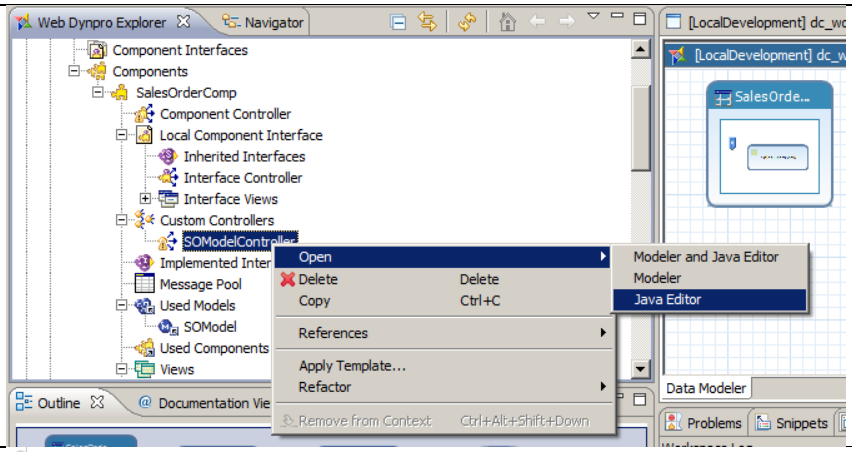
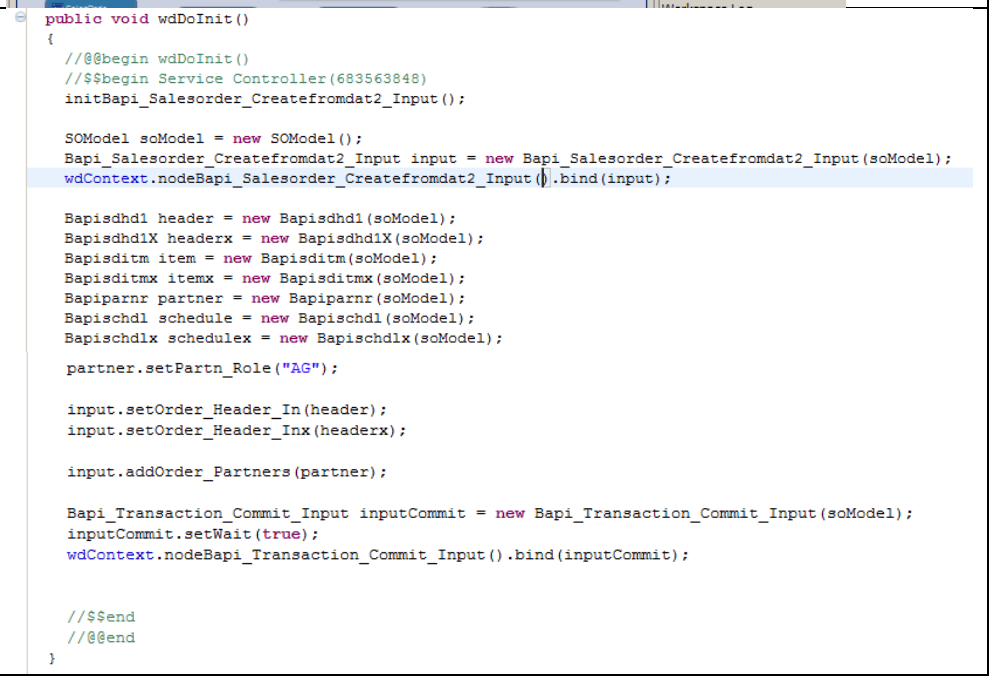
STEP 11. CREATE METHOD IN CUSTOM CONTROLLER

80.	Open custom controller, on Methods tab, click on plus sign. Here we will add method that will be used to add java code.	

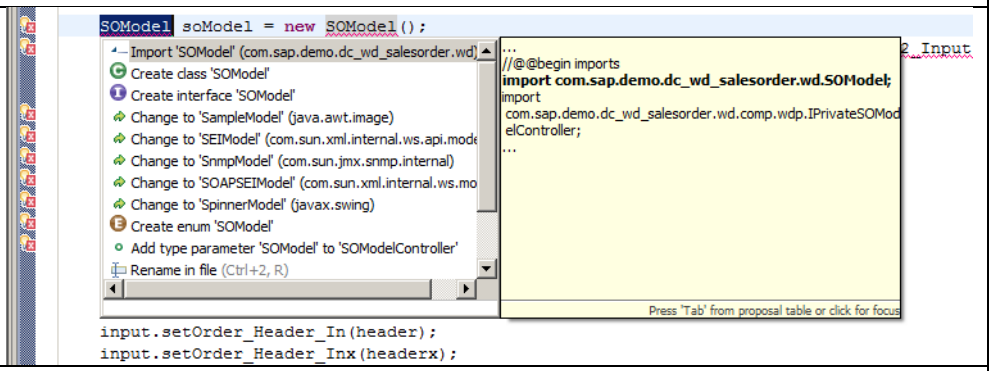
81.	Type “**addSalesOrderItem**”. This method will we use to write some logic to “Add” button. Click Finish.	
82.	Add another method “**deleteSalesOrderItem**”.	
83	Since we not yet add method for **Bapi_Transaction_Commit**. Now, add a method in custom controller for this BAPI. Click Finish.	

84.	For the three result, we now have three new methods in our custom controller. We will add some logic of this method later.	

STEP 12. IMPLEMENT JAVA CODE IN CUSTOM CONTROLLER

85.	After we add methods that we need to write some logic to be applied on our buttons we created before, now is time to implement the logic. Right click on *SOModelController > Open > Java Editor.*	
86.	After java code open, find method **wdDoInit()** and add some java code. In this method we will make an initialization when the first time java program is called and run.	

```
public void wdDoInit()
  {
    //@@begin wdDoInit()
    //$$begin Service Controller(683563848)
    initBapi_Salesorder_Createfromdat2_Input();

    SOModel soModel = new SOModel();
    Bapi_Salesorder_Createfromdat2_Input input = new
Bapi_Salesorder_Createfromdat2_Input(soModel);


wdContext.nodeBapi_Salesorder_Createfromdat2_Input().bin
d(input);

    Bapisdhd1 header = new Bapisdhd1(soModel);
    Bapisdhd1X headerx = new Bapisdhd1X(soModel);
    Bapisditm item = new Bapisditm(soModel);
    Bapisditmx itemx = new Bapisditmx(soModel);
    Bapiparnr partner = new Bapiparnr(soModel);
    Bapischdl schedule = new Bapischdl(soModel);
    Bapischdlx schedulex = new Bapischdlx(soModel);

    partner.setPartn_Role("AG");

    input.setOrder_Header_In(header);
    input.setOrder_Header_Inx(headerx);

    input.addOrder_Partners(partner);

    Bapi_Transaction_Commit_Input inputCommit = new
Bapi_Transaction_Commit_Input(soModel);
    inputCommit.setWait(true);

      wdContext.nodeBapi_Transaction_Commit_Input().bind
      (inputCommit);


    //$$end
    //@@end
}
```

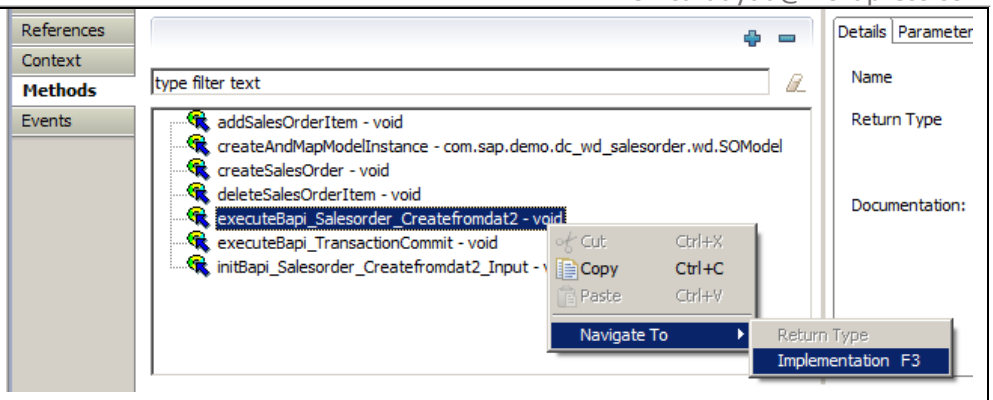
87.	If there is still appear error, click on bubble lamp that appear beside error and do appropriate import.	
88.	Finally SAVE.	

89. Now we will add logic to the methods we created before.

Double click on custom controller. After properties appear, open the Methods tab.

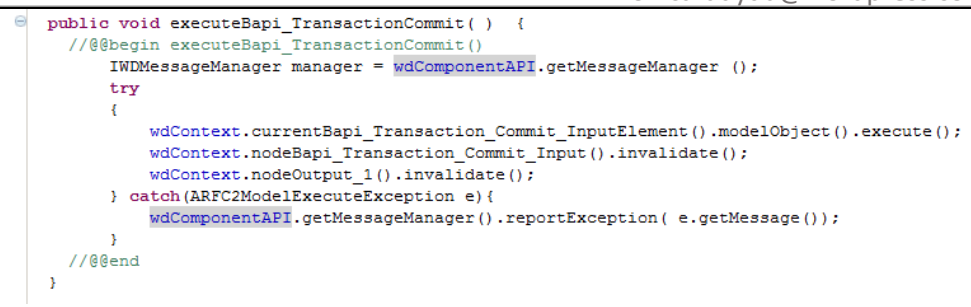
First right click on method **executeBapi_Salesorder_Createfromdat2()**, *Navigate To > Implementation.*

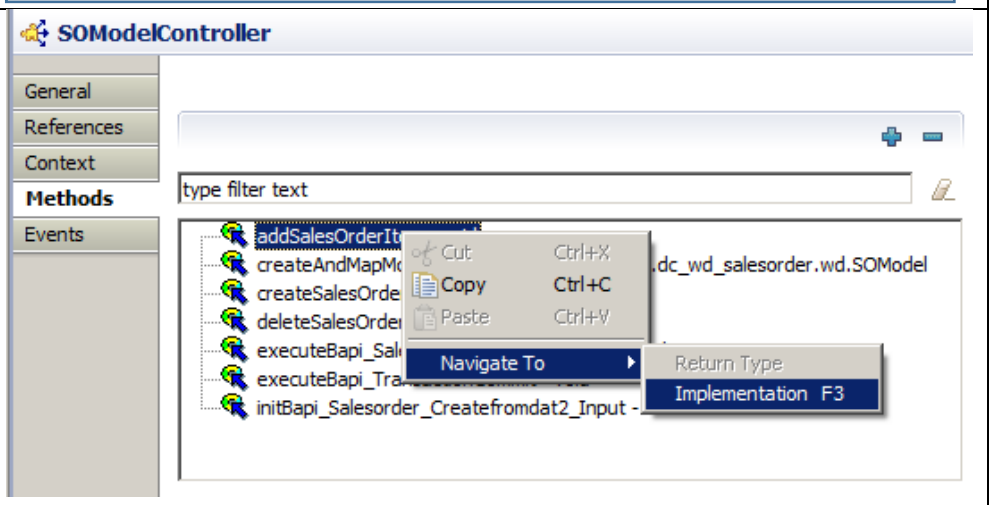
SAVE.




```
public void executeBapi_Salesorder_Createfromdat2( )  {
  //@@begin executeBapi_Salesorder_Createfromdat2()
  //$$begin Service Controller(-460758325)

  IWDMessageManager manager = wdComponentAPI.getMessageManager ();

  try
  {
      wdContext.currentBapi_Salesorder_Createfromdat2_InputElement().modelObject().execute();
      wdContext.nodeOutput().invalidate();

  } catch(ARFC2ModelExecuteException e){
      wdComponentAPI.getMessageManager().reportException( e.getMessage());
  }
  //$$end
  //@@end
}
```

```
  public void executeBapi_Salesorder_Createfromdat2( )
{
    //@@begin executeBapi_Salesorder_Createfromdat2()
    //$$begin Service Controller(-460758325)

      IWDMessageManager manager =
wdComponentAPI.getMessageManager ();

    try
    {

      wdContext.currentBapi_Salesorder_Createfromdat2_In
putElement().modelObject().execute();
      wdContext.nodeOutput().invalidate();

    } catch(ARFC2ModelExecuteException e){

      wdComponentAPI.getMessageManager().reportException
( e.getMessage());
    }
    //$$end
    //@@end
  }
```

90.	Find implementation of method **executeBapi_TransactionCommit()** and add some additional java code.
	
	SAVE.
		
		`public void executeBapi_TransactionCommit( ) {`
		`//@@begin executeBapi_TransactionCommit()`
		`IWDMessageManager manager = wdComponentAPI.getMessageManager ();`
		`try`
		`{`
		`wdContext.currentBapi_Transaction_Commit_InputElement().modelObject().execute();`
		`wdContext.nodeBapi_Transaction_Commit_Input().invalidate();`
		`wdContext.nodeOutput_1().invalidate();`
		`} catch(ARFC2ModelExecuteException e){`
		`wdComponentAPI.getMessageManager().reportException( e.getMessage());`
		`}`
		`//@@end`
		`}`
		
		`public void executeBapi_TransactionCommit( ) {`
		`//@@begin executeBapi_TransactionCommit()`
		`IWDMessageManager manager = wdComponentAPI.getMessageManager ();`
		
		`try`
		`{`
		
		`wdContext.currentBapi_Transaction_Commit_InputElement().modelObject().execute();`
		
		`wdContext.nodeBapi_Transaction_Commit_Input().invalidate();`
		`wdContext.nodeOutput_1().invalidate();`
		
		`} catch(ARFC2ModelExecuteException e){`
		
		`wdComponentAPI.getMessageManager().reportException( e.getMessage());`
		`}`
		`//@@end`
		`}`

91.	Now, create the implementation code for method **addSalesOrderItem()**. SAVE.	 SOModelController General References Context Methods Events type filter text addSalesOrderIt createAndMapM createSalesOrde deleteSalesOrde executeBapi_Sal executeBapi_Tra initBapi_Salesorder_Createfromdat2_Input - Cut Ctrl+X Copy Ctrl+C Paste Ctrl+V Navigate To Return Type Implementation F3 dc_wd_salesorder.wd.SOModel

```
public void addSalesOrderItem( )  {
  //@@begin addSalesOrderItem()
    SOModel soModel = new SOModel();

    Bapisdhd1 header = new Bapisdhd1(soModel);
    Bapisdhd1X headerx = new Bapisdhd1X(soModel);
    Bapisditm orderItem = new Bapisditm(soModel);
    Bapisditmx orderItemx = new Bapisditmx(soModel);
    Bapiparnr partner = new Bapiparnr(soModel);
    Bapischdl schedule = new Bapischdl(soModel);
    Bapischdlx schedulex = new Bapischdlx(soModel);

    IPublicSOModelController.IOrder_Header_InElement eleh = wdContext.nodeOrder_Header_In().createOrder_Header_InElement(header);
    IPublicSOModelController.IOrder_Header_InxElement elehx = wdContext.nodeOrder_Header_Inx().createOrder_Header_InxElement(headerx);
    IPublicSOModelController.IOrder_Items_InElement elei = wdContext.nodeOrder_Items_In().createOrder_Items_InElement(orderItem);
    IPublicSOModelController.IOrder_Items_InxElement eleix = wdContext.nodeOrder_Items_Inx().createOrder_Items_InxElement(orderItemx);
    IPublicSOModelController.IOrder_PartnersElement elep = wdContext.nodeOrder_Partners().createOrder_PartnersElement(partner);
    IPublicSOModelController.IOrder_Schedules_InElement eles = wdContext.nodeOrder_Schedules_In().createOrder_Schedules_InElement(schedule);
    IPublicSOModelController.IOrder_Schedules_InxElement elesx = wdContext.nodeOrder_Schedules_Inx().createOrder_Schedules_InxElement(schedulex);

    String material = wdContext.currentContextElement().getMaterial();
    int qty = wdContext.currentContextElement().getQty();
    BigDecimal targetQty = new BigDecimal(qty);
    String uom = wdContext.currentContextElement().getUoM();
    int counterItem = wdContext.nodeOrder_Items_In().size();

    int item = 0;
    if (counterItem == 0){
        item = 10;
    }else{
        item = (counterItem * 10) + 10;
    }

    String itemNumber2 = "0000" + item;

    elei.setItm_Number(itemNumber2);
    elei.setMaterial(material);
    elei.setTarget_Qty(targetQty);
    elei.setTarget_Qu(uom);

    eleix.setUpdateflag("X");
    eleix.setItm_Number(itemNumber2);
    eleix.setMaterial(true);
    eleix.setTarget_Qty(true);
    eleix.setTarget_Qu(true);

    eles.setItm_Number(itemNumber2);
    eles.setReq_Qty(targetQty);

    elesx.setUpdateflag("X");
    elesx.setItm_Number("000010");
    elesx.setReq_Qty(true);


    wdContext.nodeOrder_Items_In().addElement(elei);
    wdContext.nodeOrder_Items_Inx().addElement(eleix);
    wdContext.nodeOrder_Schedules_In().addElement(eles);
    wdContext.nodeOrder_Schedules_Inx().addElement(elesx);

  //@@end
}
```

```
public void addSalesOrderItem( )  {
    //@@begin addSalesOrderItem()

SOModel soModel = new SOModel();

Bapisdhd1 header = new Bapisdhd1(soModel);
Bapisdhd1X headerx = new Bapisdhd1X(soModel);
Bapisditm orderItem = new Bapisditm(soModel);
Bapisditmx orderItemx = new Bapisditmx(soModel);
Bapiparnr partner = new Bapiparnr(soModel);
Bapischdl schedule = new Bapischdl(soModel);
Bapischdlx schedulex = new Bapischdlx(soModel);

IPublicSOModelController.IOrder_Header_InElement  eleh  =
wdContext.nodeOrder_Header_In().createOrder_Header_InEle
ment(header);

IPublicSOModelController.IOrder_Header_InxElement elehx =
wdContext.nodeOrder_Header_Inx().createOrder_Header_InxE
lement(headerx);
IPublicSOModelController.IOrder_Items_InElement  elei  =
wdContext.nodeOrder_Items_In().createOrder_Items_InEleme
nt(orderItem);
IPublicSOModelController.IOrder_Items_InxElement eleix =
wdContext.nodeOrder_Items_Inx().createOrder_Items_InxEle
ment(orderItemx);
IPublicSOModelController.IOrder_PartnersElement  elep  =
wdContext.nodeOrder_Partners().createOrder_PartnersEleme
nt(partner);

IPublicSOModelController.IOrder_Schedules_InElement  eles
=
wdContext.nodeOrder_Schedules_In().createOrder_Schedules
_InElement(schedule);

IPublicSOModelController.IOrder_Schedules_InxElement
elesx                                                  =
wdContext.nodeOrder_Schedules_Inx().createOrder_Schedule
s_InxElement(schedulex);
```

```
//Set item number automatically based on data that user
input

String material =
wdContext.currentContextElement().getMaterial();
int qty = wdContext.currentContextElement().getQty();
        BigDecimal targetQty = new BigDecimal(qty);
String uom = wdContext.currentContextElement().getUoM();

int counterItem = wdContext.nodeOrder_Items_In().size();

        int item = 0;
        if (counterItem == 0){
              item = 10;
        }else{
              item = (counterItem * 10) + 10;
        }

String itemNumber2 = "0000" + item;

elei.setItm_Number(itemNumber2);
elei.setMaterial(material);
elei.setTarget_Qty(targetQty);
elei.setTarget_Qu(uom);

eleix.setUpdateflag("X");
eleix.setItm_Number(itemNumber2);
eleix.setMaterial(true);
eleix.setTarget_Qty(true);
eleix.setTarget_Qu(true);

eles.setItm_Number(itemNumber2);
eles.setReq_Qty(targetQty);

elesx.setUpdateflag("X");
elesx.setItm_Number("000010");
elesx.setReq_Qty(true);

wdContext.nodeOrder_Items_In().addElement(elei);
wdContext.nodeOrder_Items_Inx().addElement(eleix);
wdContext.nodeOrder_Schedules_In().addElement(eles);
wdContext.nodeOrder_Schedules_Inx().addElement(elesx);

  //@@end
  }
```

92. Implement method **deleteSalesOrderItem()** .

SAVE.

```
public void deleteSalesOrderItem( )  {
  //@@begin deleteSalesOrderItem()
    int n = wdContext.nodeOrder_Items_In().size();
    int leadSelected = wdContext.nodeOrder_Items_In().getLeadSelection();

    for(int i = n - 1; i >= 0; --i){
        if (wdContext.nodeOrder_Items_In().isMultiSelected(i) || leadSelected == i){
            wdContext.nodeOrder_Items_In().removeElement(wdContext.nodeOrder_Items_In().getElementAt(i));
        }

    }
  //@@end
}
```

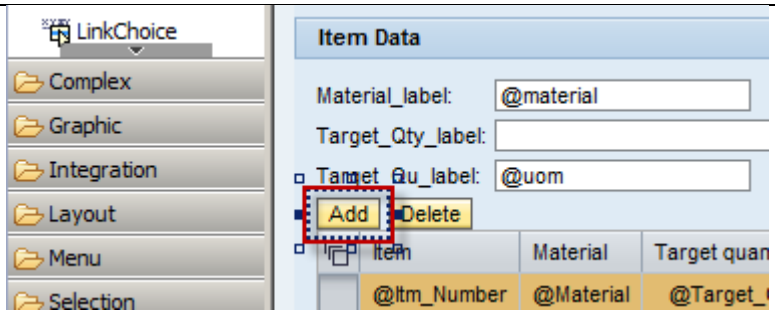
```
public void deleteSalesOrderItem( )  {
    //@@begin deleteSalesOrderItem()
        int n = wdContext.nodeOrder_Items_In().size();
        int leadSelected =
wdContext.nodeOrder_Items_In().getLeadSelection();

        for(int i = n - 1; i >= 0; --i){
            if
(wdContext.nodeOrder_Items_In().isMultiSelected(i) ||
leadSelected == i){

wdContext.nodeOrder_Items_In().removeElement(wdContext.n
odeOrder_Items_In().getElementAt(i));
            }

        }
    //@@end
  }
```

STEP 13. CALL CUSTOM CONTROLLER METHOD ON EVENT BUTTON IN VIEW

93. Open the layout of SalesOrderView.

Double click on "Add" button to go in implementation menthod.

SAVE.




```
public void onActionAddSalesOrderItem(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent wdEvent )
{
  //@@begin onActionAddSalesOrderItem(ServerEvent)
    wdThis.wdGetSOModelControllerController().addSalesOrderItem();
  //@@end
}
```

This code means that when "Add" button is clicked then the event will call method **onActionAddSalesOrderItem()**. Furthermore, it will call method **addSalesOrderItem** that already implemented in custom controller.

```
public void
onActionAddSalesOrderItem(com.sap.tc.webdynpro.progmodel
.api.IWDCustomEvent wdEvent )
  {
    //@@begin onActionAddSalesOrderItem(ServerEvent)

wdThis.wdGetSOModelControllerController().addSalesOrderI
tem();
    //@@end
  }
```

94. Continue with "Delete" button.

SAVE.

```
public void onActionDeleteItemSo(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent wdEvent )
{
  //@@begin onActionDeleteItemSo(ServerEvent)
    wdThis.wdGetSOModelControllerController().deleteSalesOrderItem();
  //@@end
}
```

```
public void
onActionDeleteItemSo(com.sap.tc.webdynpro.progmodel.api.
IWDCustomEvent wdEvent )
  {
    //@@begin onActionDeleteItemSo(ServerEvent)

wdThis.wdGetSOModelControllerController().deleteSalesOrd
erItem();
    //@@end
  }
```

95. Finish with button "Create Sales Order".

SAVE.

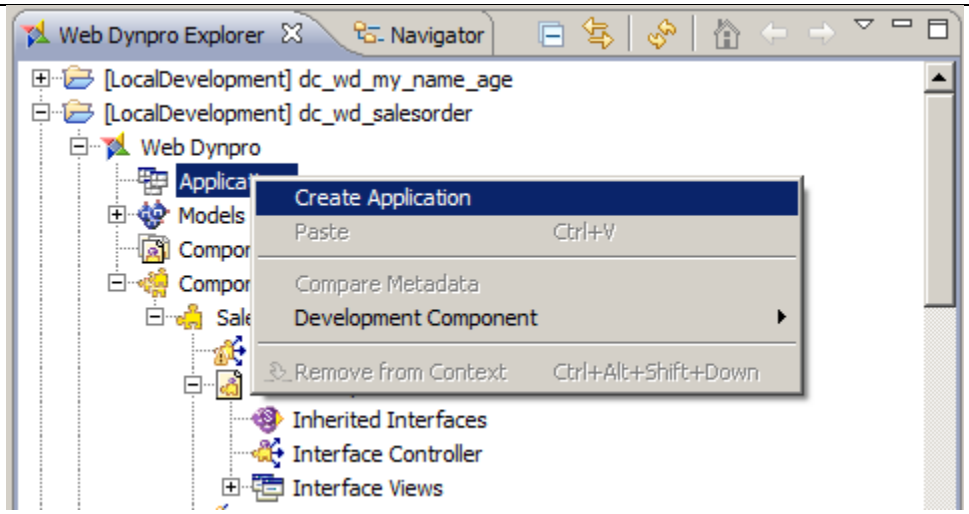
```
public void onActionCreateSalesOrder(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent wdEvent )
{
  //@@begin onActionCreateSalesOrder(ServerEvent)
    wdThis.wdGetSOModelControllerController().executeBapi_Salesorder_Createfromdat2();
    wdThis.wdGetSOModelControllerController().executeBapi_TransactionCommit();
  //@@end
}
```

```
public void
onActionCreateSalesOrder(com.sap.tc.webdynpro.progmodel.
api.IWDCustomEvent wdEvent )
  {
    //@@begin onActionCreateSalesOrder(ServerEvent)

wdThis.wdGetSOModelControllerController().executeBapi_Sa
lesorder_Createfromdat2();

wdThis.wdGetSOModelControllerController().executeBapi_Tr
ansactionCommit();
    //@@end
  }
```

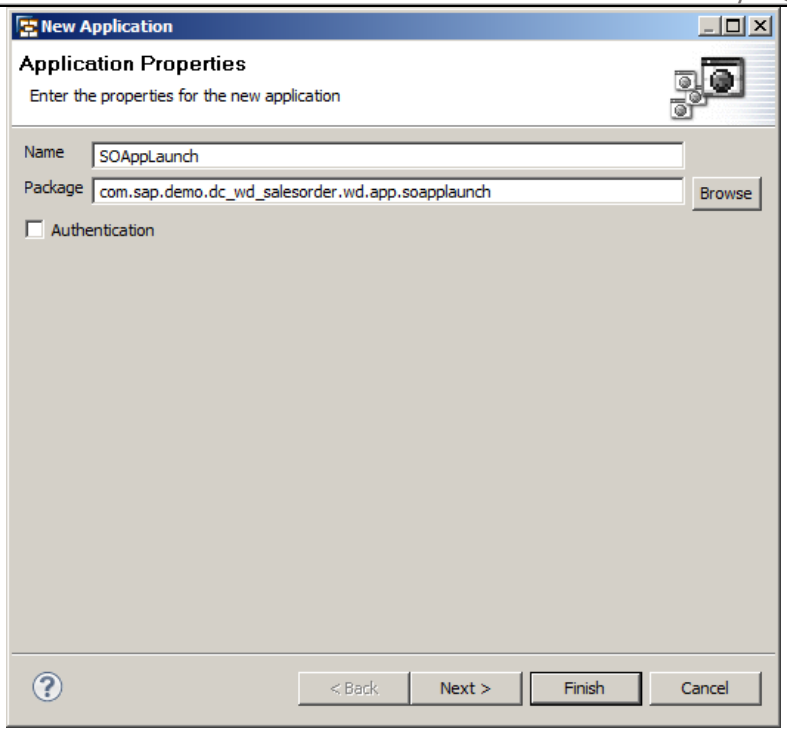
STEP 14. BUILDING, DEPLOYING, CONFIGURING, AND RUNNING YOUR APPLICATION

96. Create application.

Right click on *Application > Create Application.*

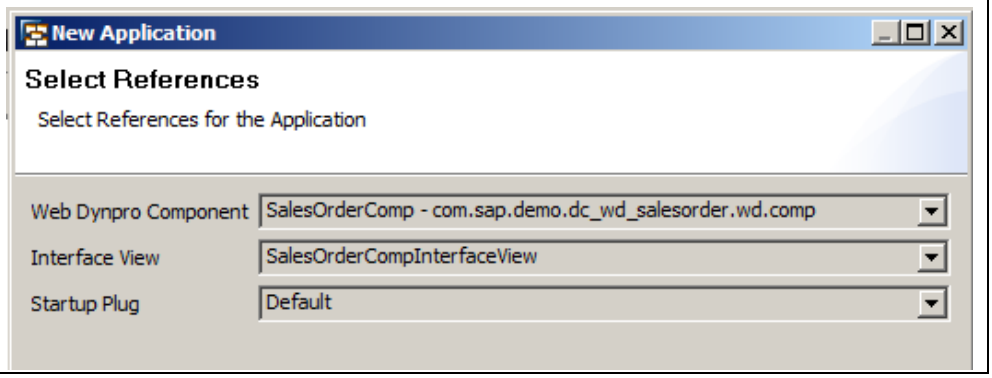


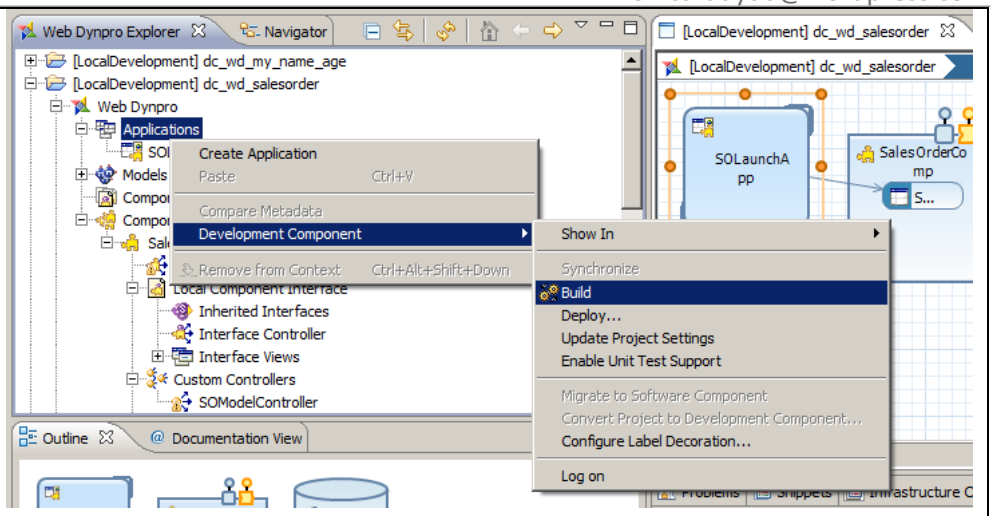
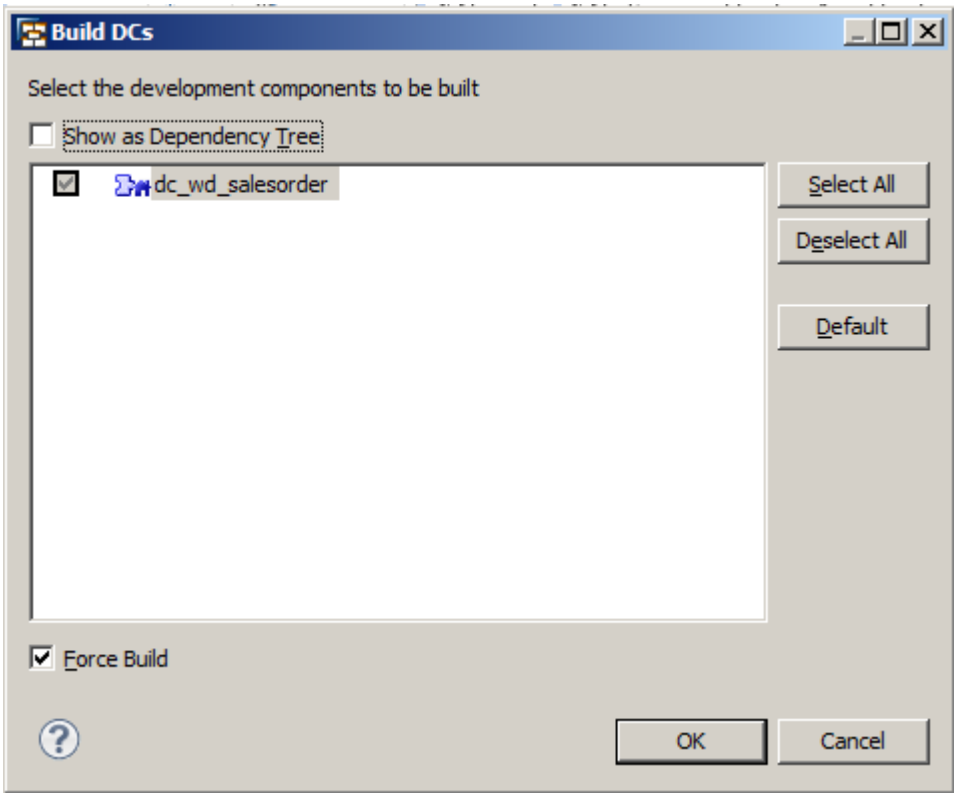
A new window appear, type “SOAppLaunch”. Click Next.

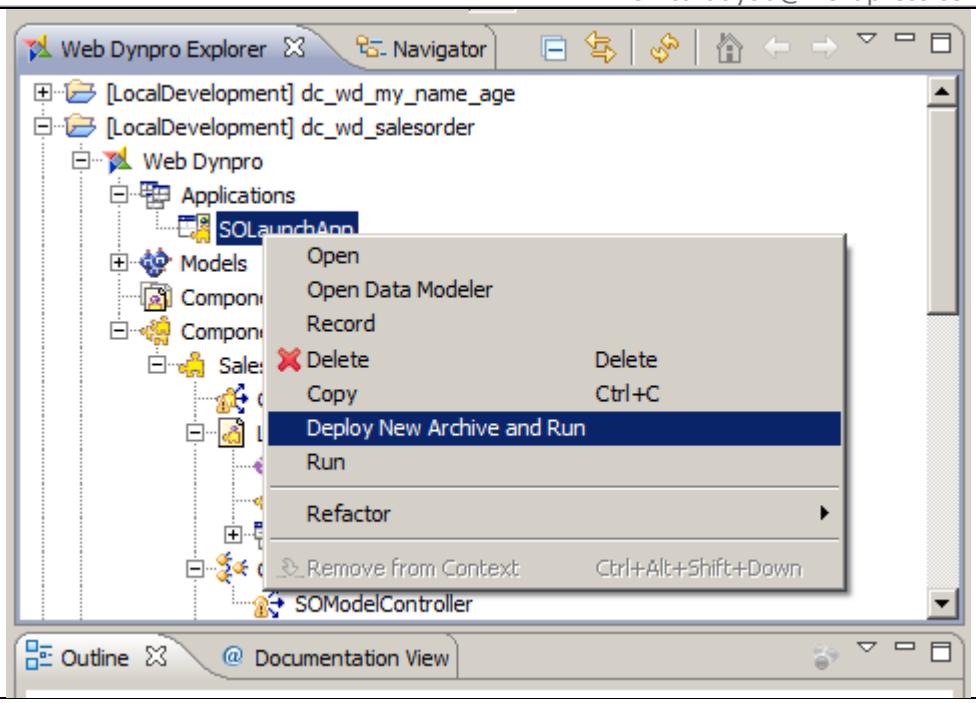
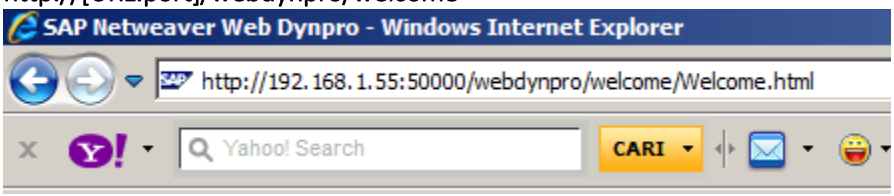


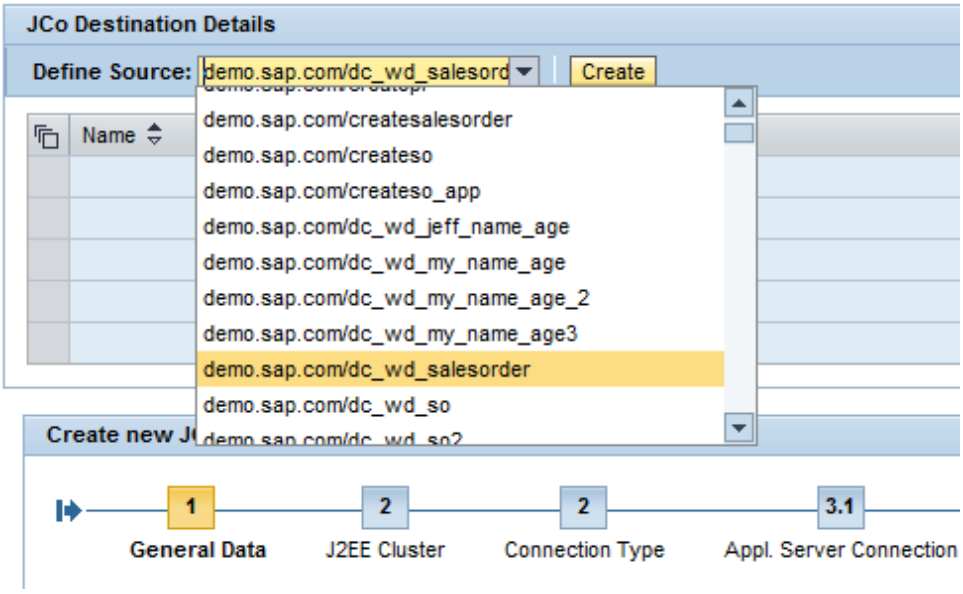
Choose *Use Existing Component,* click Next.

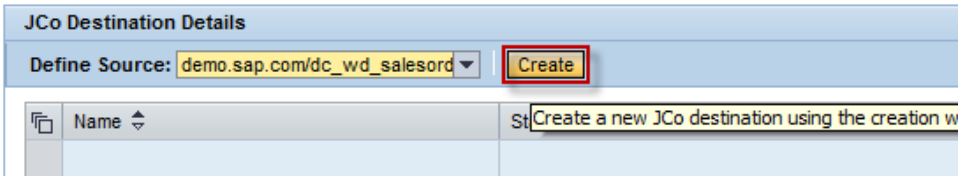
Click Finish.



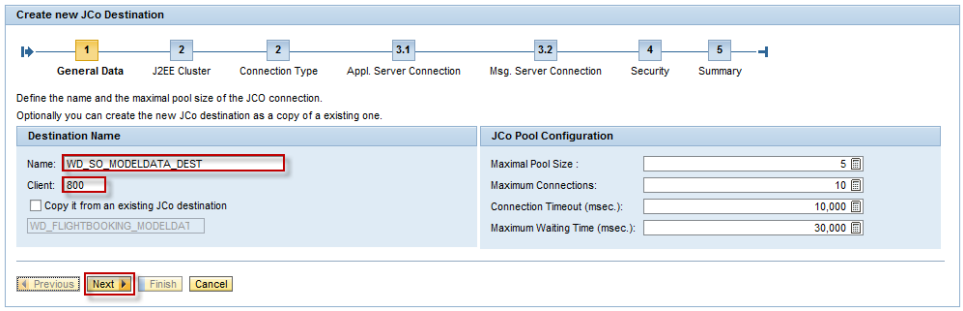
97.	Building the project. Click Finish.	 

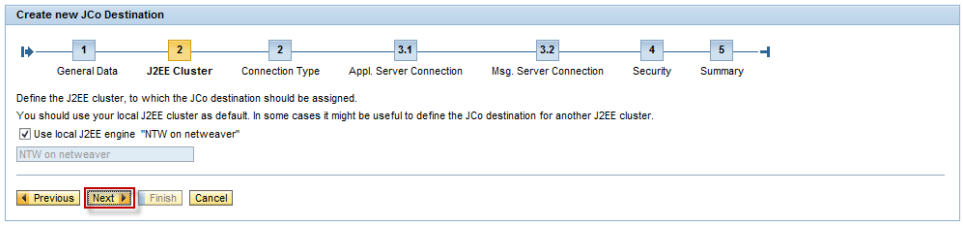
98.	Deploying the project. Right click on SOLaunchApplication > *Deploy New Archive and Run.*	
99.	Creating JCO Connection in Content Administrator.	Open your NWA Content Administrator: http://[URL:port]/webdynpro/welcome  

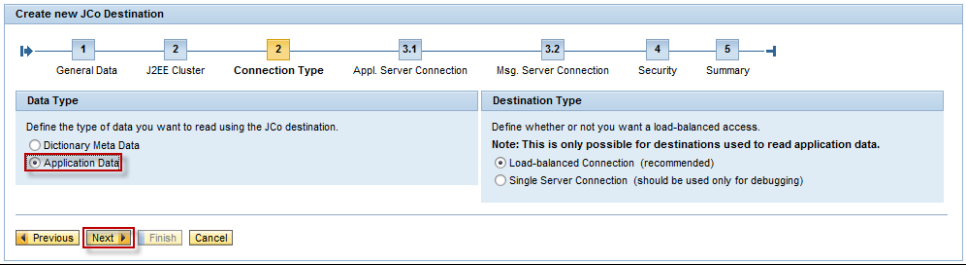


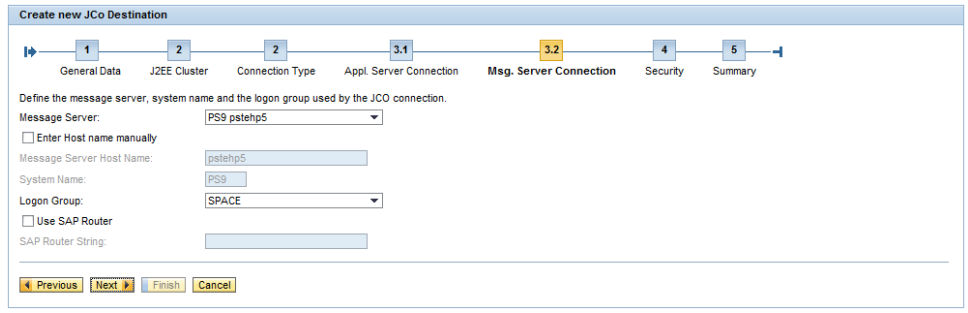


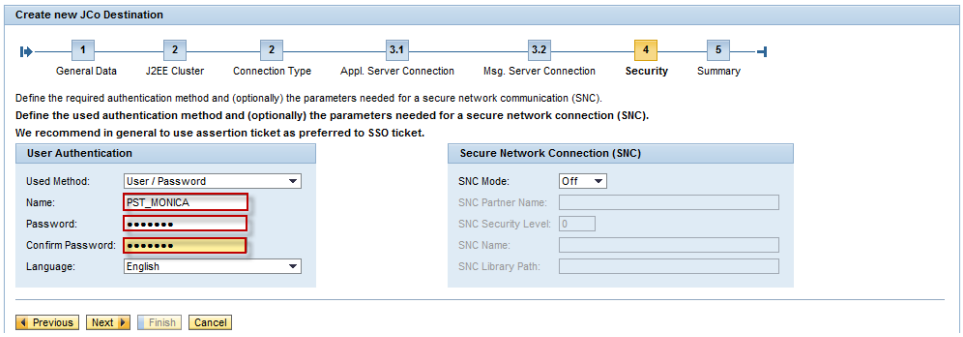
Create RFC Application Data:

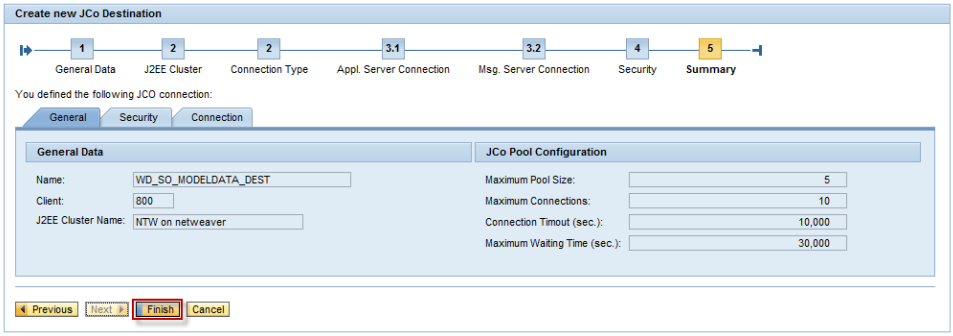




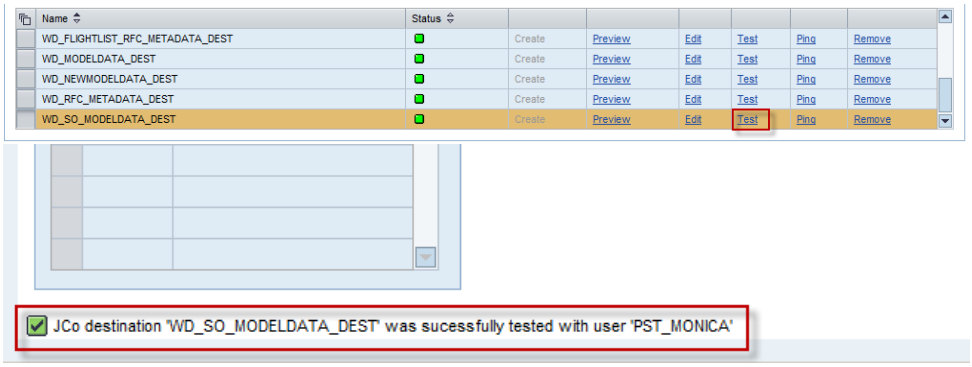




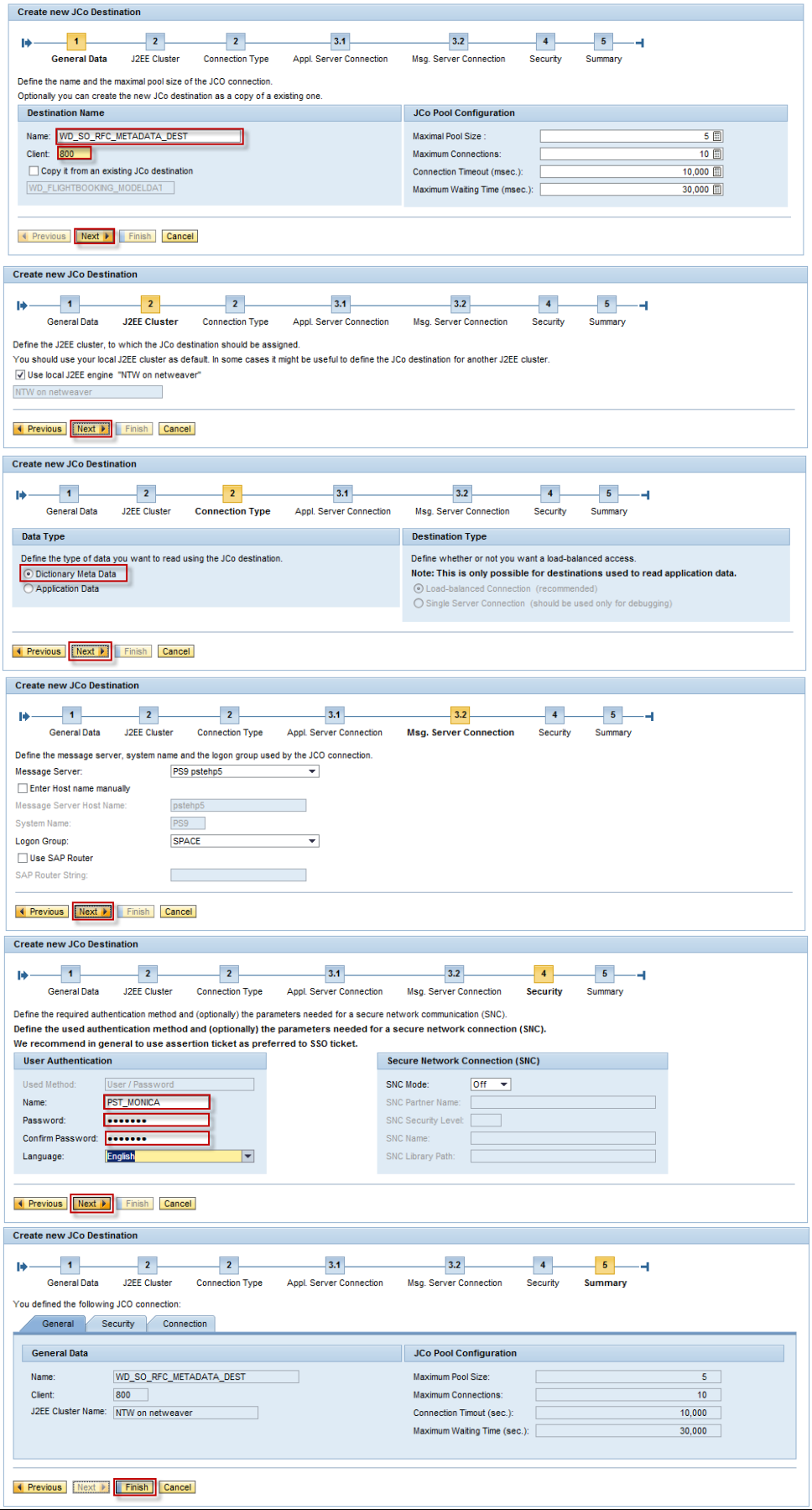


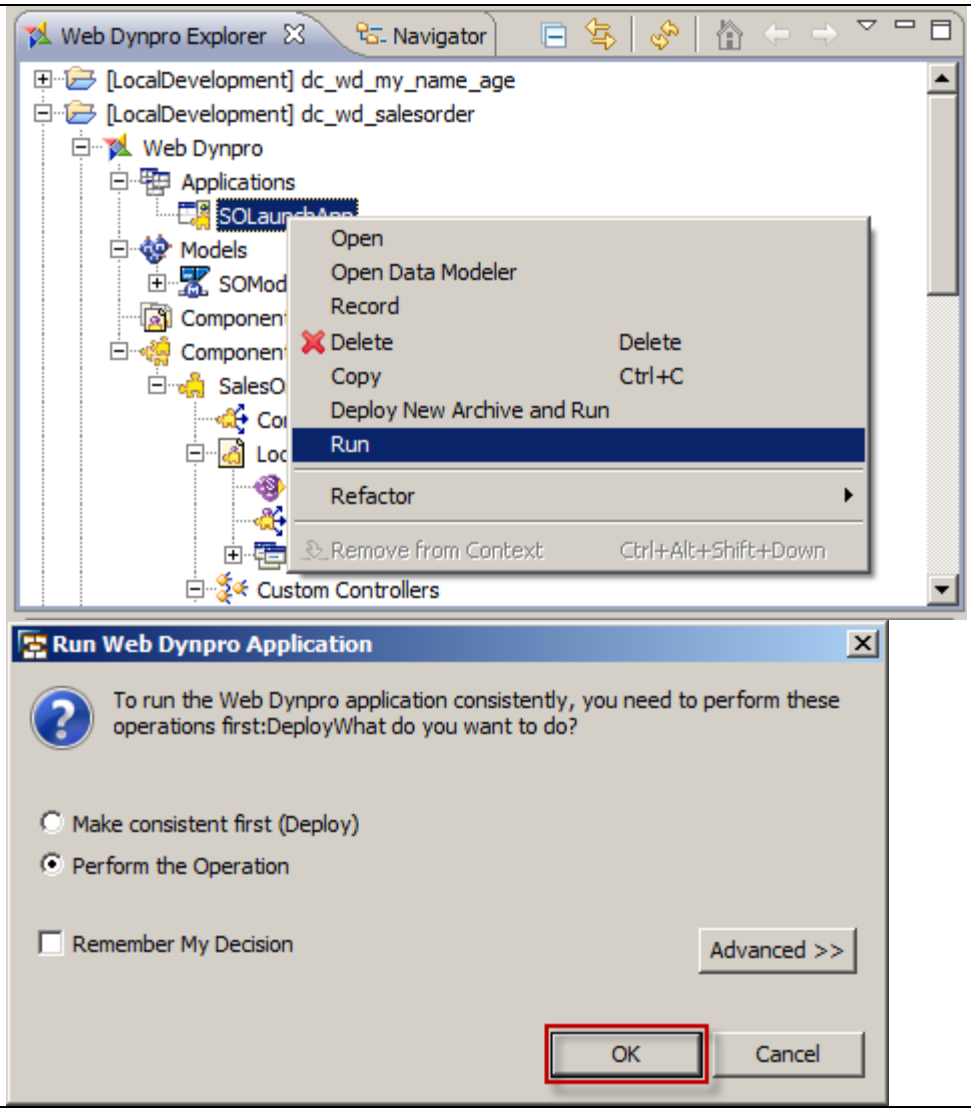


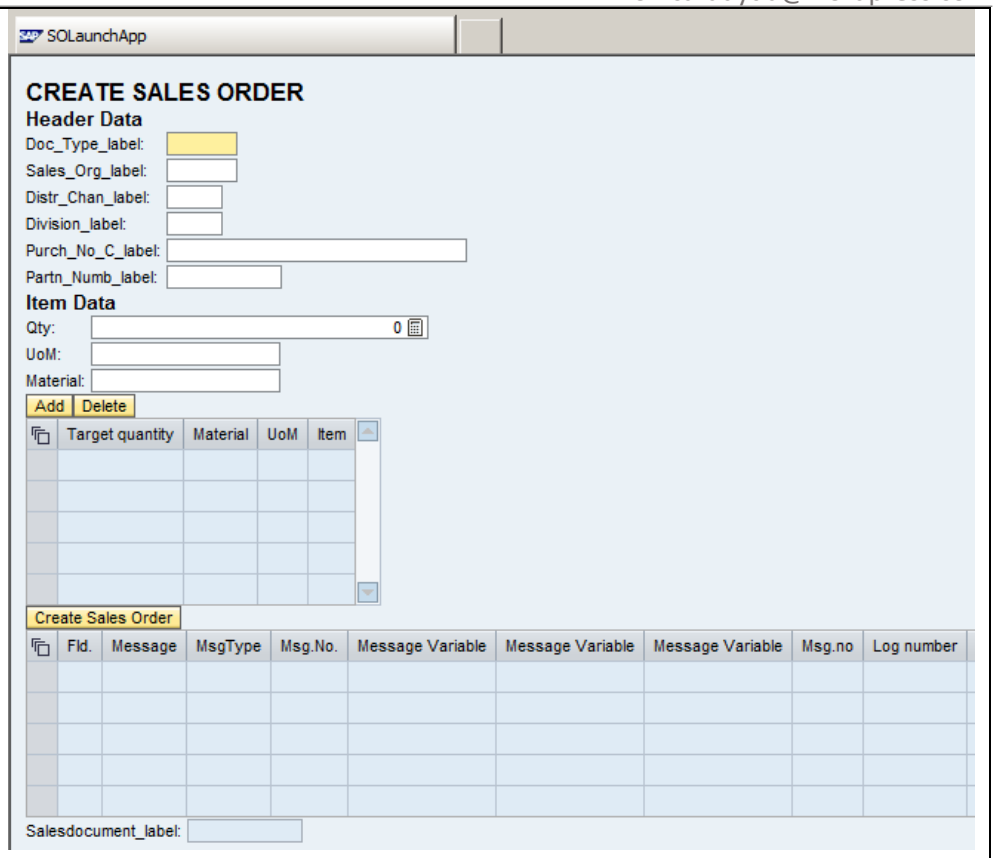
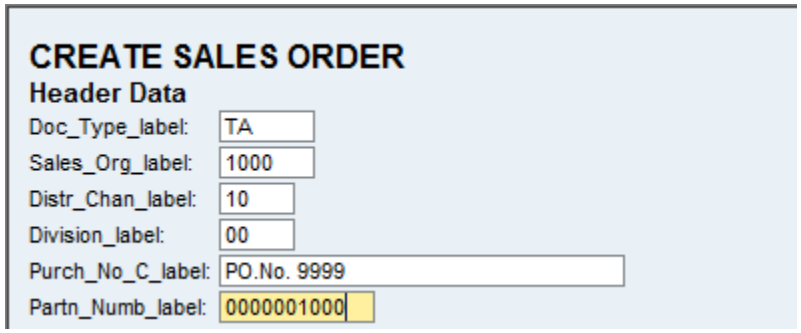
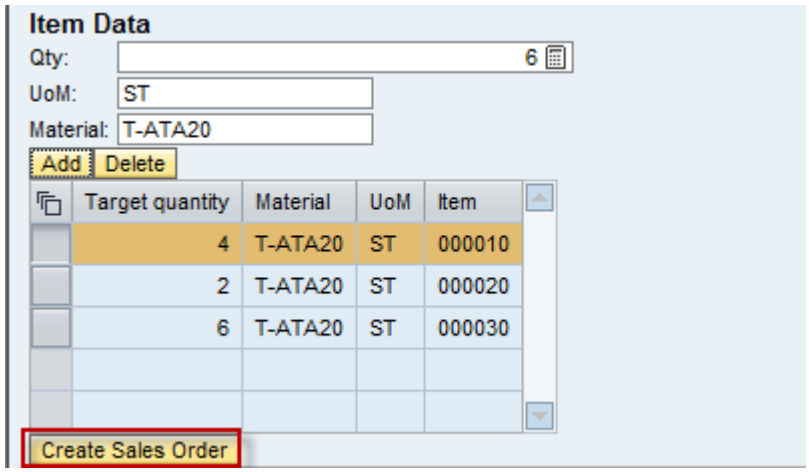
Test connection:

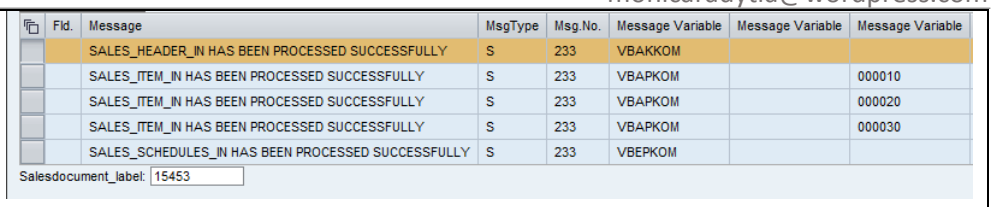


Create RFC Data Dictionary:


Create new JCo Destination
1 General Data
2 J2EE Cluster
2 Connection Type
3.1 Appl. Server Connection
3.2 Msg. Server Connection
4 Security
5 Summary
Define the name and the maximal pool size of the JCO connection.
Optionally you can create the new JCo destination as a copy of a existing one.
Destination Name
Name: WD_SO_RFC_METADATA_DEST
Client: 800
Copy it from an existing JCo destination
WD_FLIGHTBOOKING_MODELDAT
JCo Pool Configuration
Maximal Pool Size : 5
Maximum Connections: 10
Connection Timeout (msec.): 10,000
Maximum Waiting Time (msec.): 30,000
Previous Next Finish Cancel
Create new JCo Destination
Define the J2EE cluster, to which the JCo destination should be assigned.
You should use your local J2EE cluster as default. In some cases it might be useful to define the JCo destination for another J2EE cluster.
Use local J2EE engine "NTW on netweaver"
NTW on netweaver
Previous Next Finish Cancel
Create new JCo Destination
Data Type
Define the type of data you want to read using the JCo destination.
Dictionary Meta Data
Application Data
Destination Type
Define whether or not you want a load-balanced access.
Note: This is only possible for destinations used to read application data.
Load-balanced Connection (recommended)
Single Server Connection (should be used only for debugging)
Previous Next Finish Cancel
Create new JCo Destination
Define the message server, system name and the logon group used by the JCO connection.
Message Server: PS9 pstehp5
Enter Host name manually
Message Server Host Name: pstehp5
System Name: PS9
Logon Group: SPACE
Use SAP Router
SAP Router String:
Previous Next Finish Cancel
Create new JCo Destination
Define the required authentication method and (optionally) the parameters needed for a secure network communication (SNC).
Define the used authentication method and (optionally) the parameters needed for a secure network connection (SNC).
We recommend in general to use assertion ticket as preferred to SSO ticket.
User Authentication
Used Method: User / Password
Name: PST_MONICA
Password: •••••••
Confirm Password: •••••••
Language: English
Secure Network Connection (SNC)
SNC Mode: Off
SNC Partner Name:
SNC Security Level:
SNC Name:
SNC Library Path:
Previous Next Finish Cancel
Create new JCo Destination
You defined the following JCO connection:
General Security Connection
General Data
Name: WD_SO_RFC_METADATA_DEST
Client: 800
J2EE Cluster Name: NTW on netweaver
JCo Pool Configuration
Maximum Pool Size: 5
Maximum Connections: 10
Connection Timout (sec.): 10,000
Maximum Waiting Time (sec.): 30,000
Previous Next Finish Cancel

		
100.	Launching the application.	

101.	Result. Since, we not implemented yet data conversion, data such as order type is still using default language that acknowledged by BAPI.	  

		
102.	Checking the result. Based on the Sales Order number that created in Java Web Dynpro , we then checked the data in the SAP backend system. Now we already succeed to create simple application Create Sales Order in Java Web Dynpro and recorded automatically in SAP backend system.	In SAP backend system, open t-code VA03 display Sales Order.