



An Oracle White Paper

August 2012

ADF Sales Funnel POC Tutorial





PARTNER
BUSINESS
DEVELOPMENT

ECEMEA A&C TECHNOLOGY ADOPTION OFFICE

Author(s)

Jernej Kase

jernej.kase@oracle.com

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

This material or any portion of it may not be copied in any form or by any means without the express prior written permission of Oracle Corporation. Any other copying is a violation of copyright law and may result in civil and/or criminal penalties.

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them in writing to the Authors of this document.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Disclaimer

The following is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

Hardware and Software, Engineered to Work Together

ORACLE®

Executive Overview	3
I. Creating Business and Data Access Layer	4
1. Create ADF Application	4
2. Create business components	6
3. Setting Accounts entity properties	17
4. Setting Contacts entity properties	22
5. Setting Employee entity properties	28
6. Setting Opportunity entity properties.....	34
7. Accounts LOV View.....	44
8. Contacts LOV View	50
9. Employees LOV View.....	57
10. Shared LOV AM	63
11. Employees ManagerId LOV.....	68
12. Opportunities Account LOV	73
13. Opportunities Contact LOV	82
II. Creating User Interface and Security Foundation.....	93
14. Main Form	93
15. Login	100
16. Security	101
17. Roles and grants configuration	107
18. WLS security configuration	115
19. Page Fragment template	137
20. BTF Template	144
III. Implementing Core Use Cases	147
21. Lead Edit BTF	147
22. Lead Create BTF	155
23. Lead Edit Form.....	165
24. Add Lead Create to Main Form	179
25. Account Edit BTF	187
26. Account Create BTF	201
27. Adding Account Create to Lead Edit BTF	211
28. Refresh Accounts LOV after insert	216
29. Contact Edit BTF	222
30. Contact Create BTF	236
31. Adding Contact Create to Lead Edit	249
32. Empld in Session	263
IV. Implementing Dashboard	268
33. Dashboard.....	268
34. Implement Dashboard page	283
35 Add data to Dashboard.....	293
36. Geographic map.....	305
37. Bubble chart.....	320
38. Gauges	328

39. Image Servlet	344
40. Connecting Dashboard to Lead Edit	357
V. Implementing Hierarchy Viewer	367
41. Hierarchy viewer.....	367
42. Connecting hierarchy viewer to the dashboard.....	387
VI. Extending Hierarchy Viewer with DVT Components	398
43. Adding DVT to Hierarchy viewer.....	398
44. Adding funnel graph	434
45. Adding report chart.....	454

Executive Overview

This tutorial was built to showcase real-life capabilities of Oracle Fusion Middleware capabilities, specifically the Application Development Framework – ADF.

Many vendors rely on short & simple technology samples, which showcase the key features of specific technology. But usually the samples are out of real-life context and focus on one specific area, but do not address how a particular technology or feature can be used in real-life scenarios of building enterprise applications.

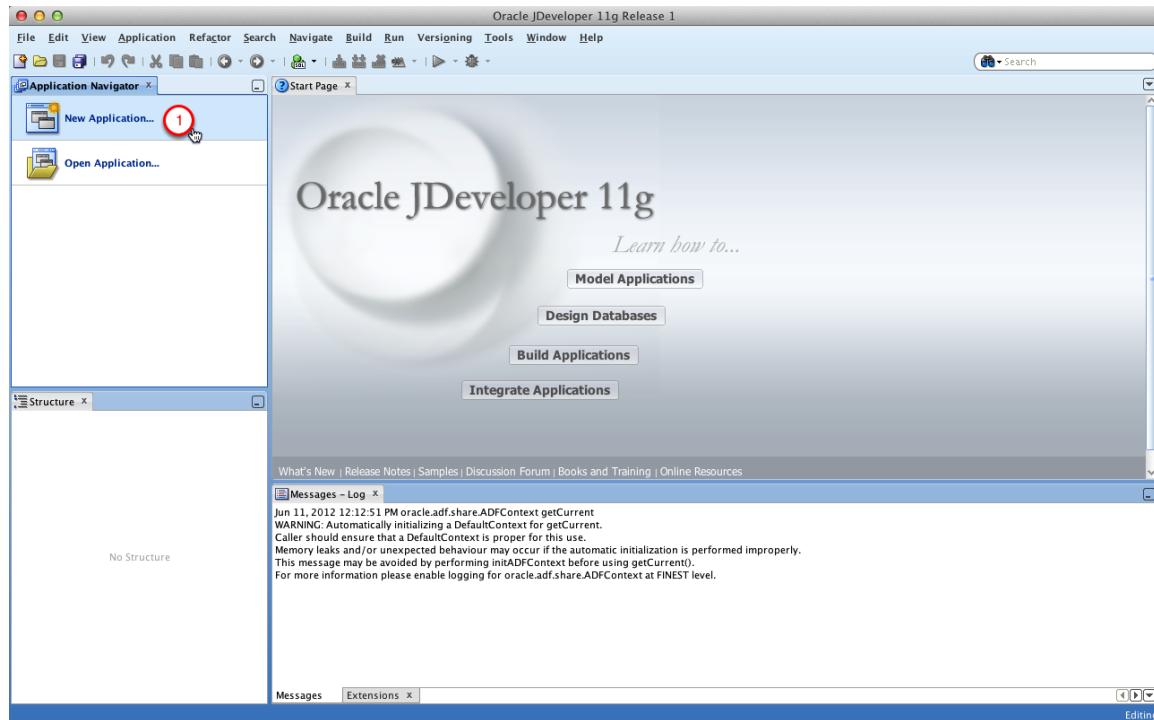
To prove outstanding ADF & Oracle Fusion Middleware capabilities we've built a complex, feature rich scenario demonstrating the true value of technology in question.

I. Creating Business and Data Access Layer

1. Create ADF Application

In this lesson, we'll start from scratch and create an empty ADF application.

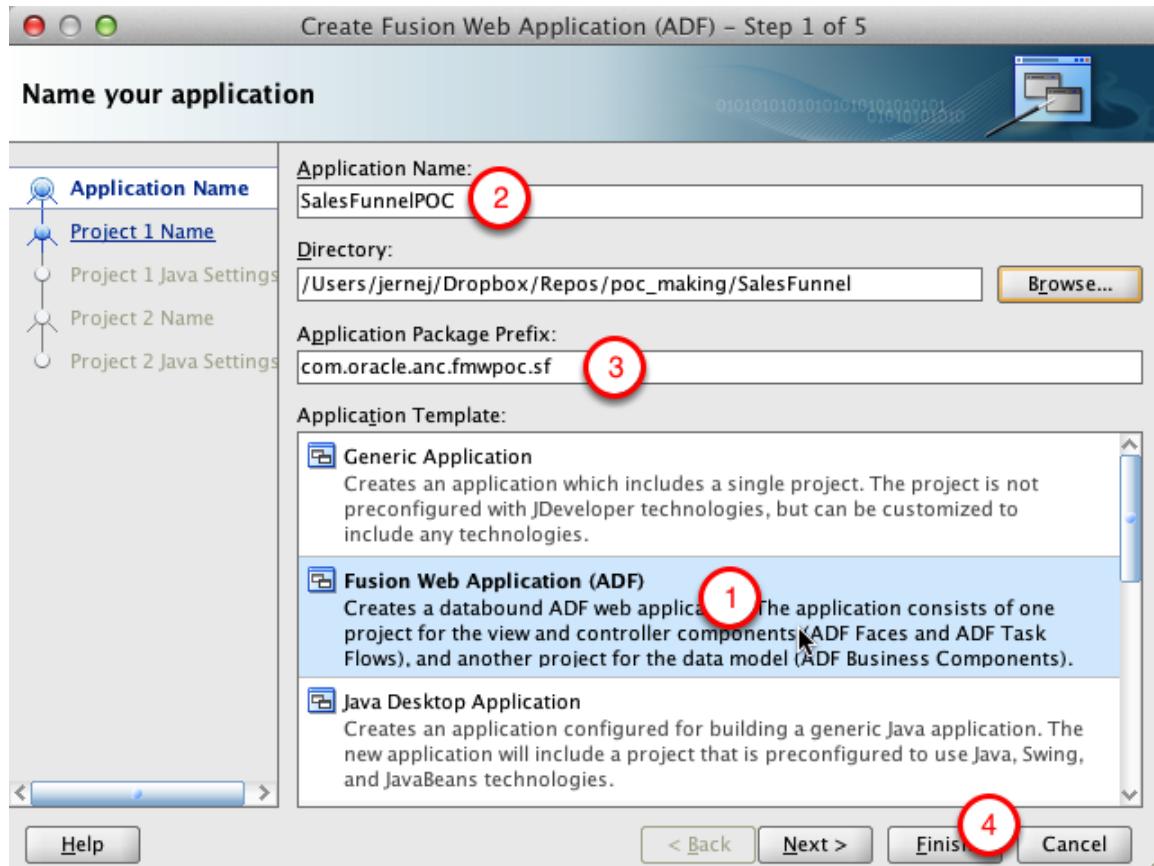
Create new application



1. Click New Application

Note: if it's not the first time you ran JDeveloper, you might not see this screen. In this case, you can also select Application->New in the menu

Create Fusion Web Application (ADF) - Step 1 of 5



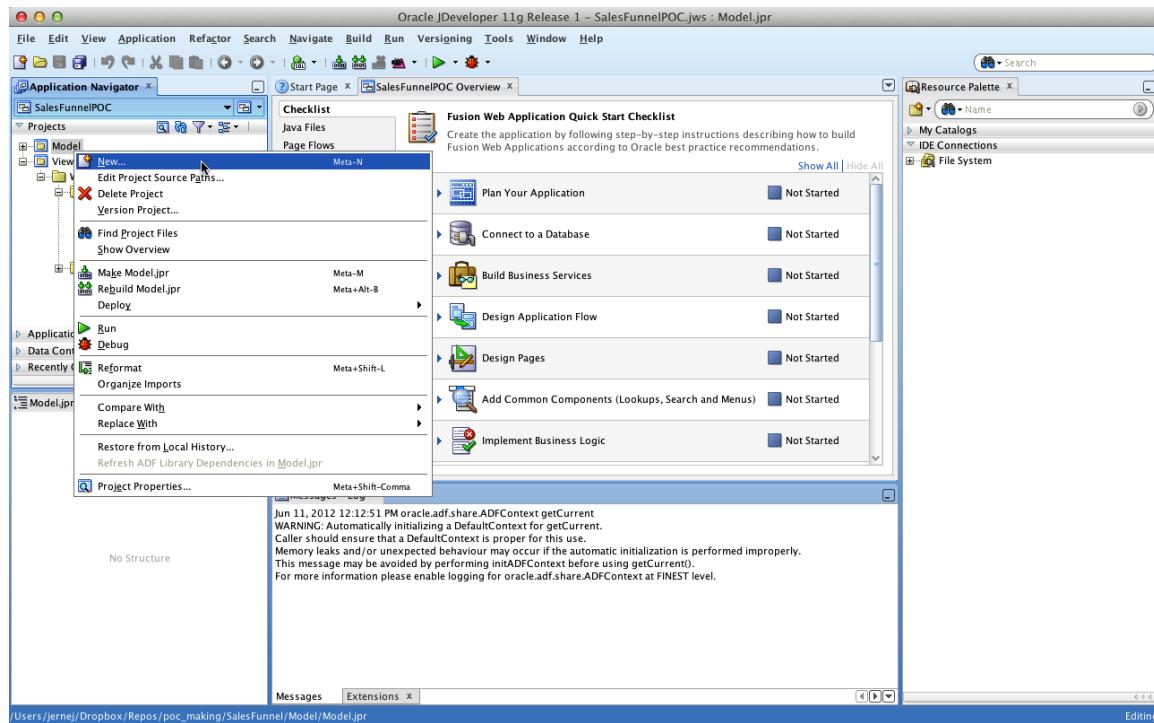
1. Select "Fusion Web Application (ADF)" Application Template
2. Name the application "SalesFunnelPOC"
3. Set application Package Prefix to "com.oracle.anc.fmw poc.sf"*
4. Click Finish

*for this application, we recommend using the same package to simplify code reuse later on.

2. Create business components

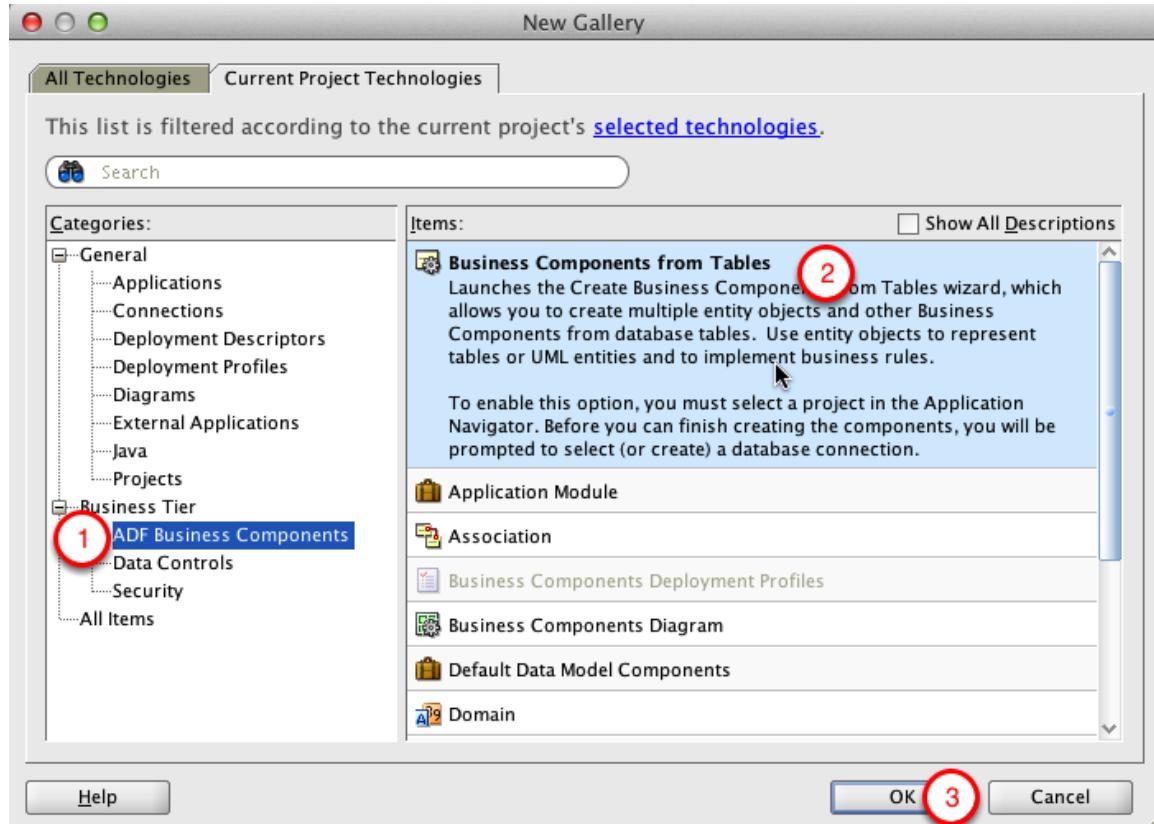
In our example, the database schema for the application has already been setup, so the best way to start developing the application is to create Data Access / Business Components layer from tables. We will use ADF Business Components for that.

Create business components



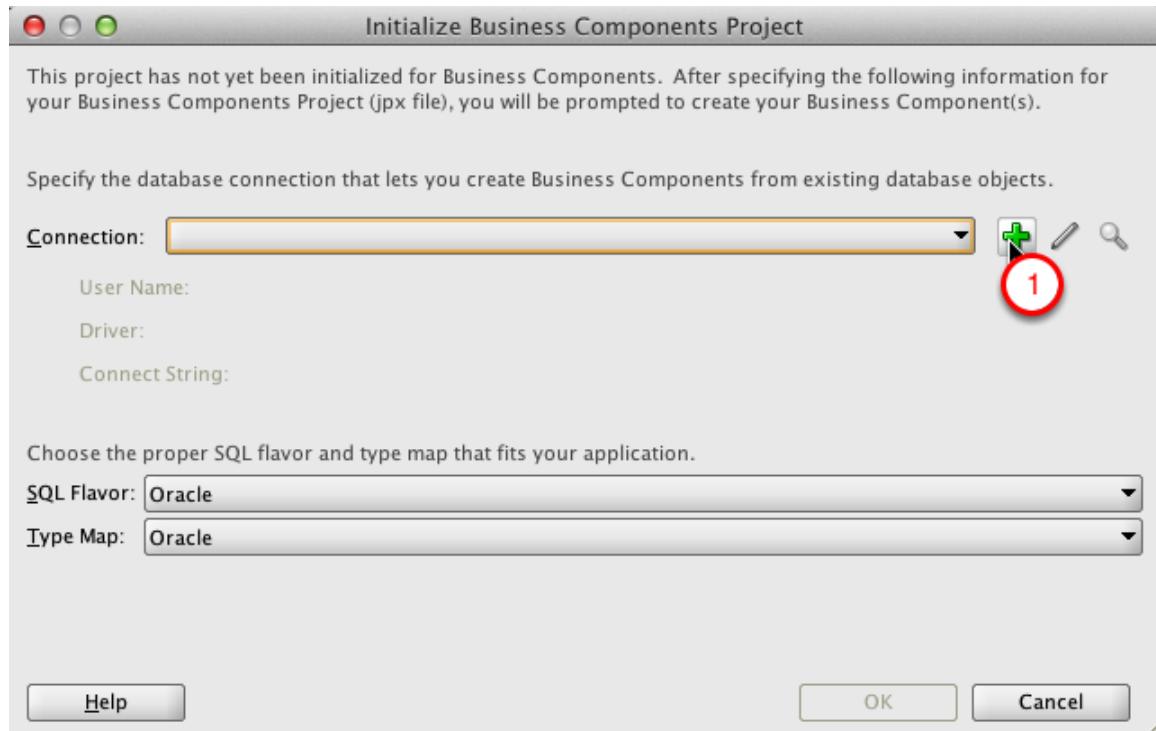
1. Right-click the model project
2. Select New in the popup menu

New Gallery



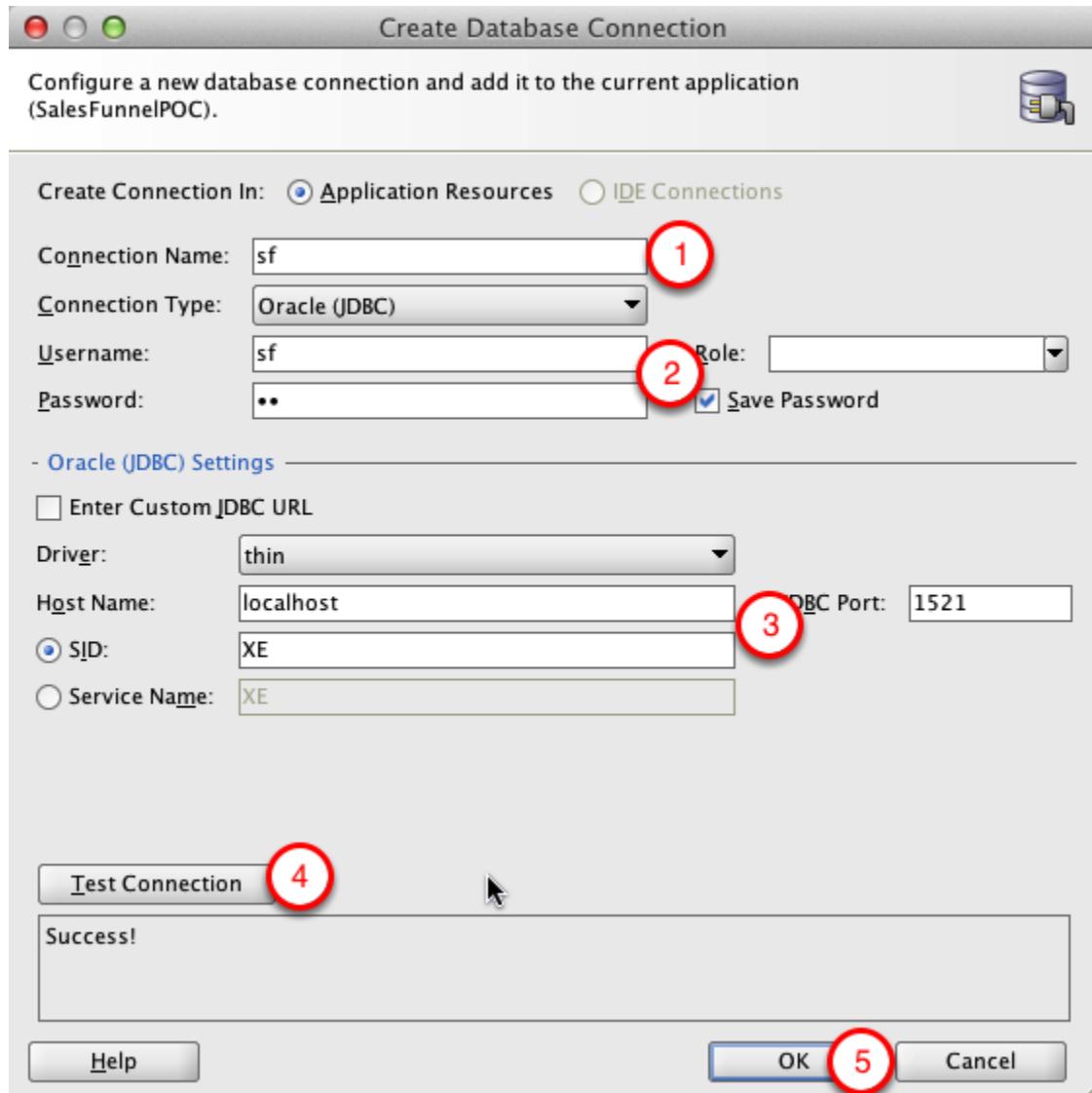
1. Select ADF Business Components category
2. Select Business Components from Tables
3. Click OK

Initialize Business Components Project



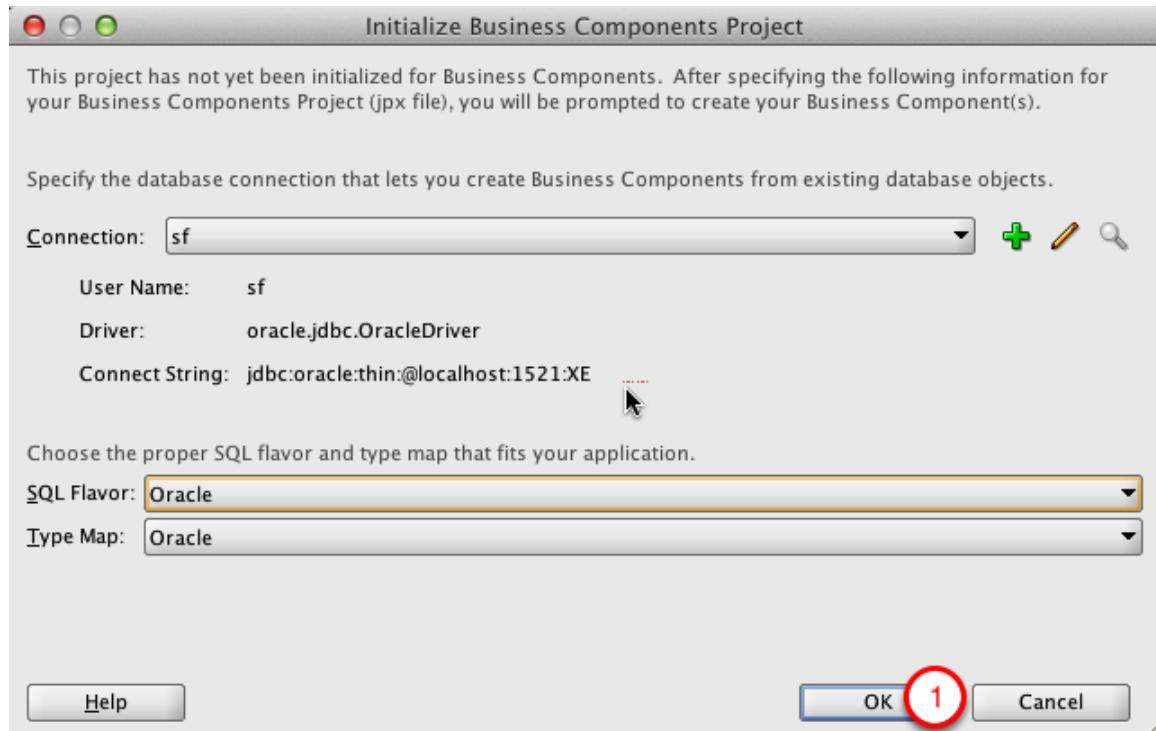
1. In the Connection dialog, click the green plus to create new connection

Create Database Connection



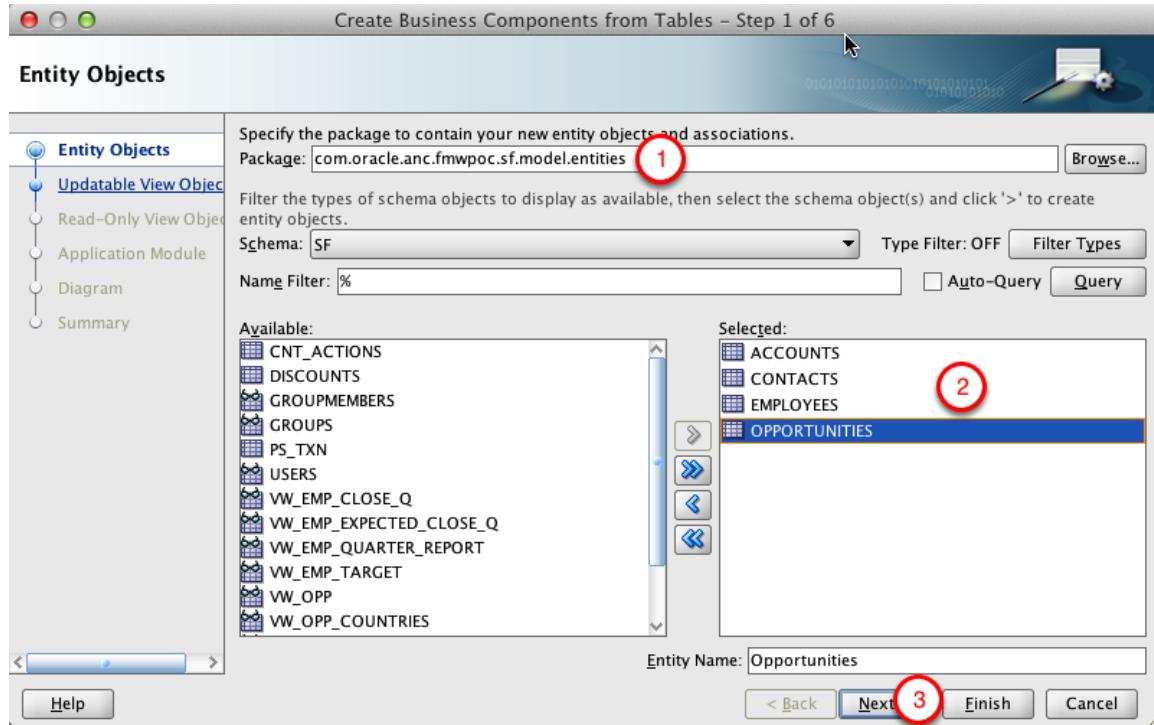
1. Set connection name to sf
2. Set username and password to sf
3. Enter your database host name, SID and port
4. Test connection and make sure it works before continuing
5. Click OK

Initialize Business Components Project



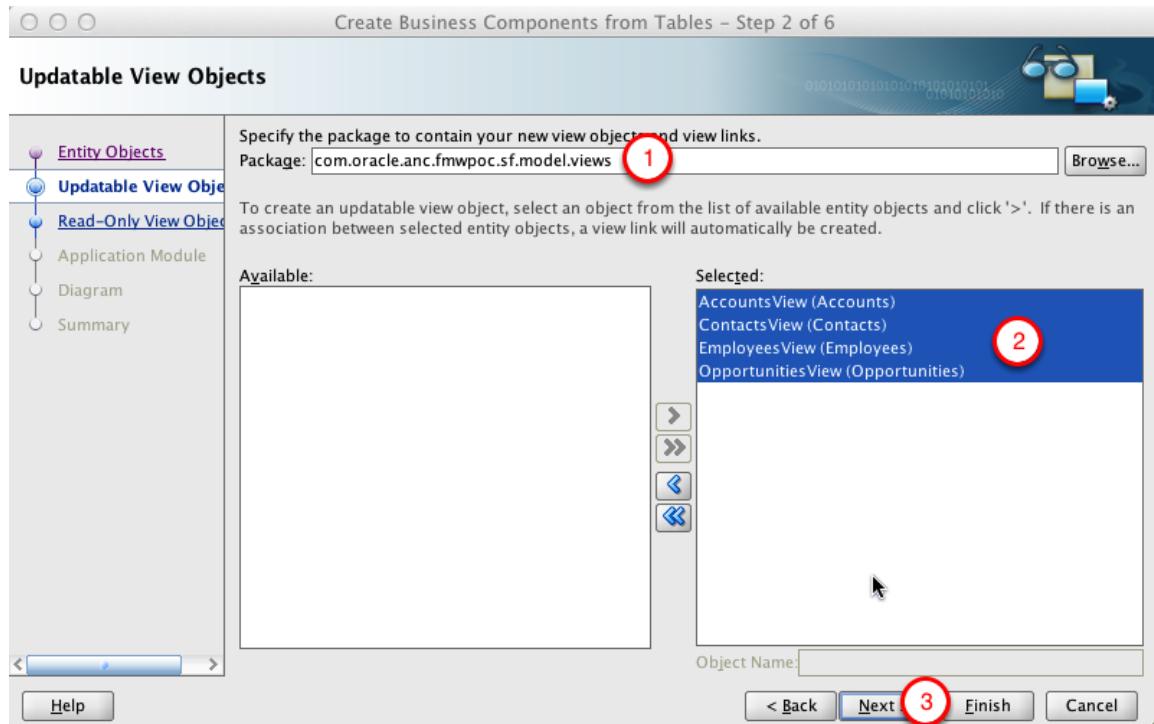
1. We'll use the default settings, so just click OK

Create Business Components from Tables - Step 1 of 6



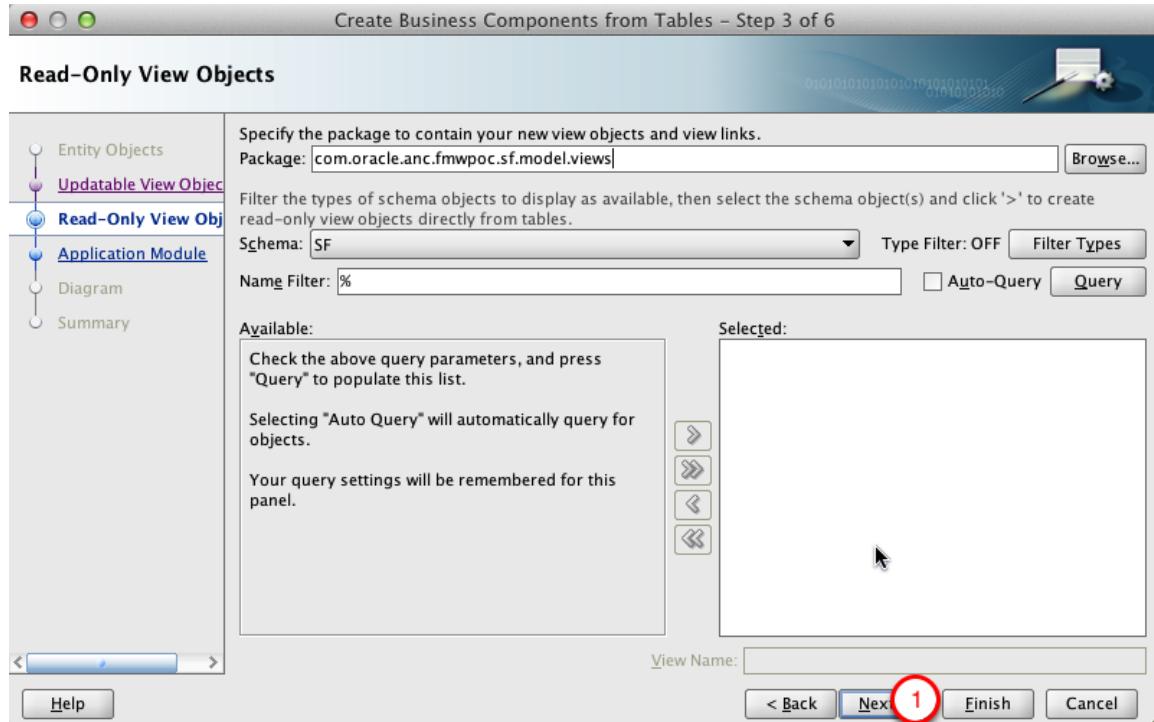
1. Append ".entities" to the package
2. Using the shuttle dialog, select Accounts, Contacts, Employees and Opportunities tables
3. Click Next

Create Business Components from Tables - Step 2 of 6



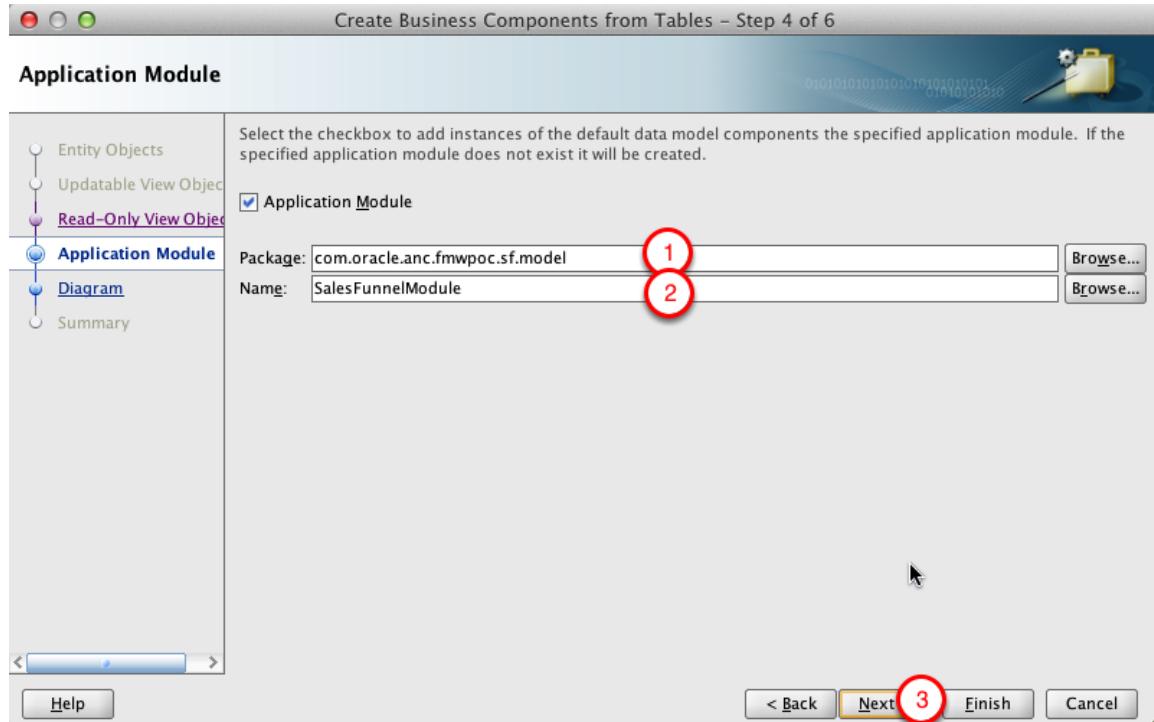
1. Remove ".entities" and append ".views" to the package name
2. Using the shuttle, select all entities
3. Click Next

Create Business Components from Tables - Step 3 of 6



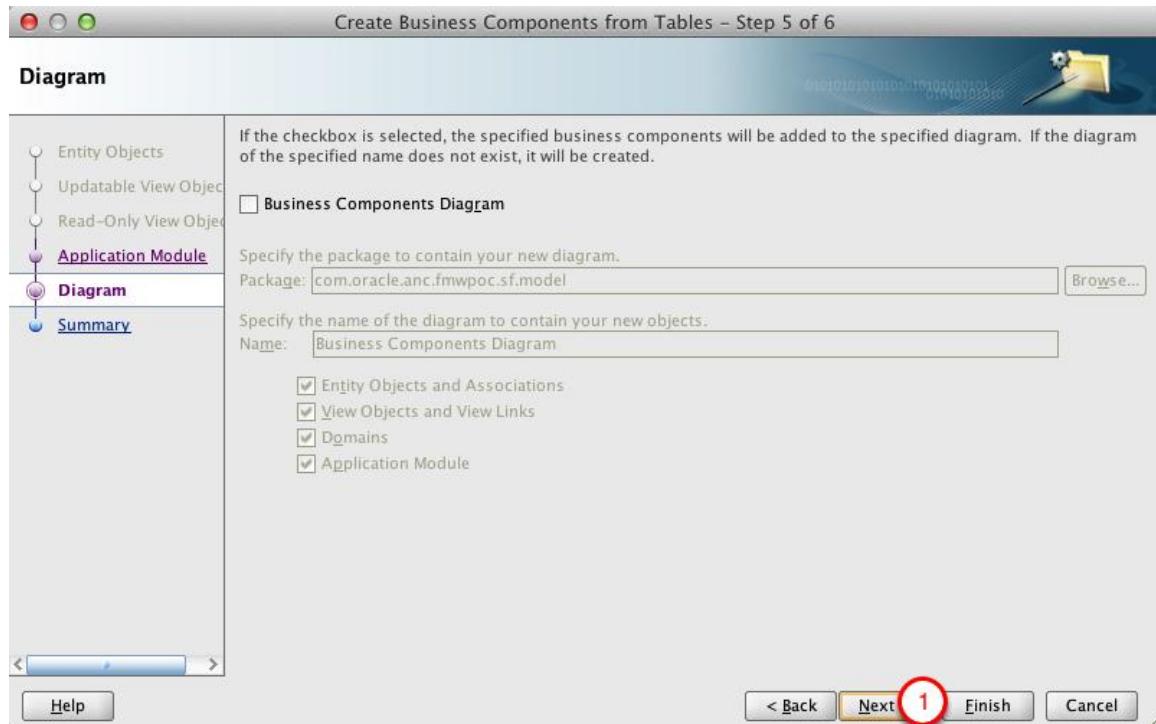
1. We will skip the read-only view objects step, click Next

Create Business Components from Tables - Step 4 of 6



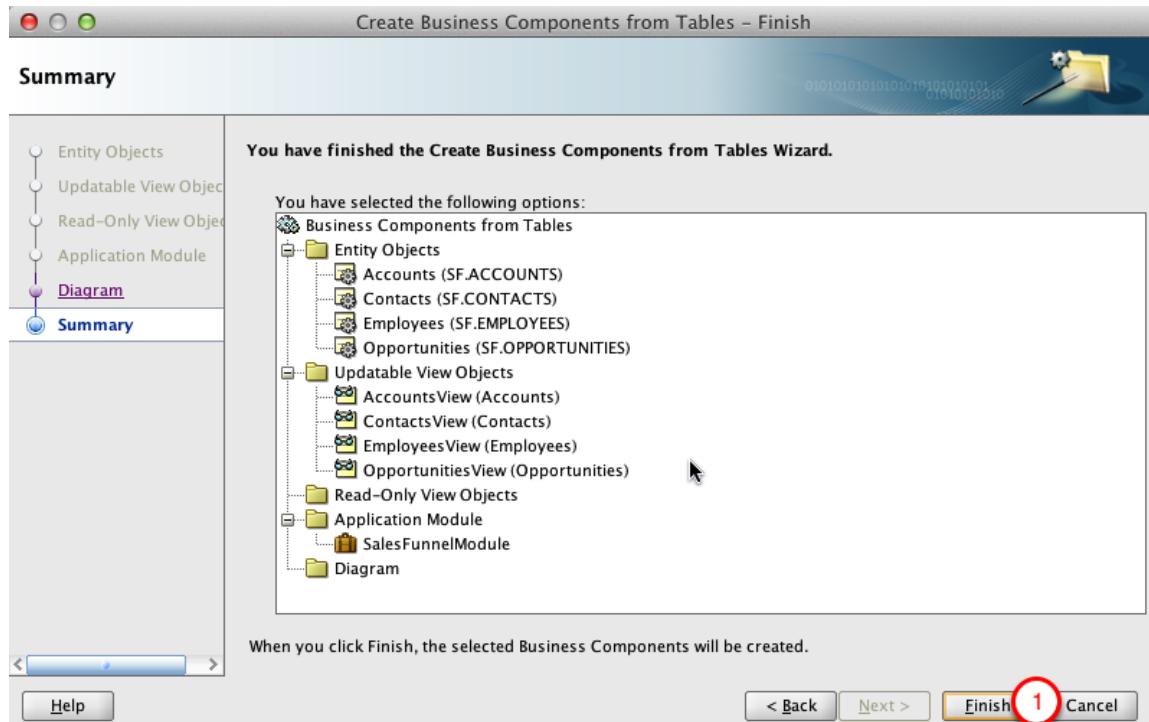
1. Remove ".views" from the package name
2. Set Application module Name to "SalesFunnelModule"
3. Click Next

Create Business Components from Tables - Step 5 of 6



1. We will skip the diagram, click Next

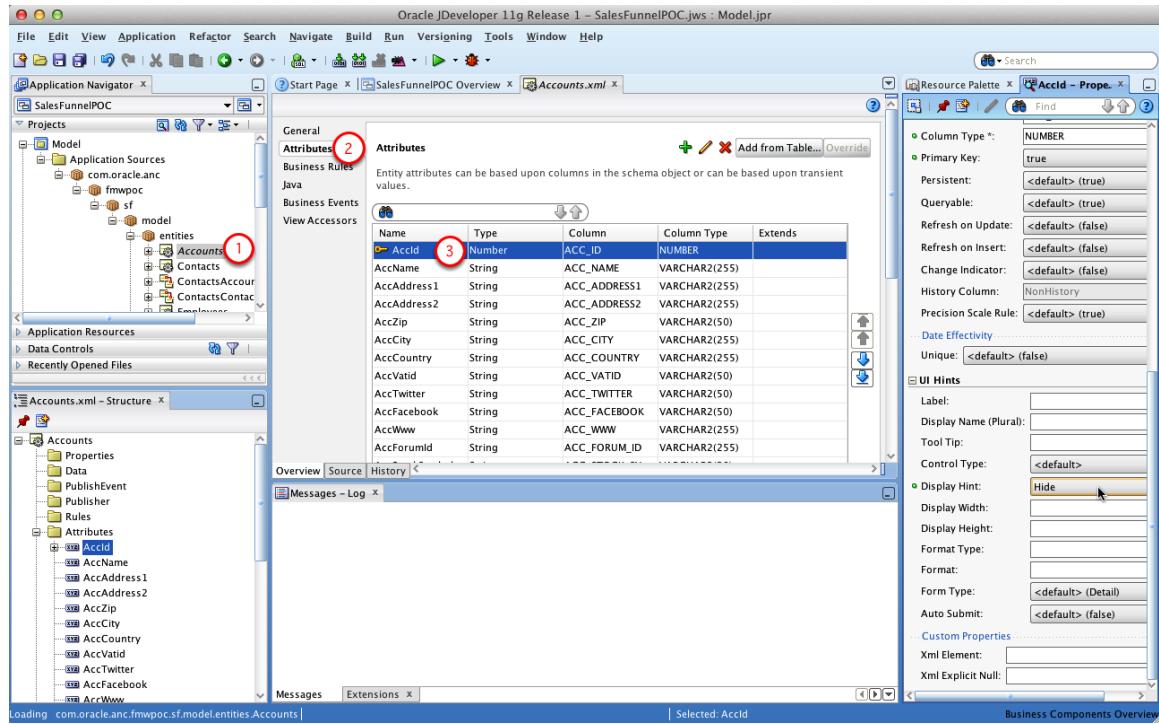
Create Business Components from Tables - Finish



1. Click Finish

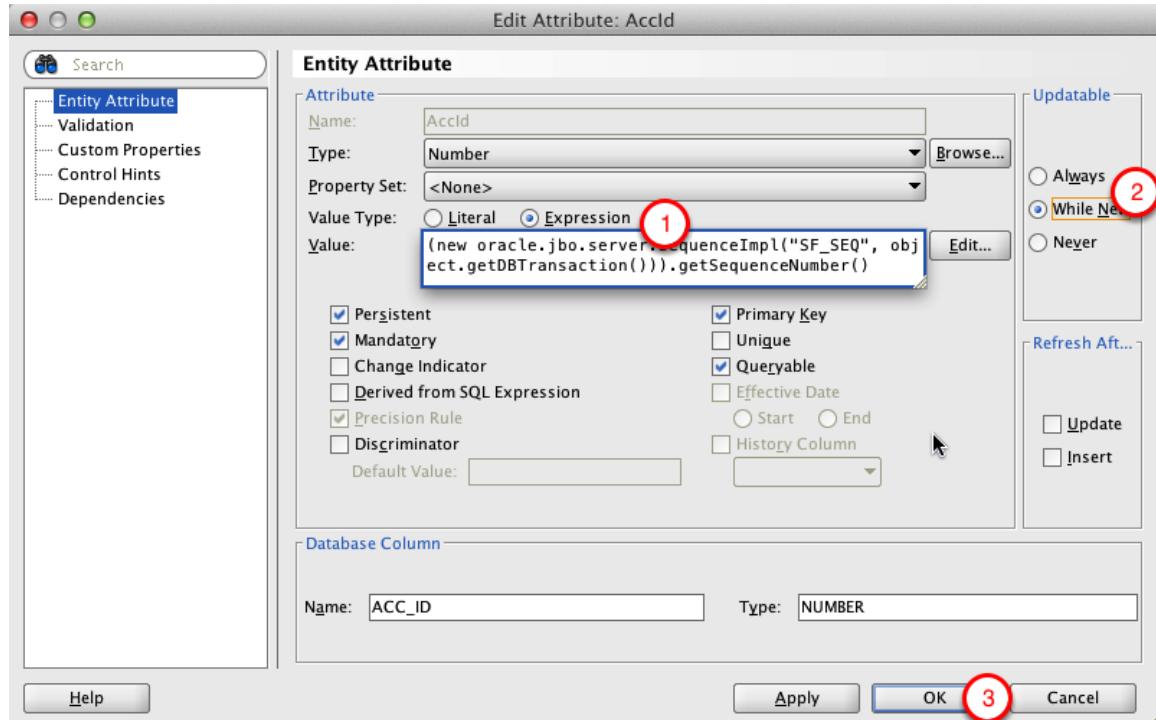
3. Setting Accounts entity properties

Select AcctId attribute



1. Locate and double-click Accounts entity in Application Navigator
2. Open Attributes Tab
3. Double-click AcctId

Edit Attribute: AccId

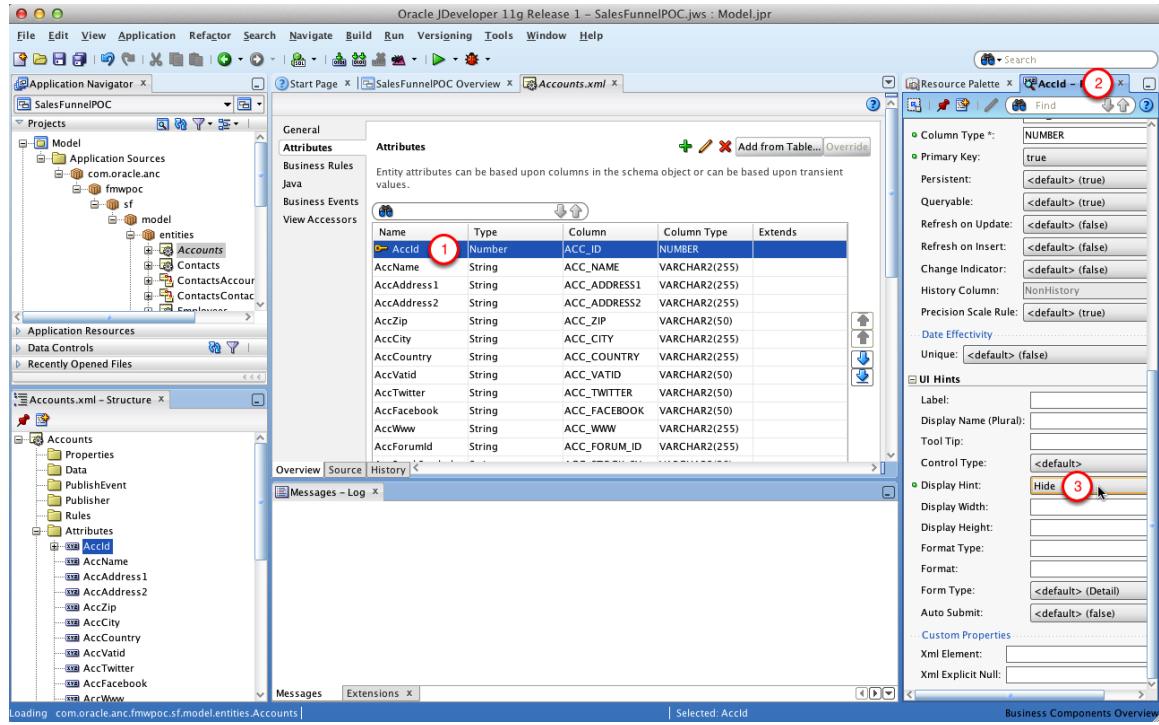


We will use DB sequences to populate ID fields.

1. Set Value Type to Expression and paste the below code in the Value field
2. Set Updatable to "While New"
3. Click OK

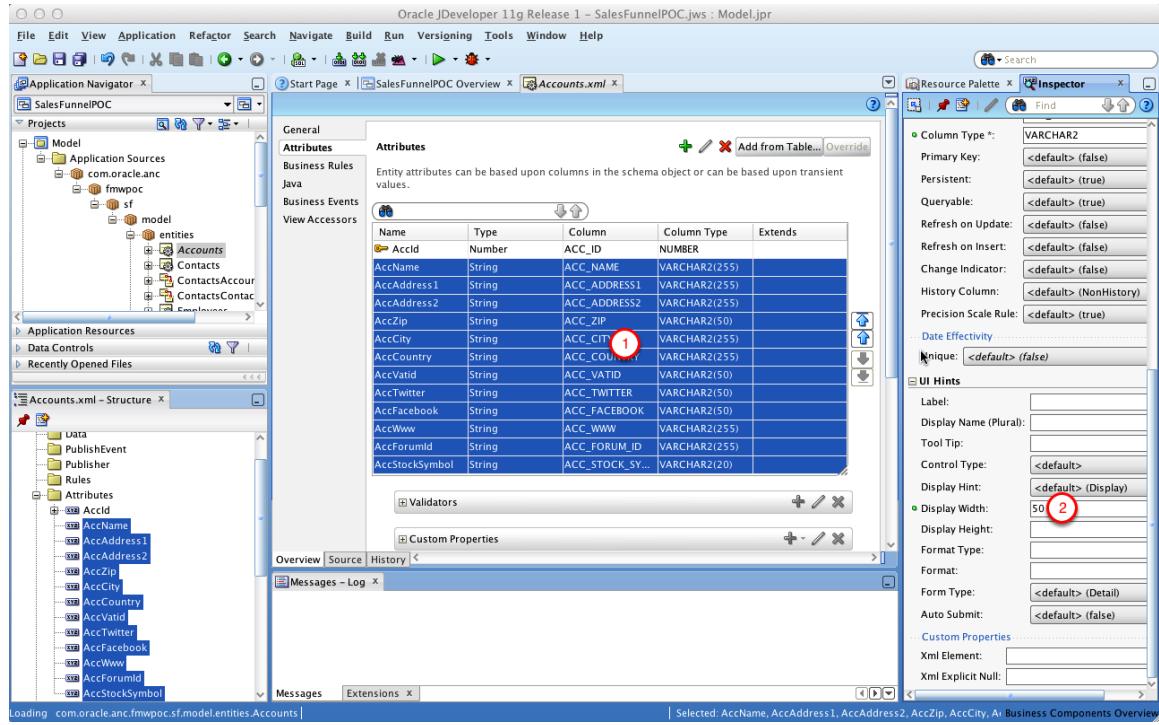
```
(new oracle.jbo.server.SequenceImpl("SF_SEQ",
object.getDBTransaction())).getSequenceNumber()
```

Edit Attribute: AccId



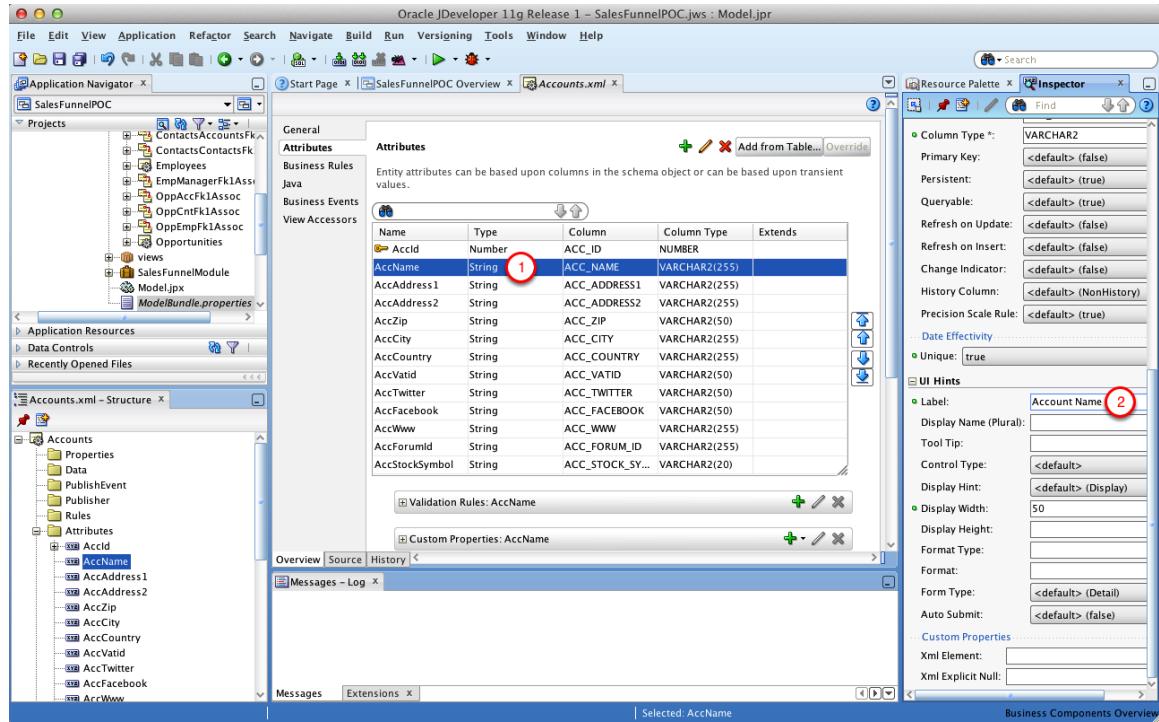
1. Select AccId attribute
2. Open Property Inspector (shortcut on most systems is Ctrl-Shift-I)
3. Set Display Hint to Hide

Set fields width



1. Select all attributes
2. Set Display Width to 50

Set fields labels

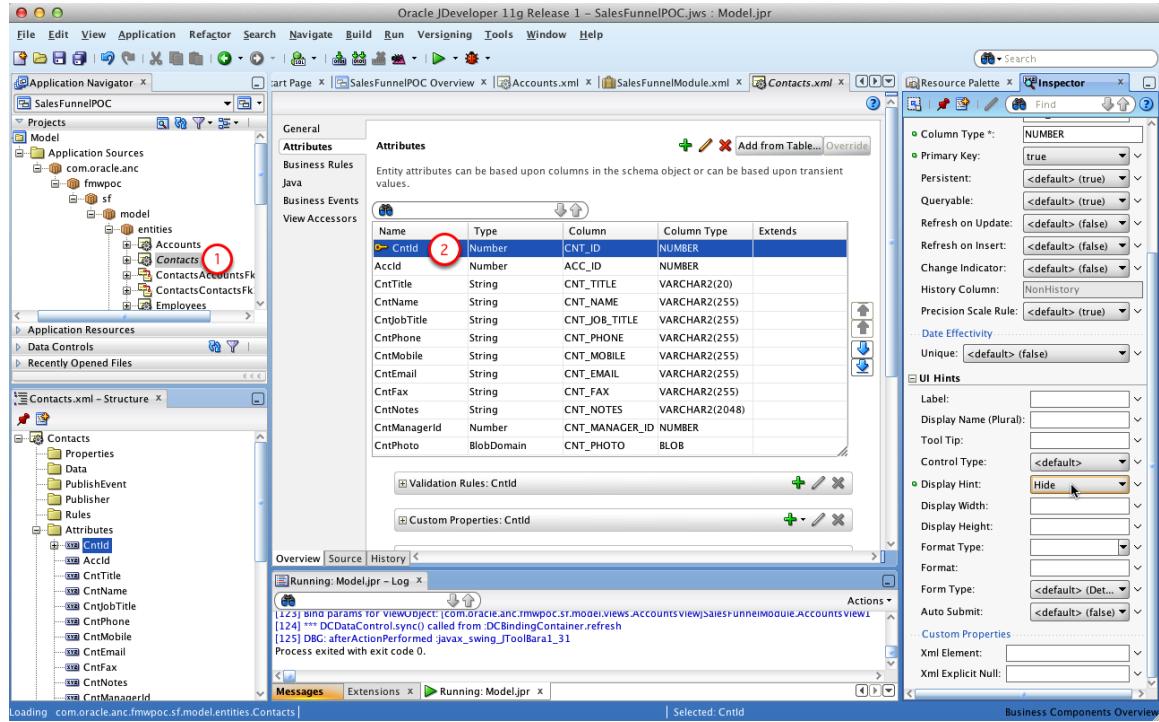


1. Select AccName
2. Set Label to "Account Name"

Set labels for all other fields in the same way.

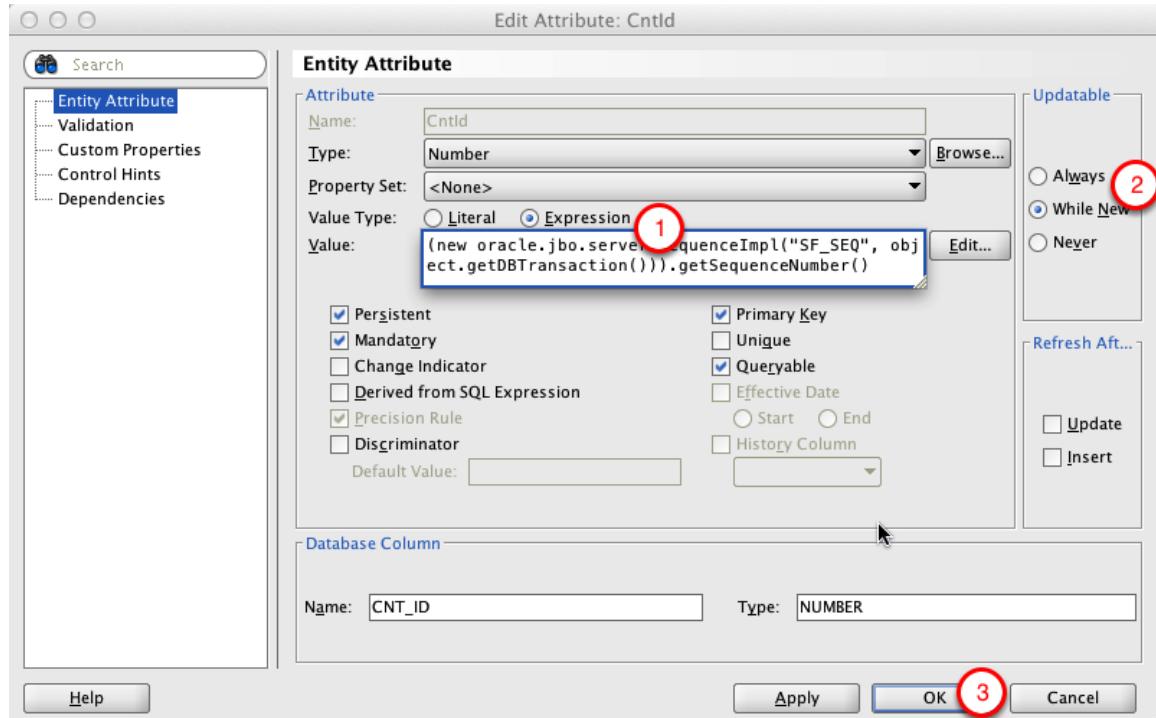
4. Setting Contacts entity properties

Select CntId Attribute



1. Open Contacts Entity
2. Double-click CntId attribute

Edit Attribute: CntId

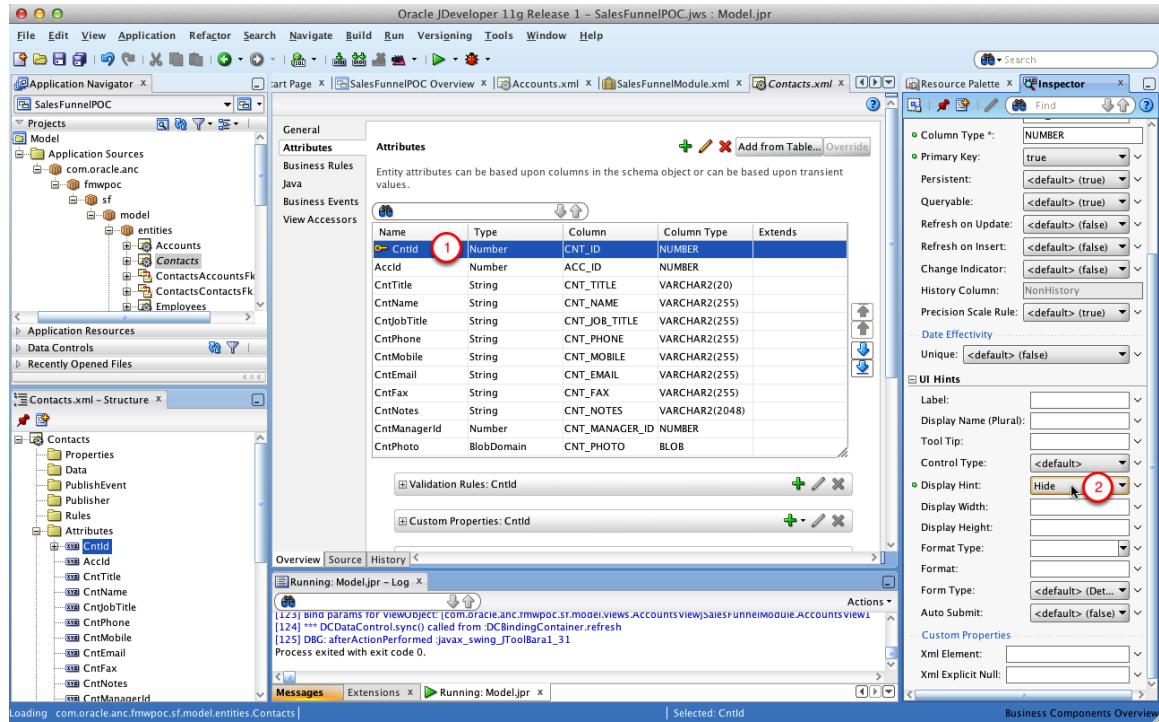


We will use DB sequences to populate ID fields.

1. Set Value Type to Expression and paste the below code in the Value field
2. Set Updatable to "While New"
3. Click OK

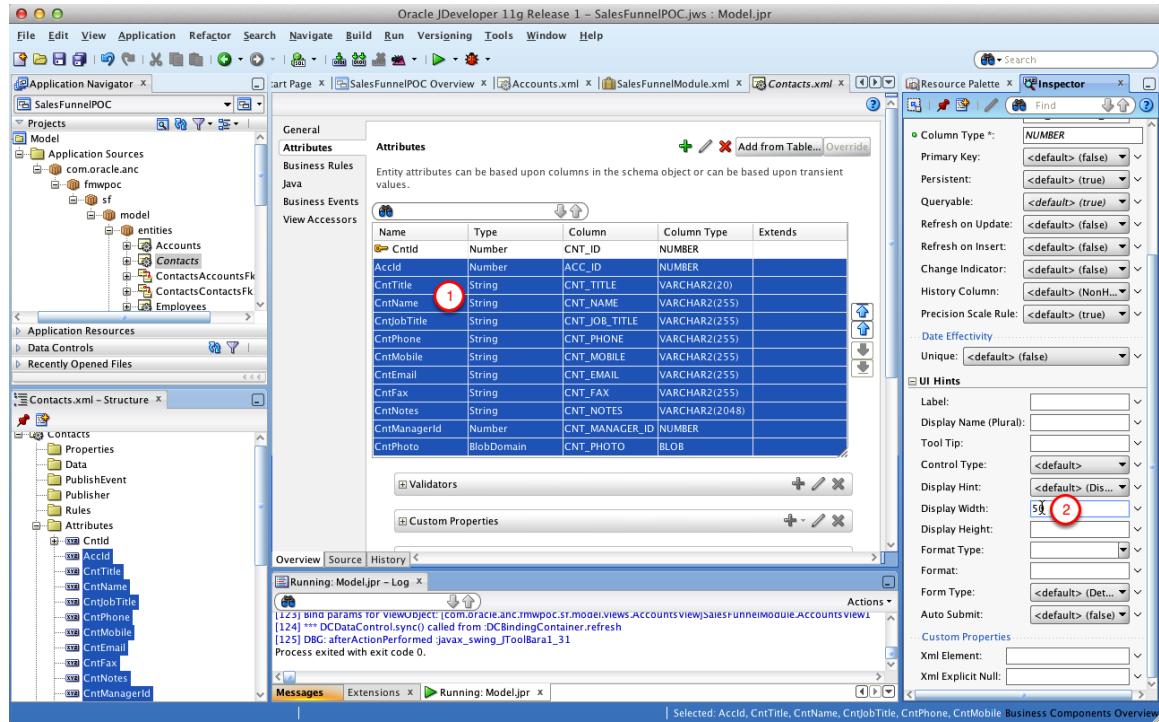
```
(new oracle.jbo.server.SequenceImpl("SF_SEQ",
object.getDBTransaction())).getSequenceNumber()
```

Edit Attribute: CntId



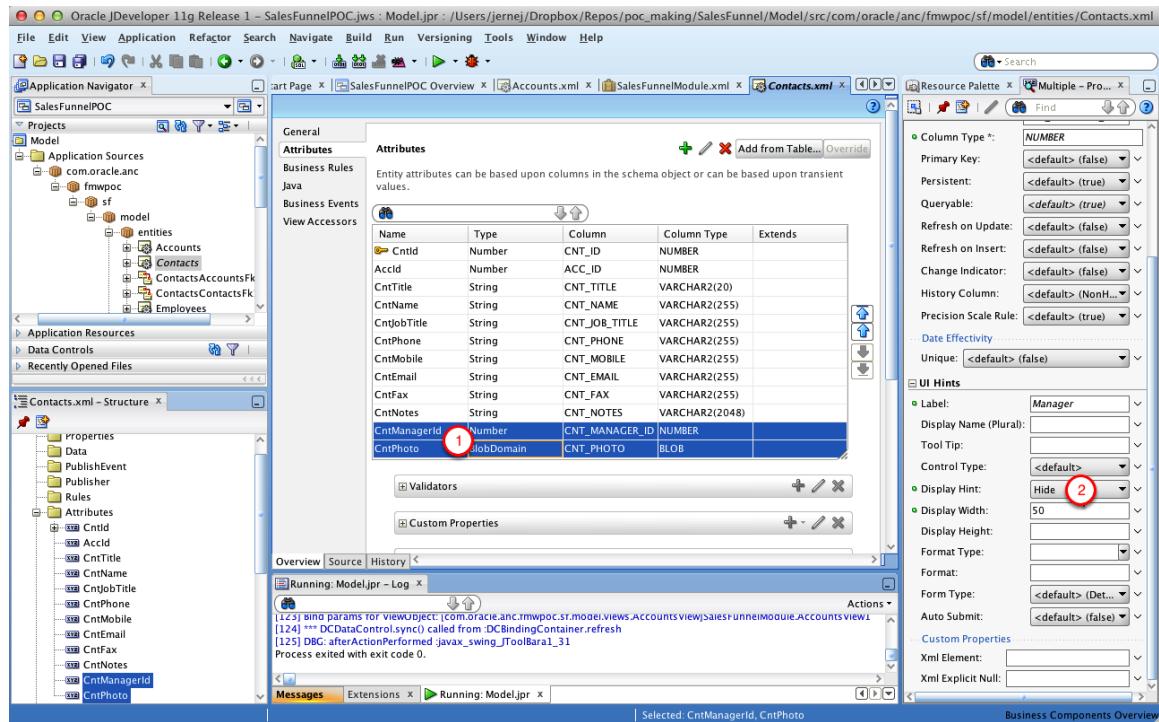
1. Select CntId attribute
2. Set Display Hint to Hide in Property Inspector

Set fields width



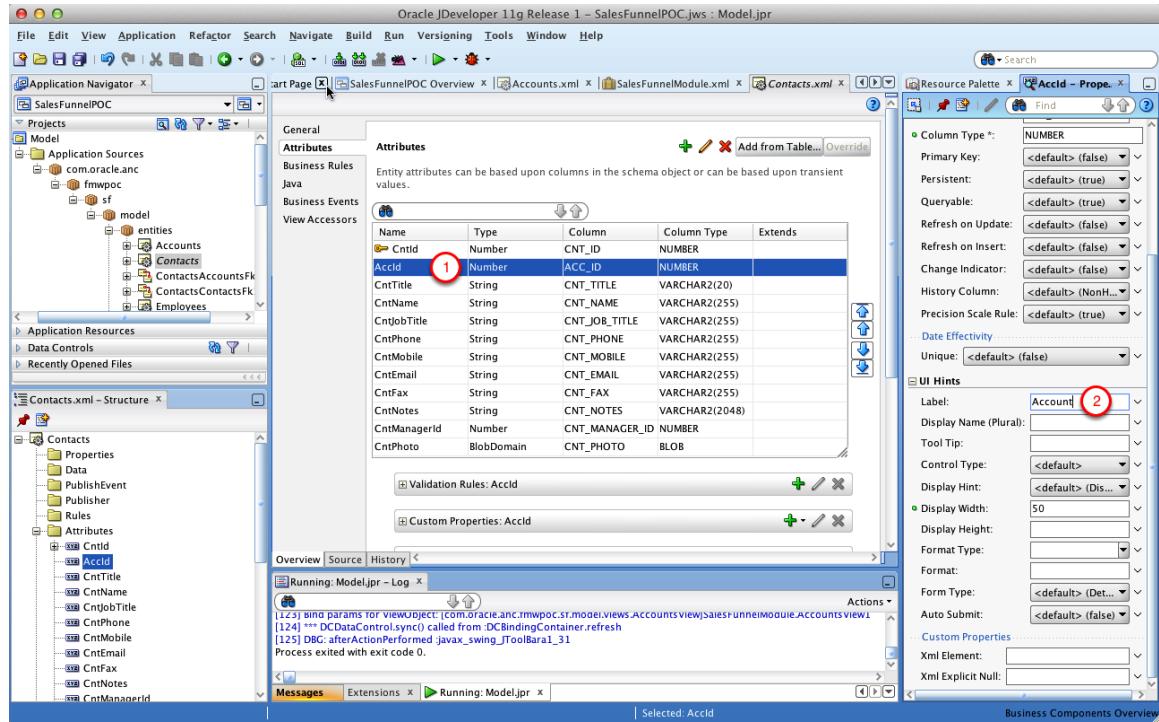
1. Select all attributes
2. Set Display Width to 50

Hide some unused fields



1. Select CntManagerId and CnPhoto attributes
2. Set Display Hint to Hide

Set field labels

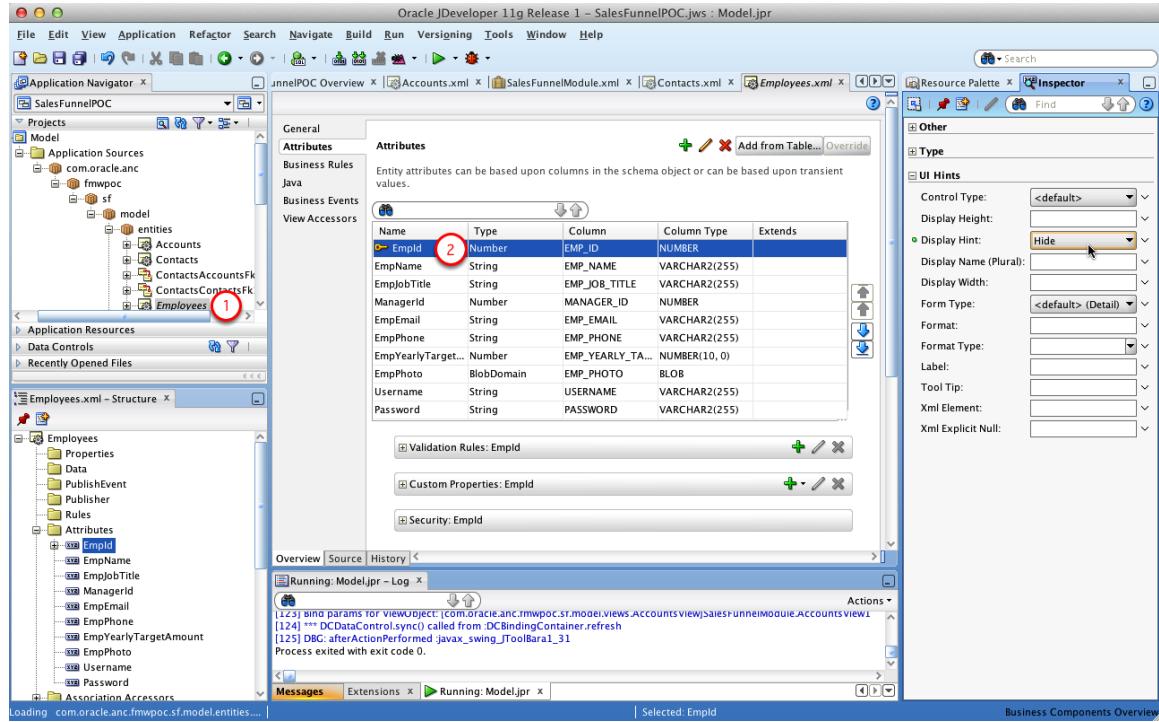


1. Select Acclid
2. Set Label to "Account"

Set labels for all other fields in the same way.

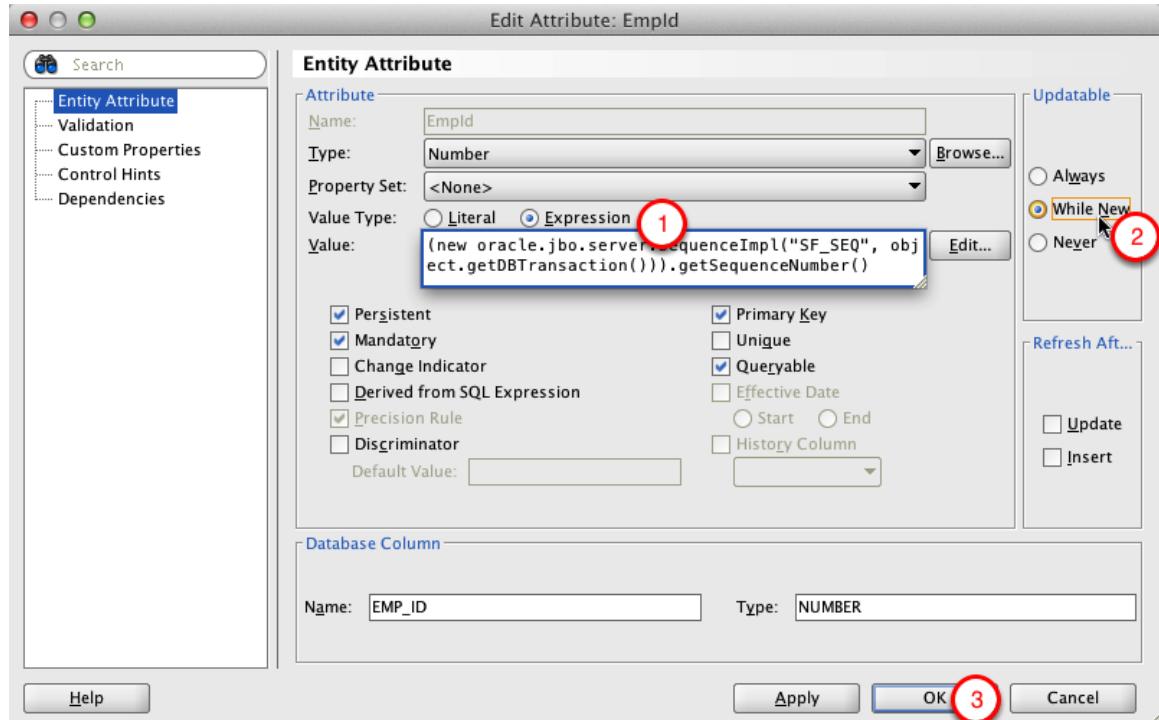
5. Setting Employee entity properties

Select EmpId Attribute



1. Open Employees Entity
2. Double-click EmpId attribute

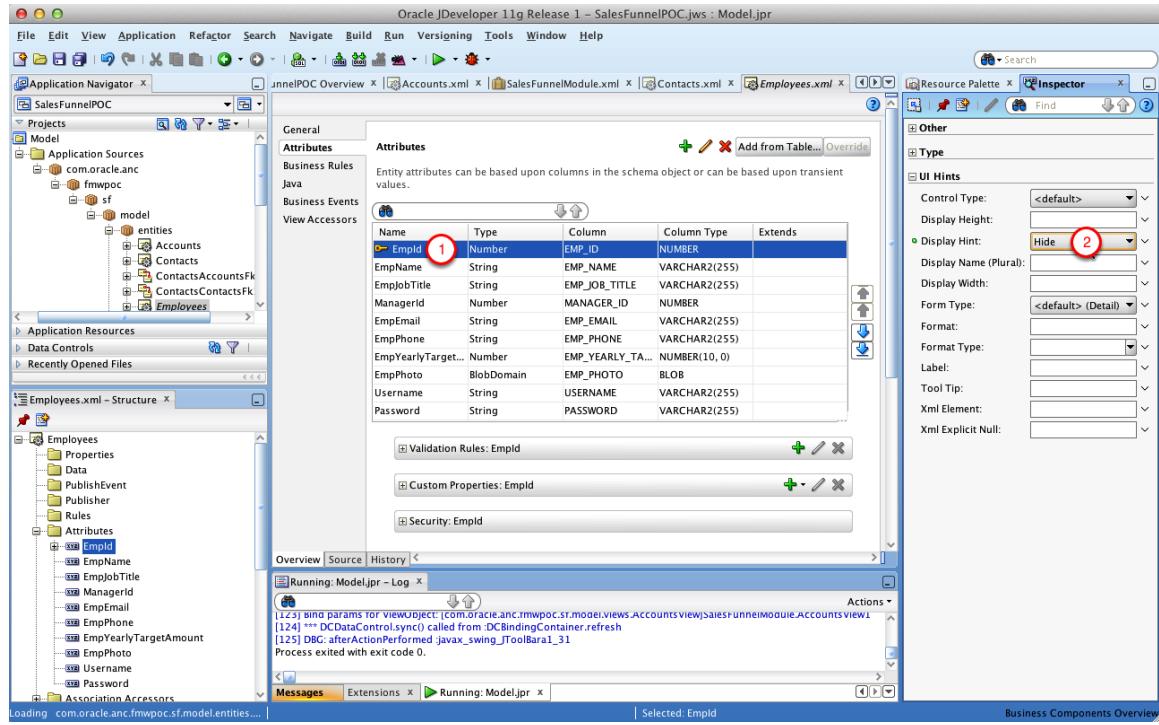
Edit Attribute: Empld



1. Set Value Type to Expression and paste the below code in the Value field
2. Set Updatable to "While New"
3. Click OK

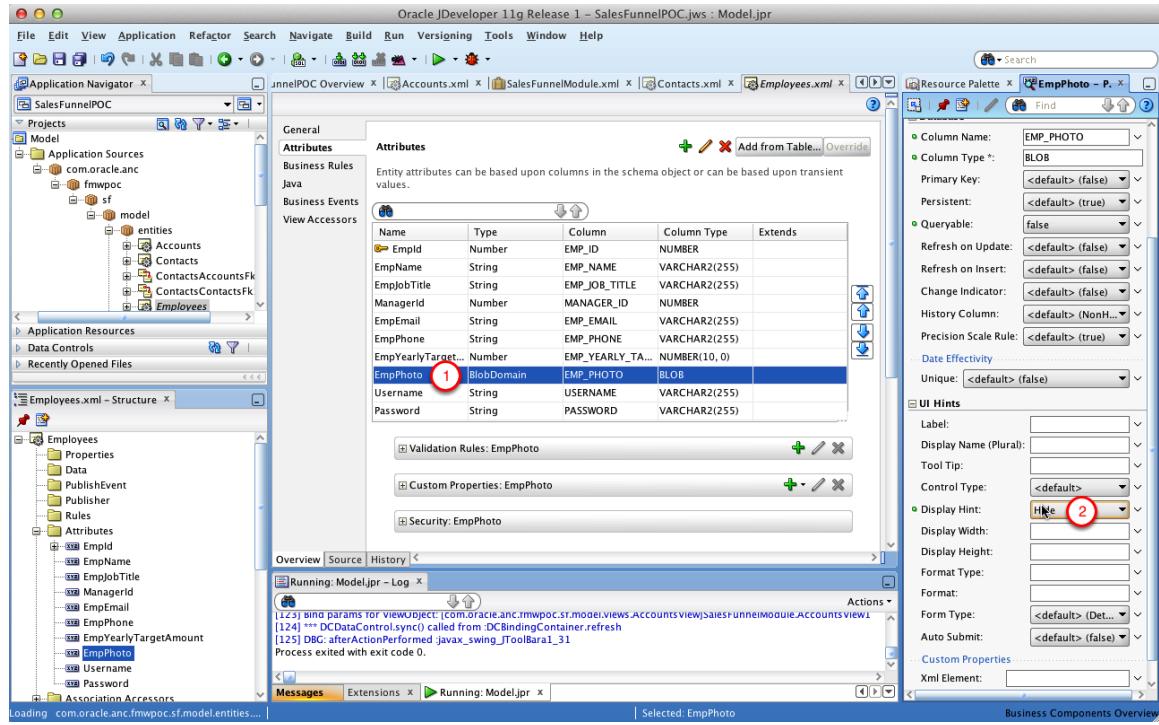
```
(new oracle.jbo.server.SequenceImpl("SF_SEQ",
object.getDBTransaction())).getSequenceNumber()
```

Edit Attribute: EmpId



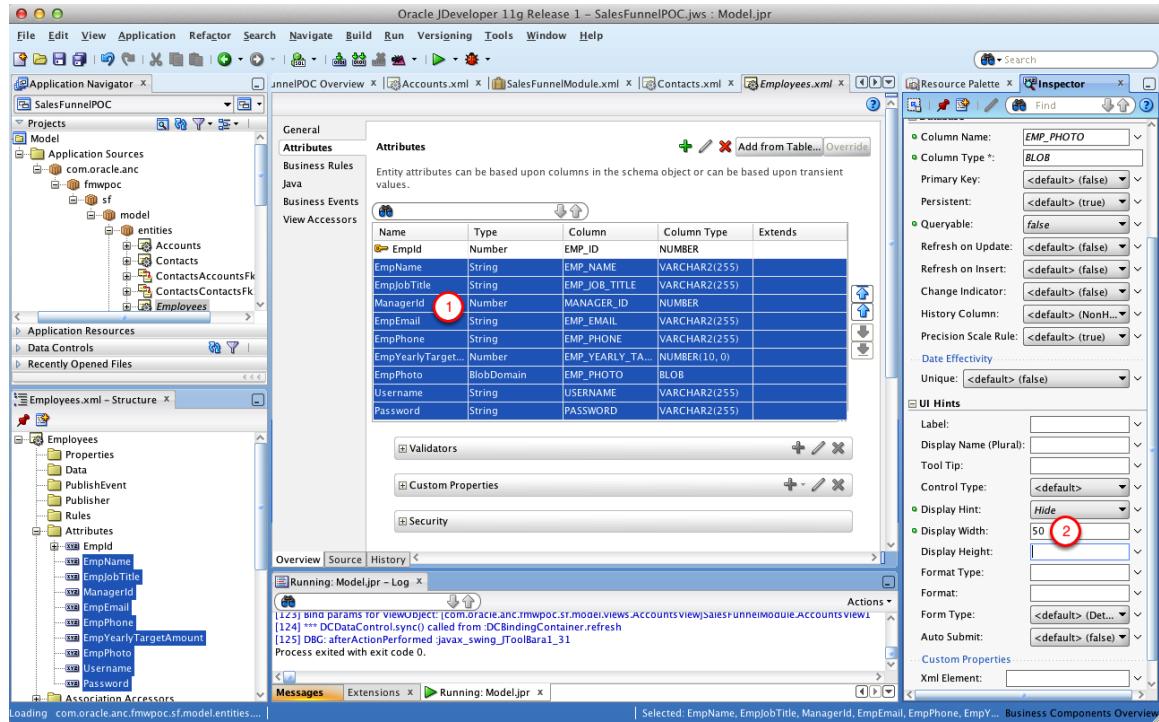
1. Select EmpId Attribute
2. Set Display Hint to Hide

Hide unused field



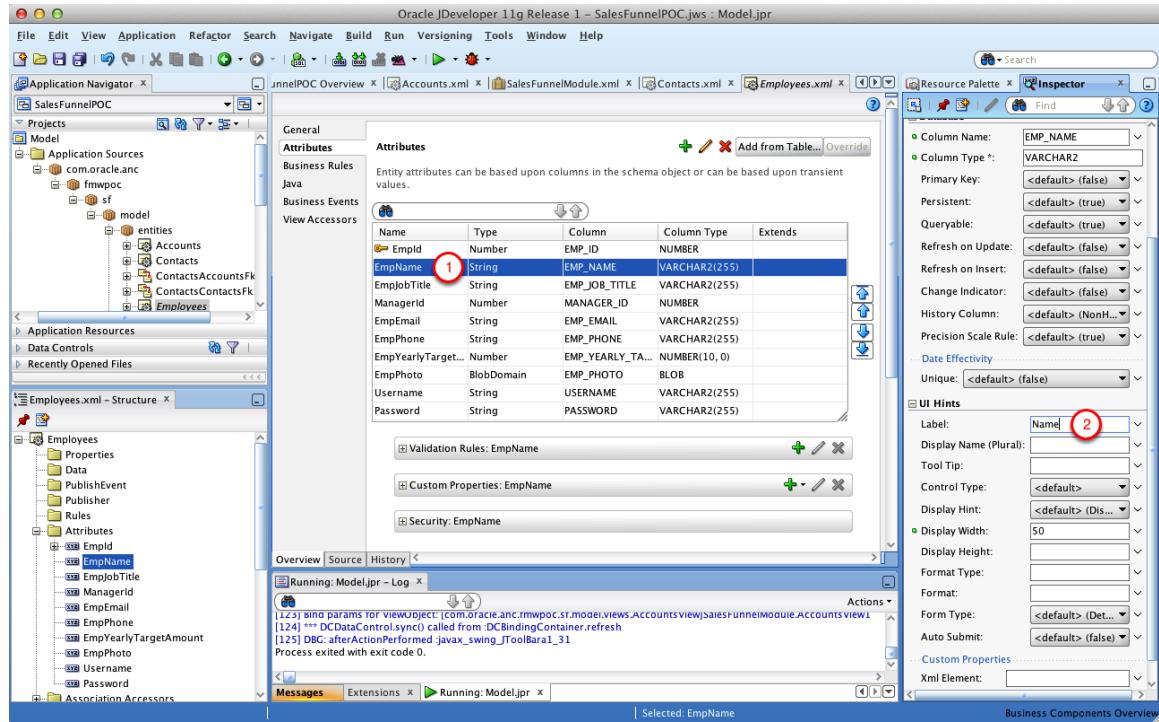
1. Select EmpPhoto attribute
2. Set Display Hint to Hide

Set fields width



1. Select all attributes
2. Set Display Width to 50

Set field labels

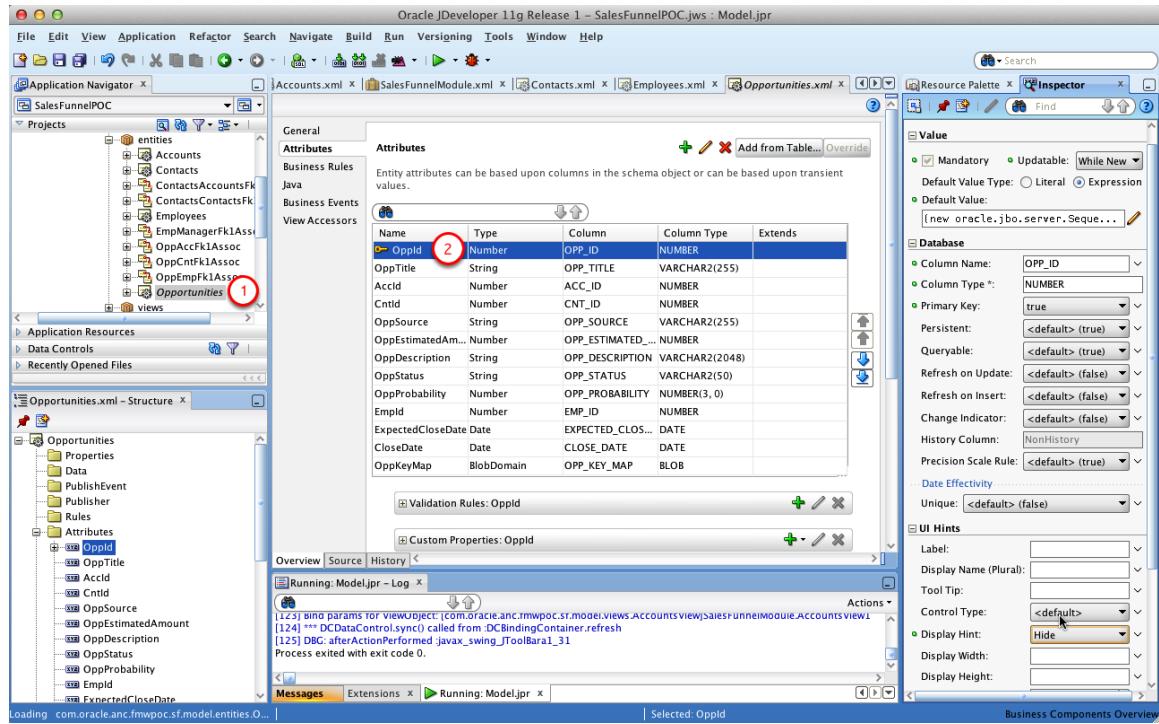


1. Select EmpName
2. Set Label to "Name"

Set labels for all other fields in the same way.

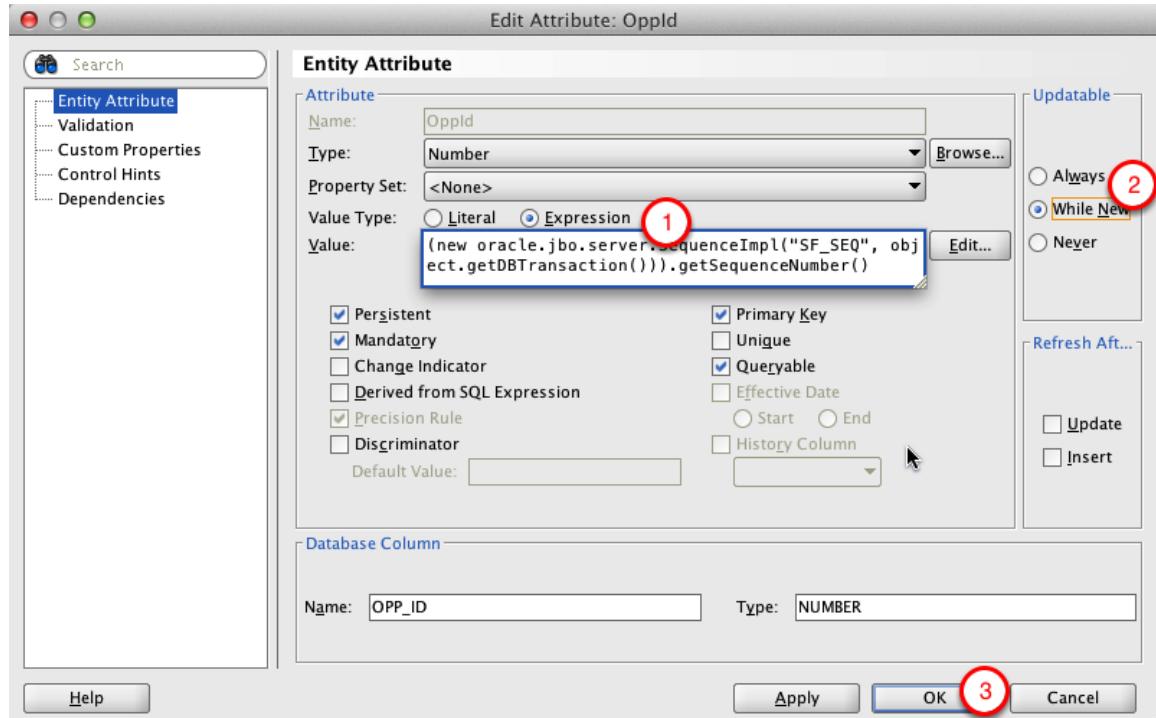
6. Setting Opportunity entity properties

Select OppId Attribute



1. Open Opportunities Entity
2. Double-click OppId attribute

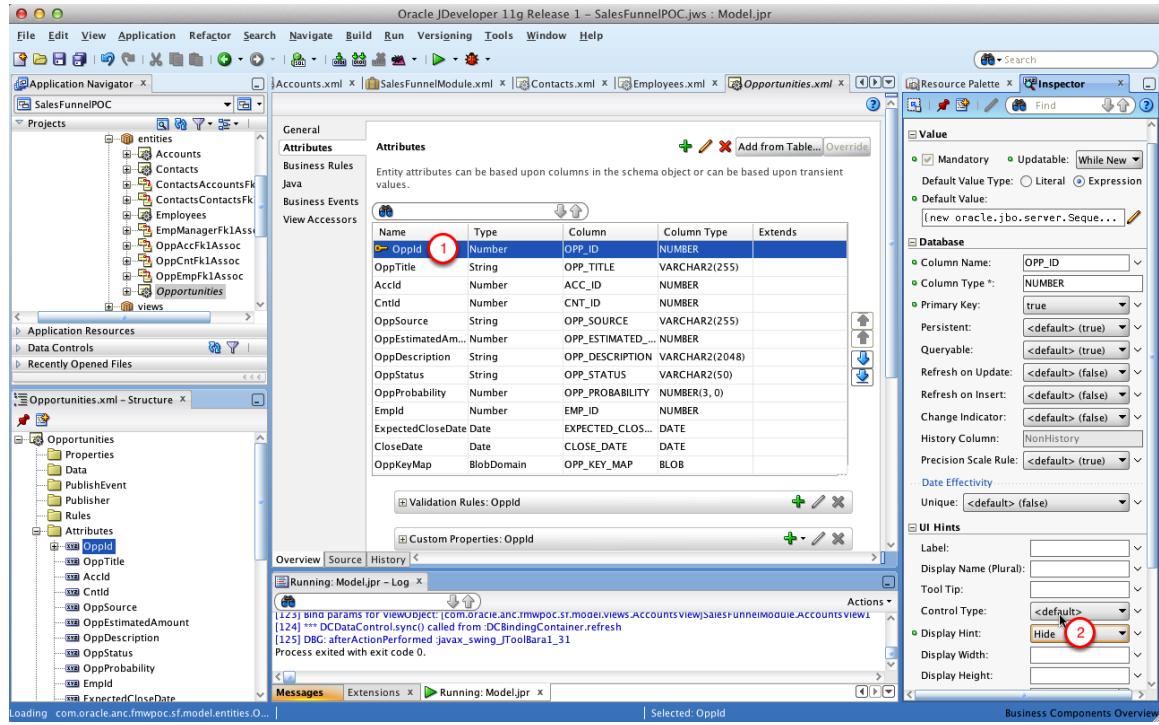
Edit Attribute: OppId



1. Set Value Type to Expression and paste the below code in the Value field
2. Set Updatable to "While New"
3. Click OK

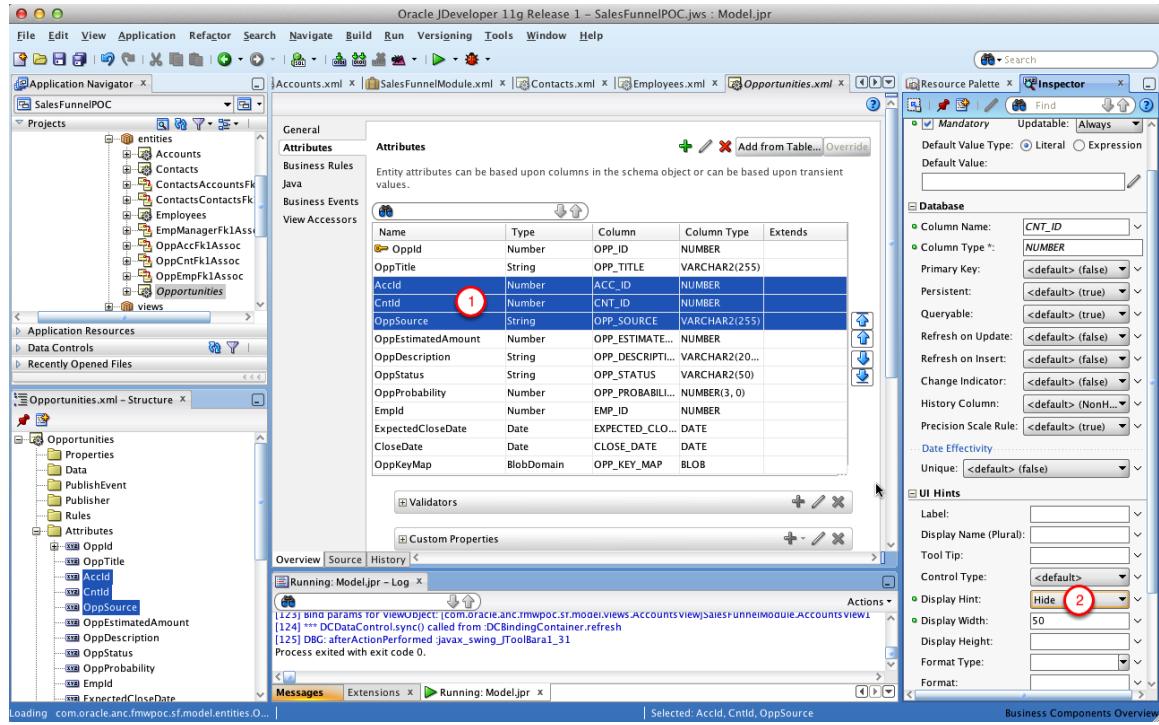
```
(new oracle.jbo.server.SequenceImpl("SF_SEQ",
object.getDBTransaction())).getSequenceNumber()
```

Edit Attribute: OppId



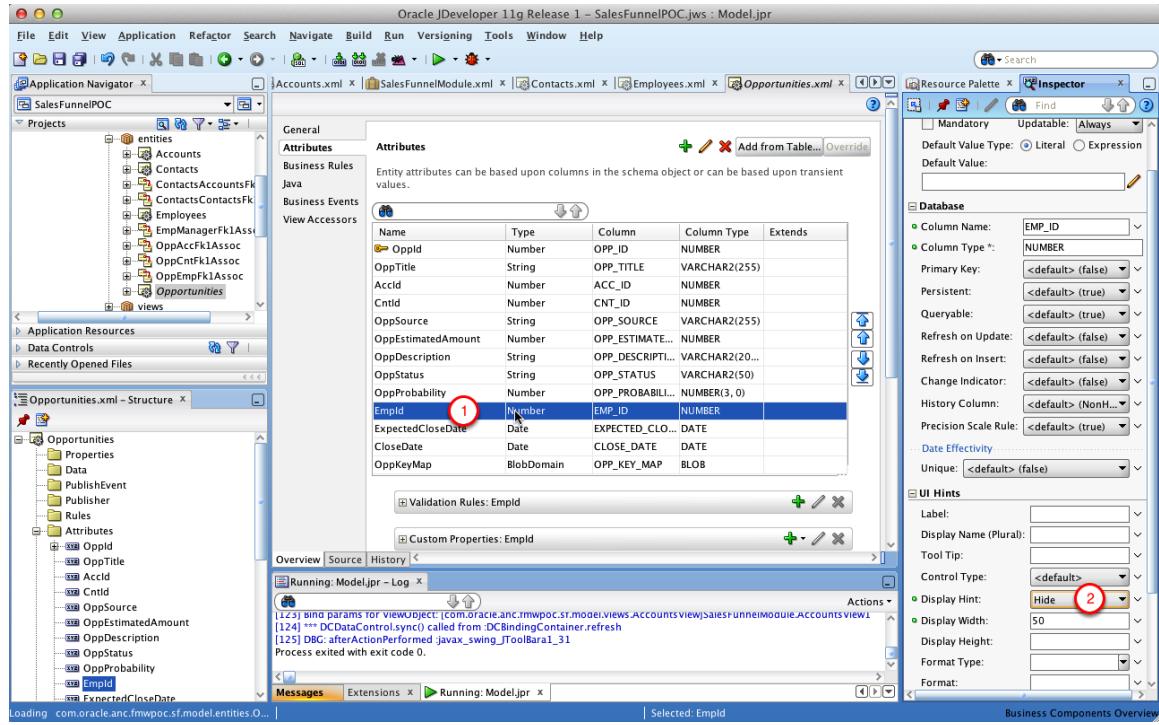
1. Select OppId attribute
2. Set Display Hint to Hide

Hide internal and unused fields



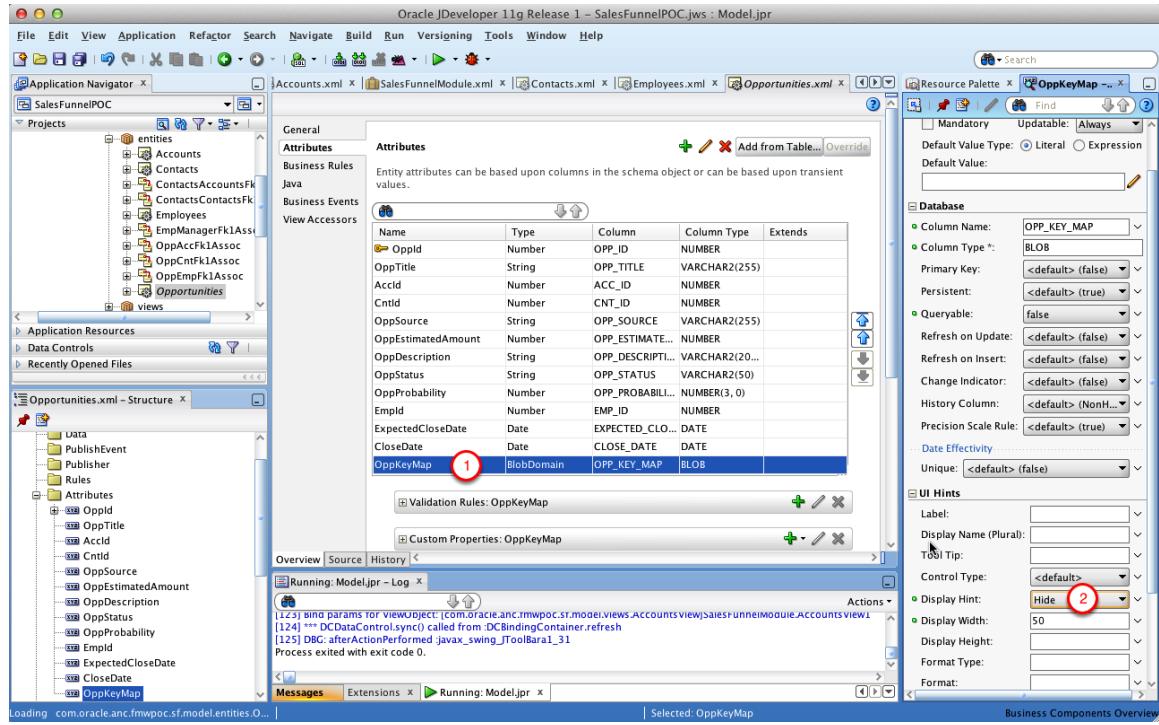
1. Select Acld, Cntld, OppSource attributes and set Display Hint to hide

Hide internal Empld field



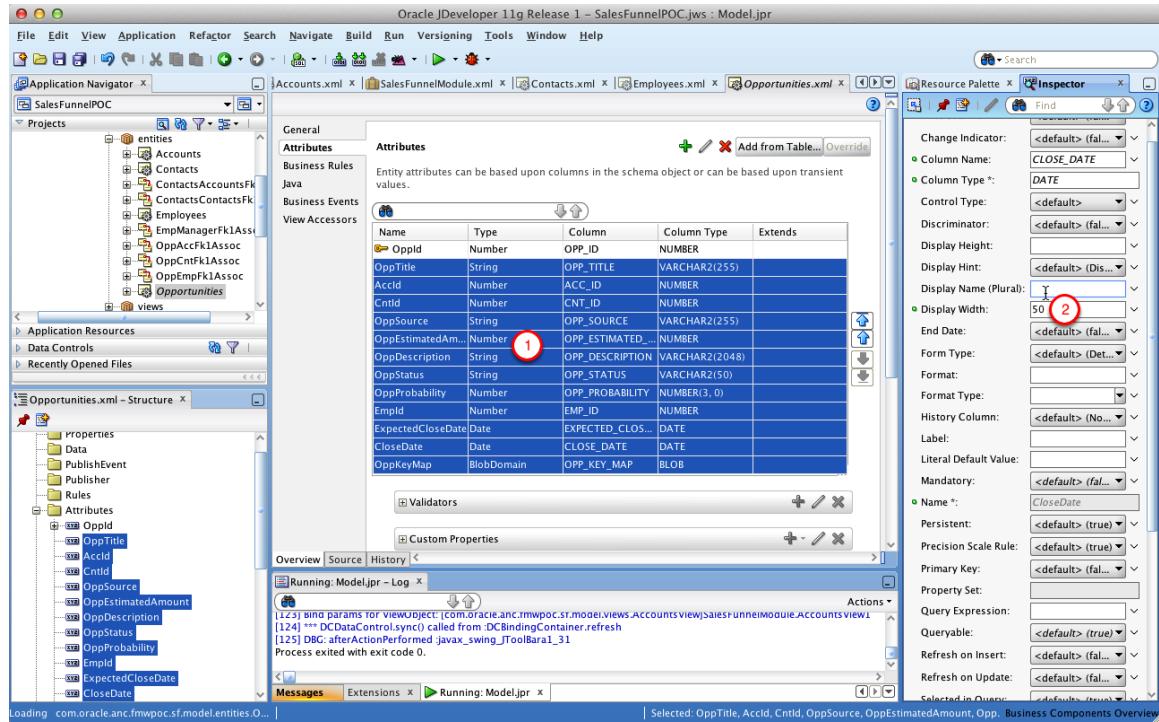
1. Select Empld attribute
2. Set Display Hint to Hide

Hide unused OppKeyMap field



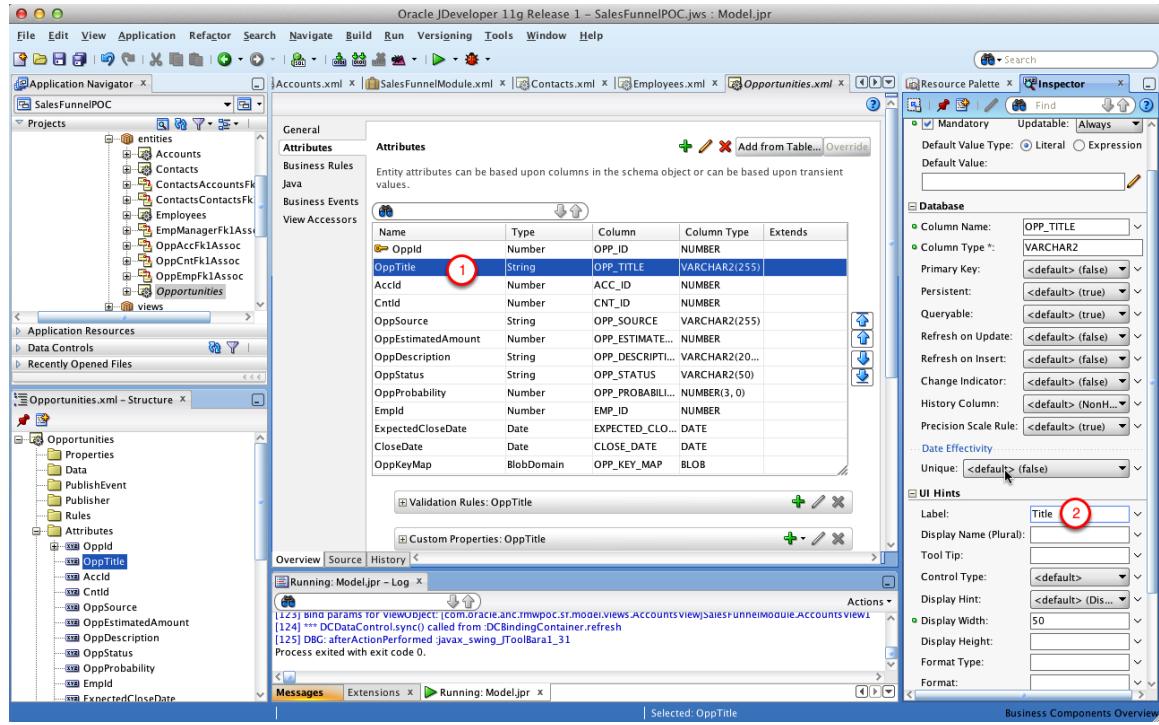
1. Select OppKeyMap attribute
2. Set Display Hint to Hide

Set fields width



1. Select all attributes
2. Set Display Width to 50

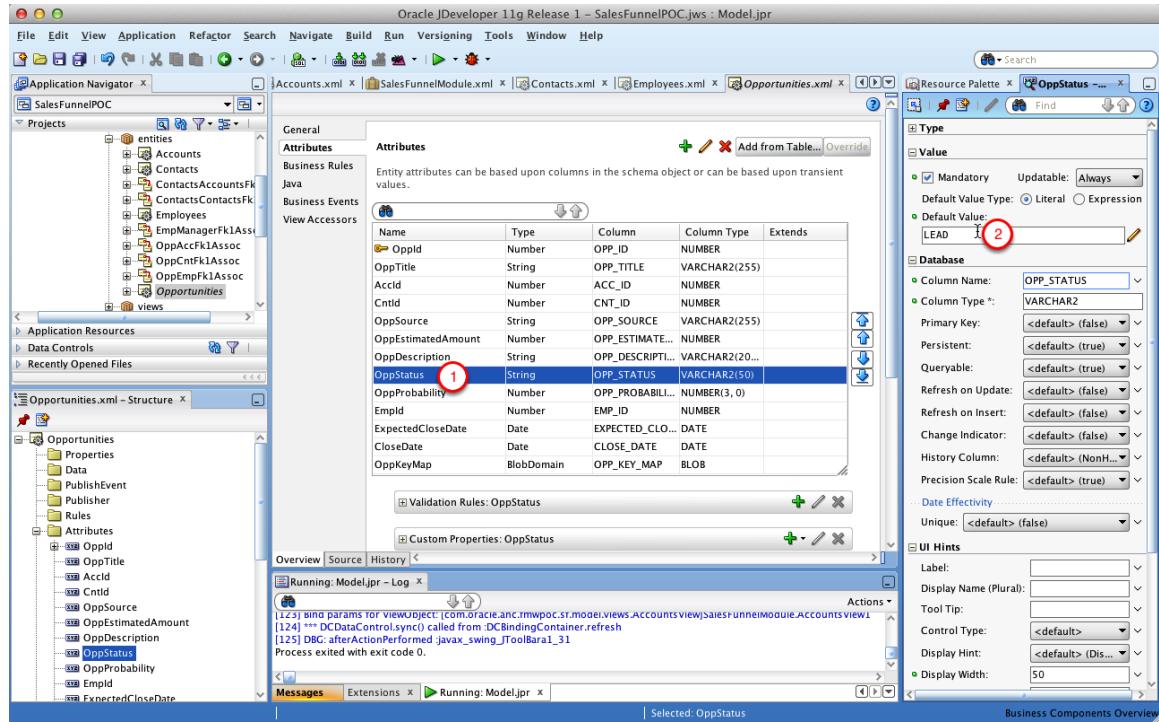
Set field labels



1. Select OppTitle attribute
2. Set Label to "Title"

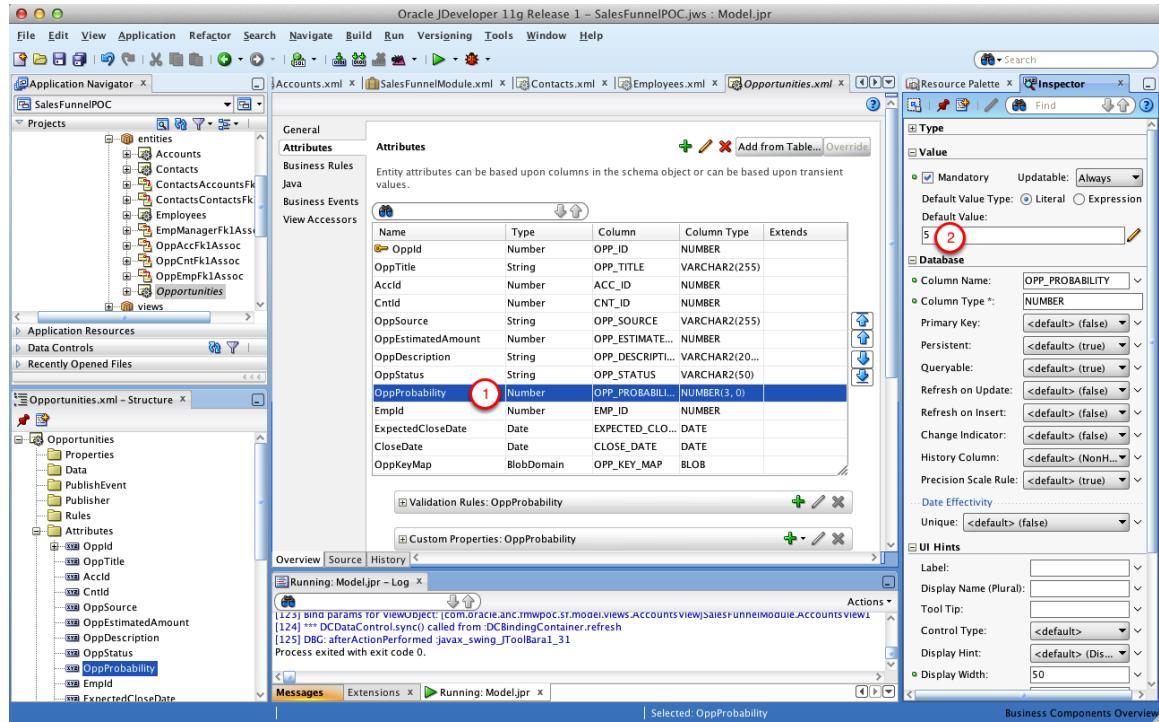
Set labels for all other fields in the same way.

Set OppStatus default value



1. Select OppStatus attribute
2. Set Default Value to "LEAD" (without quotes)

Set OppProbability default value



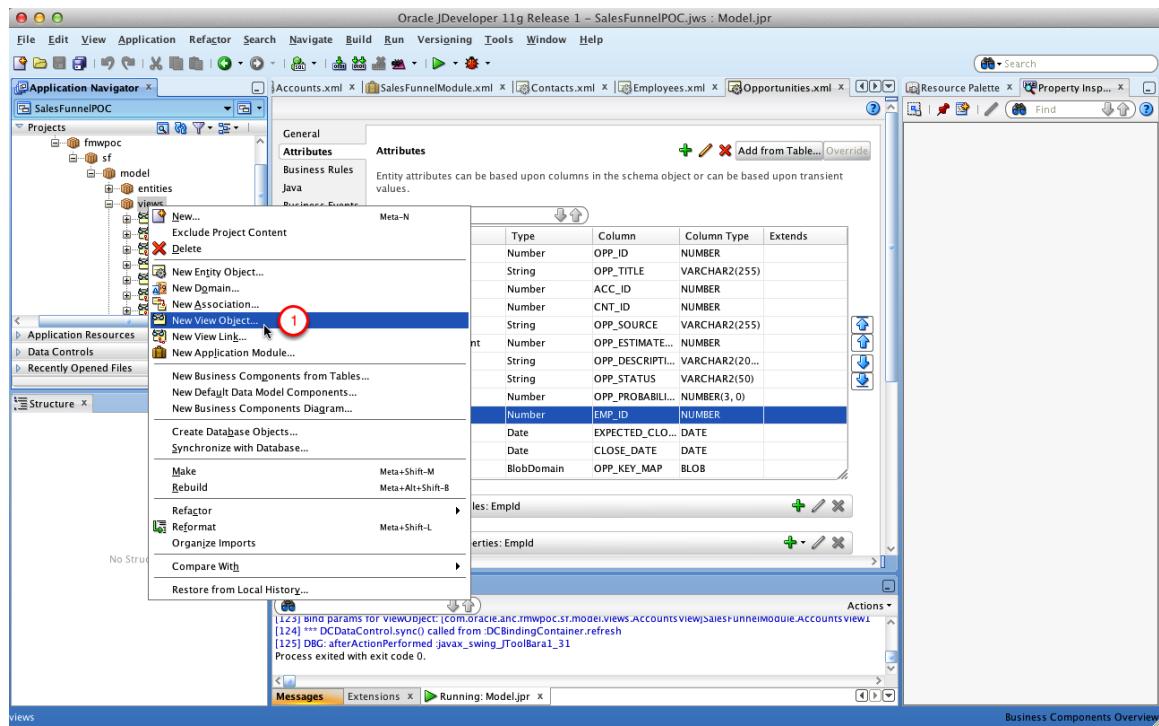
1. Select OppProbability attribute
2. Set Default Value to 5

7. Accounts LOV View

Lookup fields (commonly represented as combo boxes in the user interface) need a source data set to get values from.

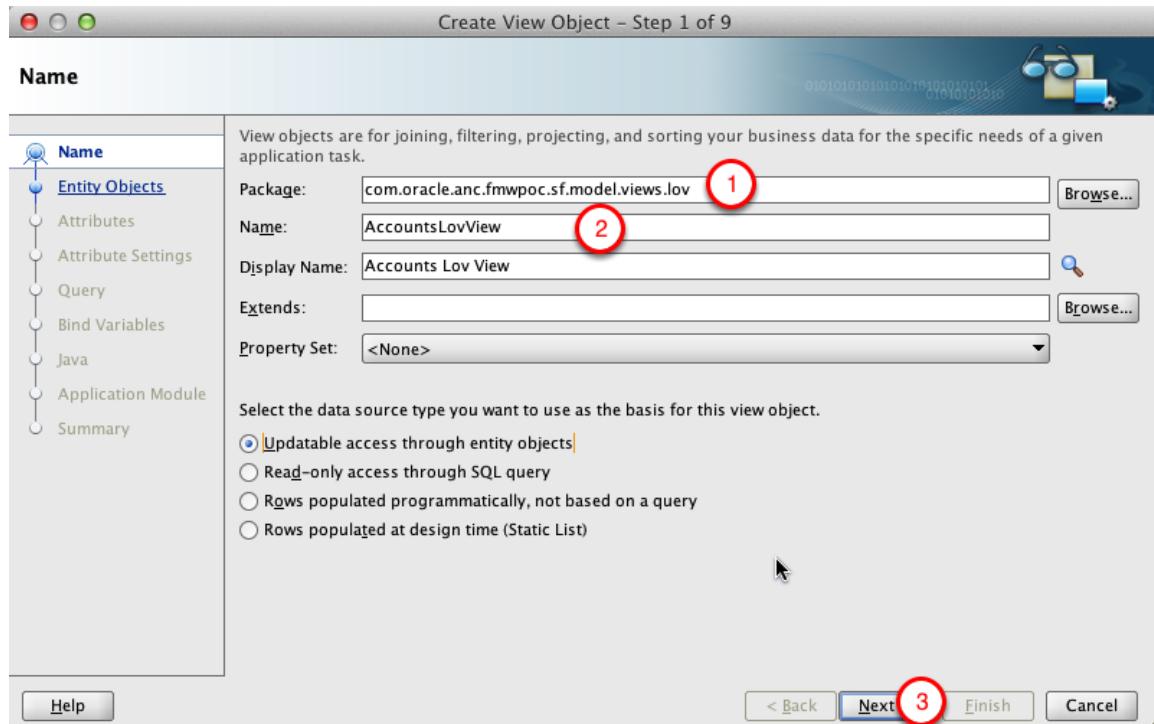
In this and the following two lessons we are going to create lean, cached data sources and use them for lookups.

Create new view object



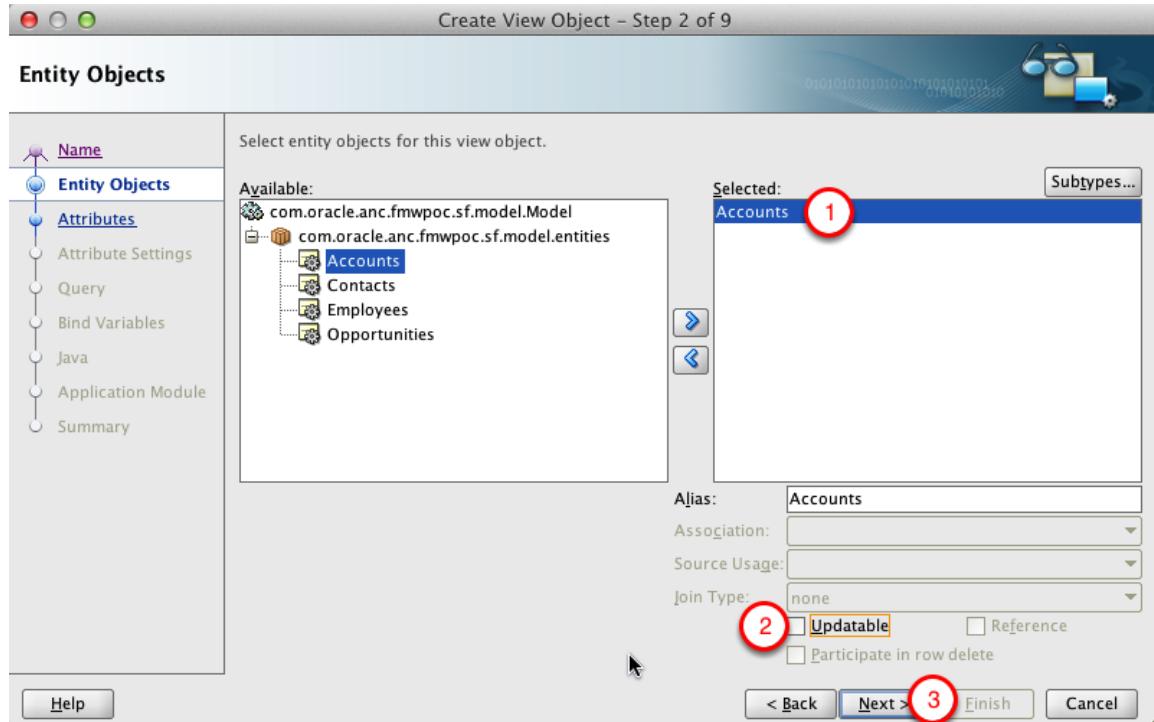
1. Right-click on the ...model.views package in the Application Navigator
2. Select New View Object from the popup menu

Create View Object - Step 1 of 9



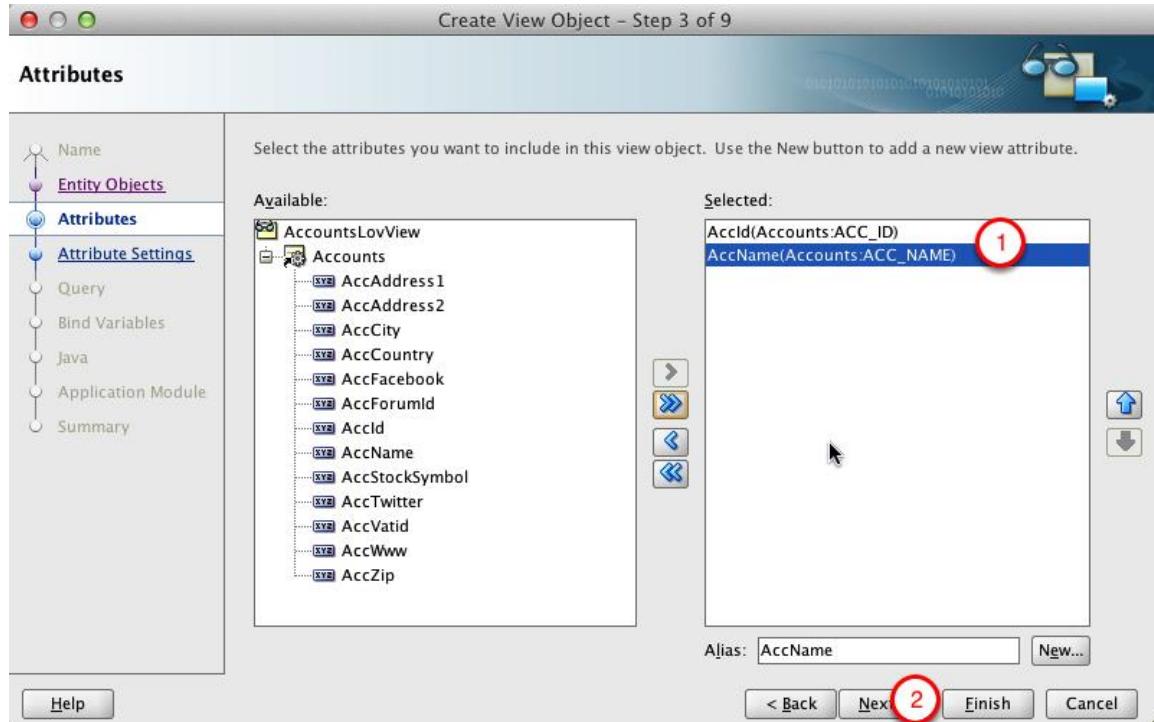
1. Append .lov to the Package name
2. Set Name to AccountsLovView
3. Click Next

Create View Object - Step 2 of 9



1. Select and move Accounts Entity to the right using Shuttle
2. Uncheck Updatable checkbox
3. Click Next

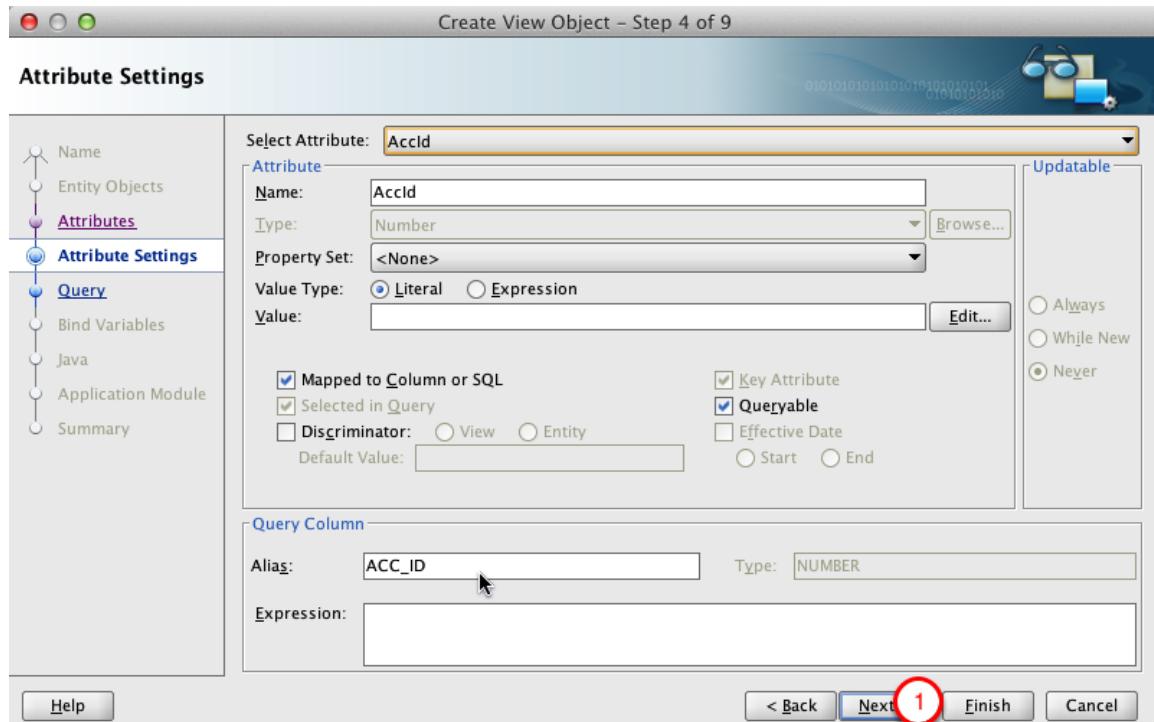
Create View Object - Step 3 of 9



1. Select AccName attribute and move it to the right using Shuttle*
2. Click Next

*AccId will be added automatically

Create View Object - Step 4 of 9



1. Click Next

Create View Object - Step 5 of 9

Create View Object – Step 5 of 9

Query

By default, the SELECT list and FROM clause are automatically maintained. To override this mechanism, select Expert Mode.

Generated Statement

```
SELECT Accounts.ACC_ID,
       Accounts.ACC_NAME
  FROM ACCOUNTS Accounts
 ORDER BY ACC_NAME
```

Query Clauses

Where:

Order By: ACC_NAME **1**

Binding Style: Oracle Named

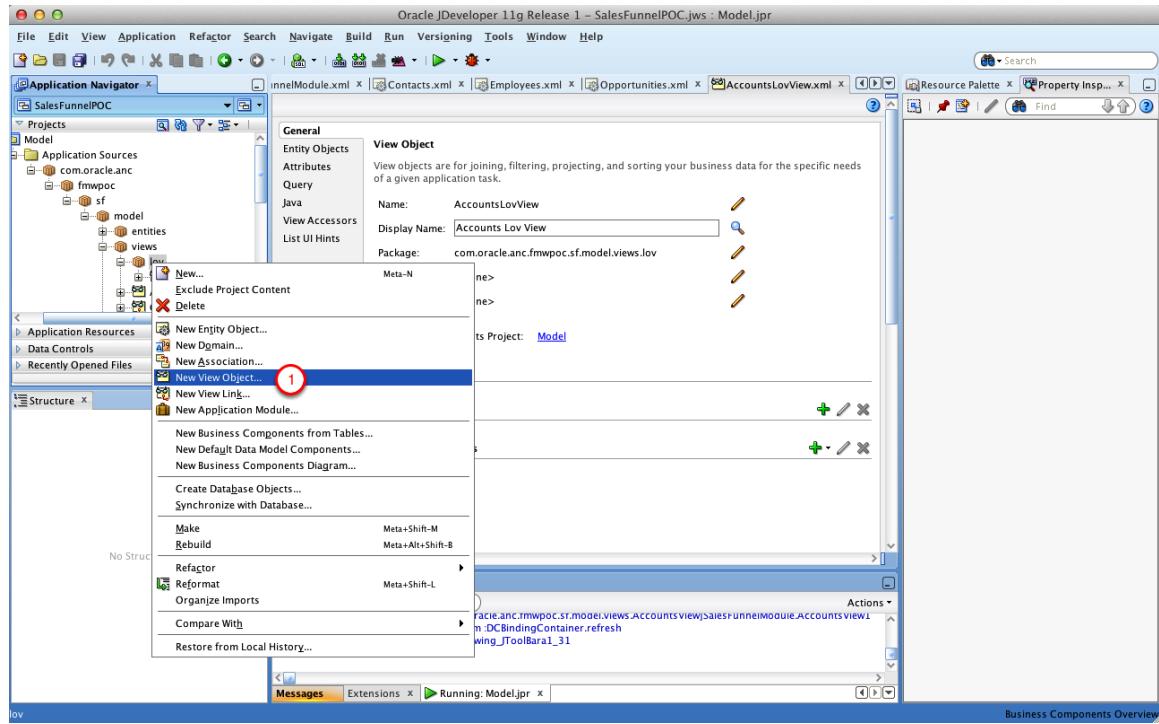
SQL Mode: Normal

Cancel

1. Set Order By to ACC_NAME
2. Click Finish

8. Contacts LOV View

Create new view object



1. Right-click ...model.view.lov and click New View Object in the popup menu

Create View Object - Step 1 of 9

Create View Object – Step 1 of 9

Name

View objects are for joining, filtering, projecting, and sorting your business data for the specific needs of a given application task.

Name	<input type="text" value="com.oracle.anc.fmw poc.sf.model.views.lov"/> 1	Browse...
Entity Objects	<input type="radio"/> Attributes	
	<input type="radio"/> Attribute Settings	
	<input type="radio"/> Query	
	<input type="radio"/> Bind Variables	
	<input type="radio"/> Java	
	<input type="radio"/> Application Module	
	<input type="radio"/> Summary	

Display Name:

Extends: **Browse...**

Property Set:

Select the data source type you want to use as the basis for this view object.

Updatable access through entity objects

Read-only access through SQL query

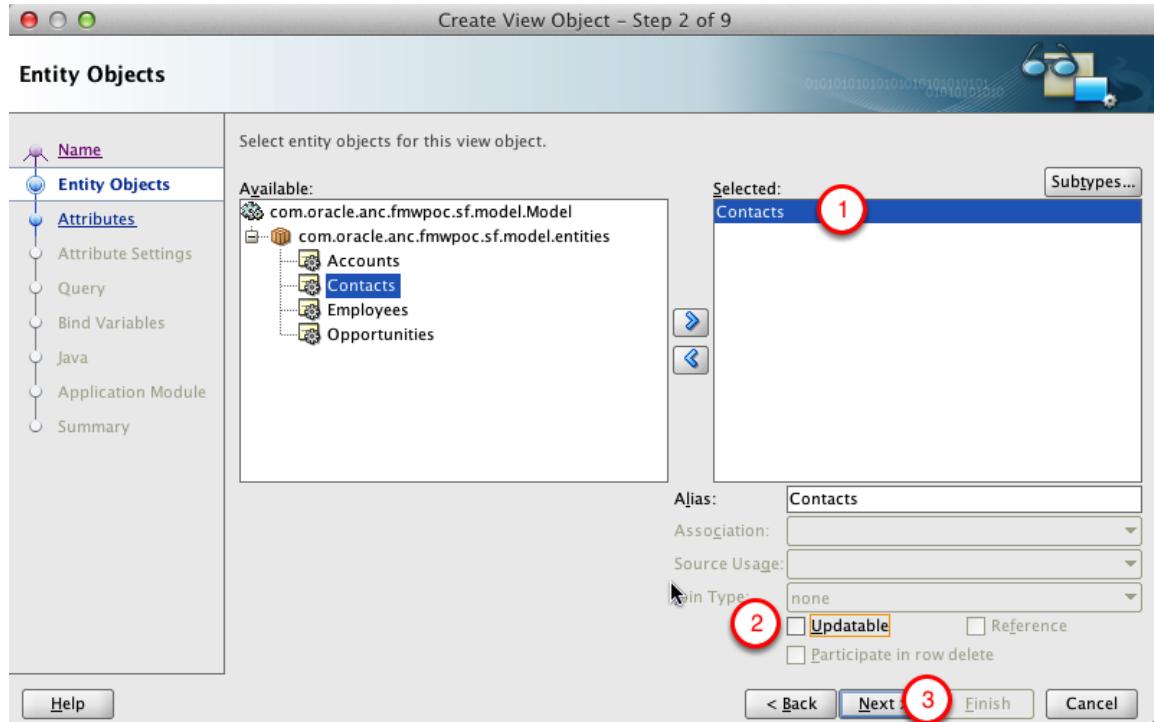
Rows populated programmatically, not based on a query

Rows populated at design time (Static List)

Help **Next** **2** **Finish** **Cancel**

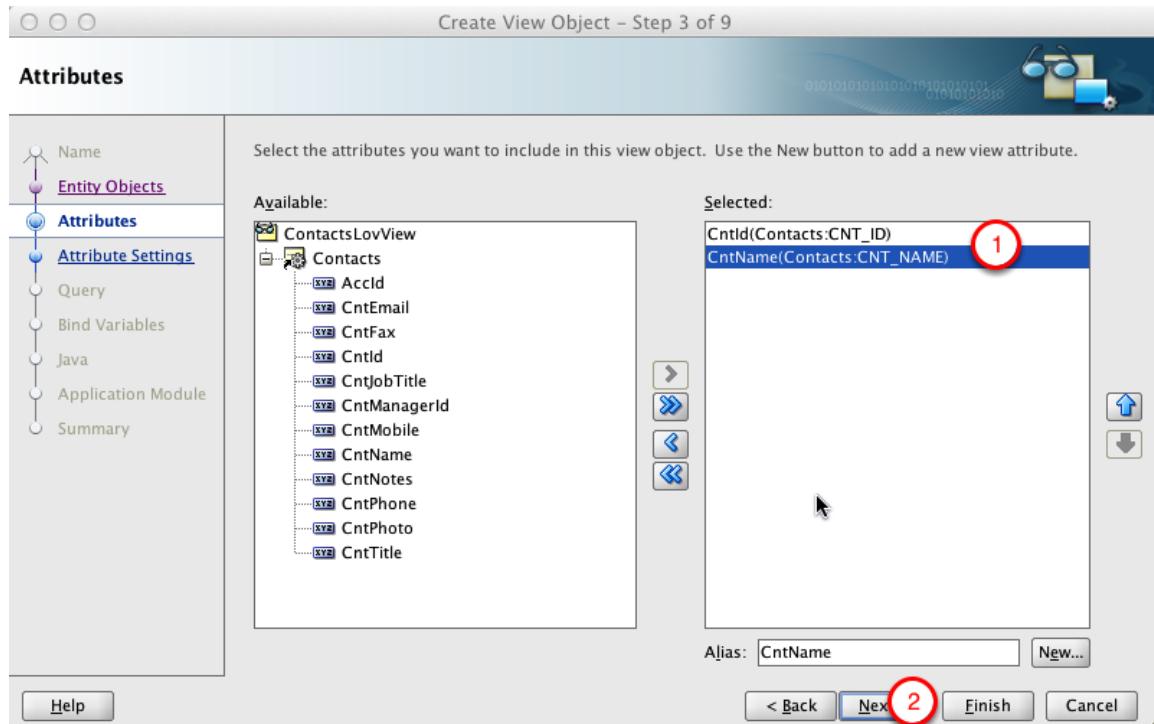
1. Set Name to ContactsLovView
2. Click Next

Create View Object - Step 2 of 9



1. Select and move Contacts entity to the right using Shuttle
2. Uncheck Updatable checkbox
3. Click Next

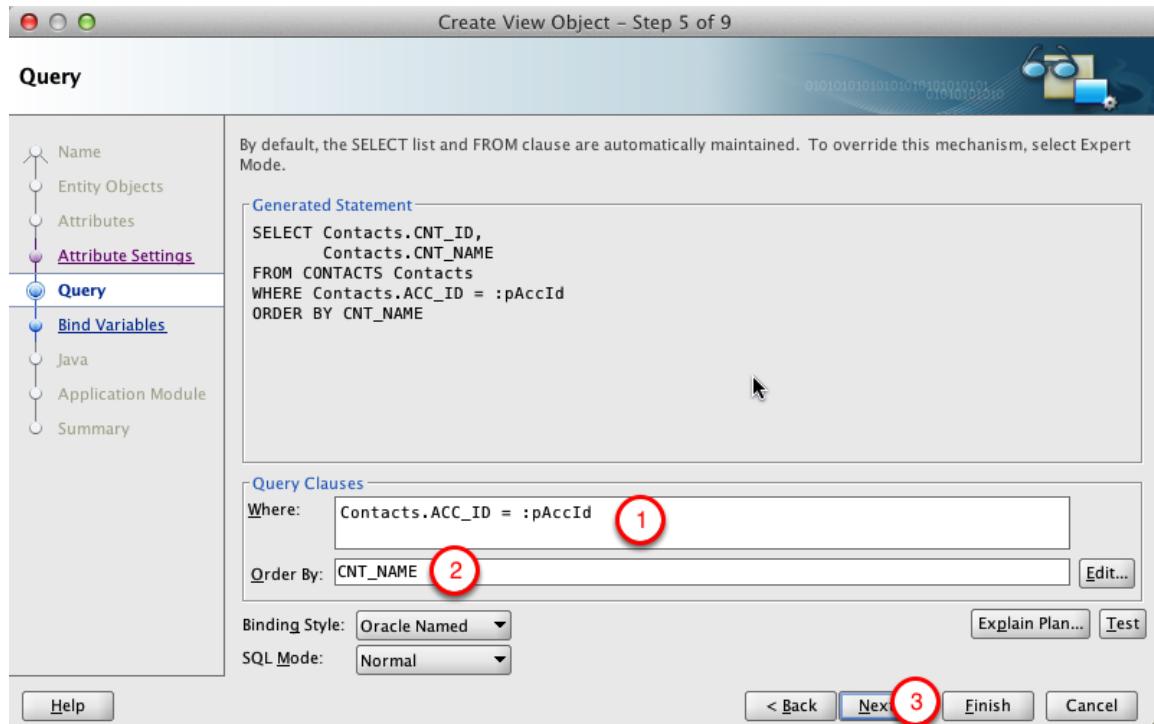
Create View Object - Step 3 of 9



1. Select and move CntName attribute to the right using Shuttle*
2. Click Next

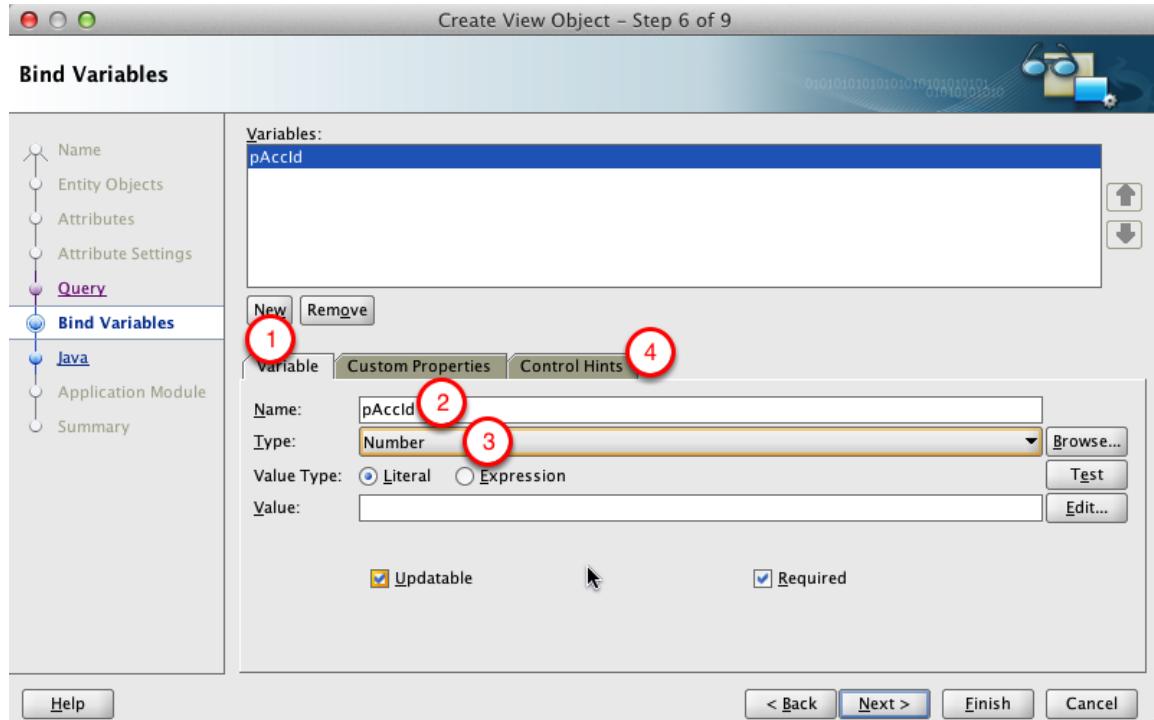
*CntId will be added automatically

Create View Object - Step 5 of 9



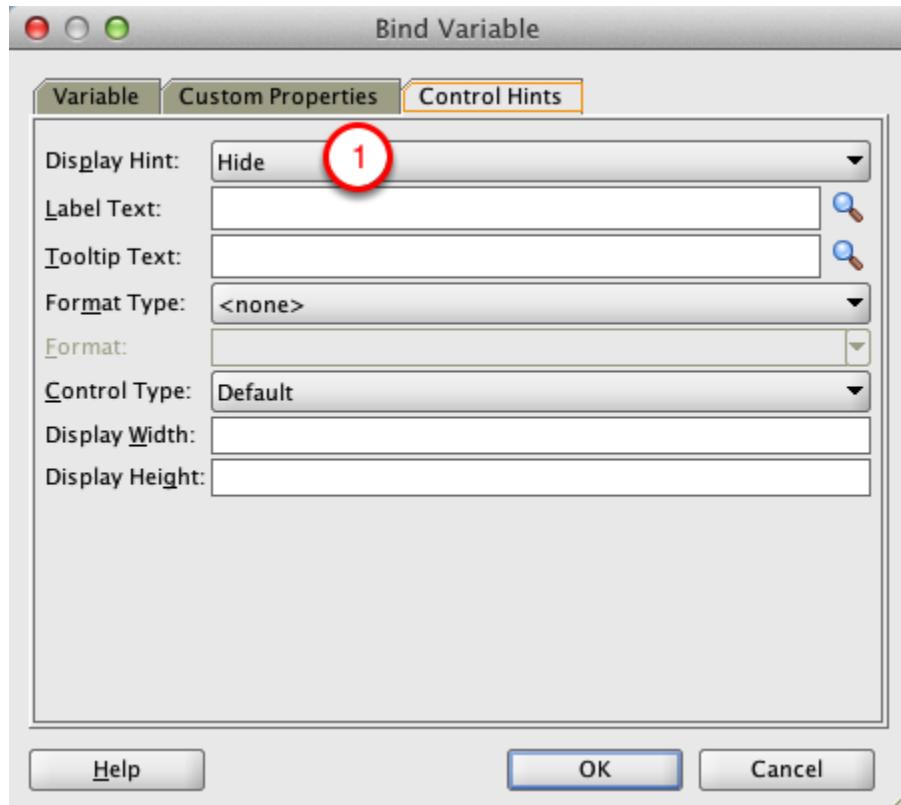
1. Set Where to "Contacts.ACC_ID = :pAccId" without the quotes
2. Set Order By to CNT_NAME
3. Click Next

Create View Object - Step 6 of 9



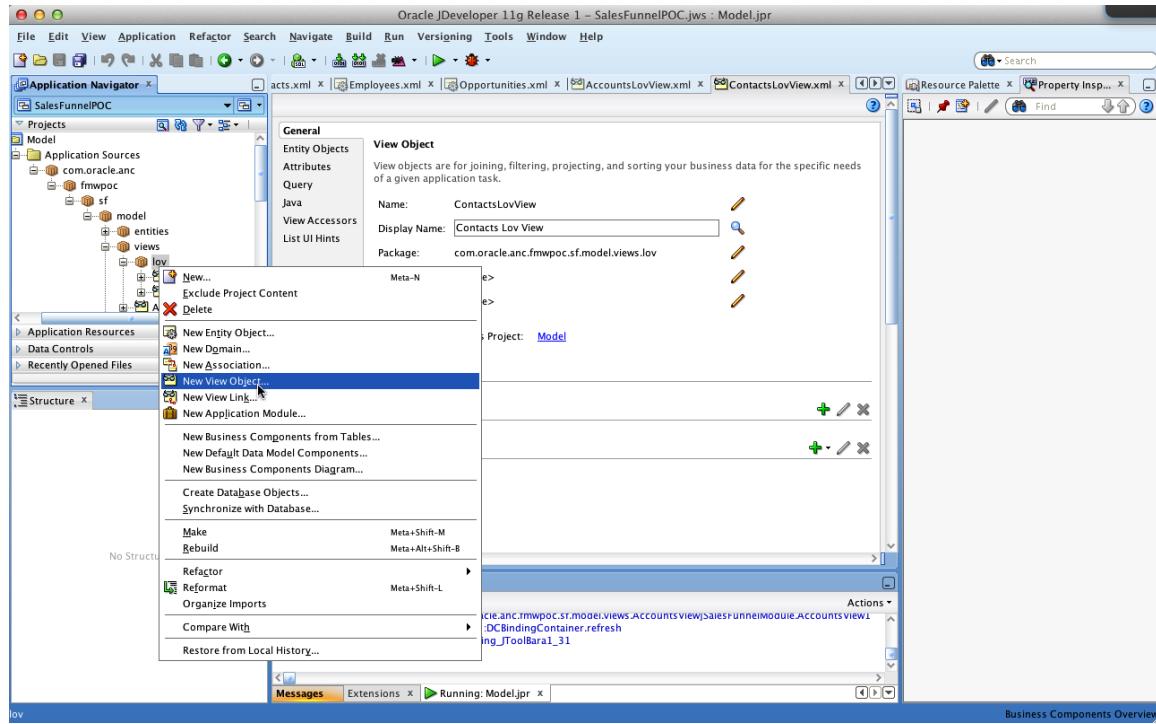
1. Click New to create new bind variable
2. Set Name to pAccId
3. Set Type to Number
4. Open Control Hints tab

Bind Variable



9. Employees LOV View

Create new view object



1. Right-click ...model.views.lov and select New View Object in the popup menu

Create View Object - Step 1 of 9

Create View Object – Step 1 of 9

Name

View objects are for joining, filtering, projecting, and sorting your business data for the specific needs of a given application task.

Name	<input type="text" value="EmployeesLovView"/> 1	Package: <input type="text" value="com.oracle.anc.fmw poc.sf.model.views.lov"/> Browse...
Entity Objects		
Attributes		
Attribute Settings		
Query		
Bind Variables		
Java		
Application Module		
Summary		

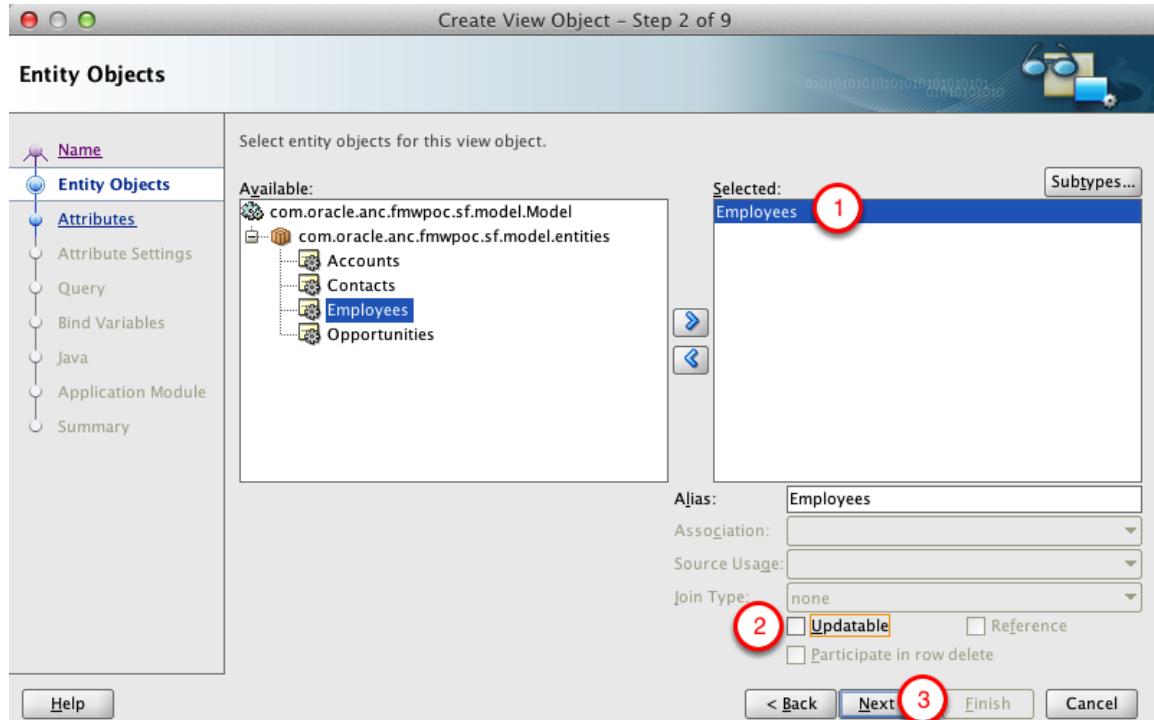
Select the data source type you want to use as the basis for this view object.

Updatable access through entity objects
 Read-only access through SQL query
 Rows populated programmatically, not based on a query
 Rows populated at design time (Static List)

Help < Back **Next** 2 Finish Cancel

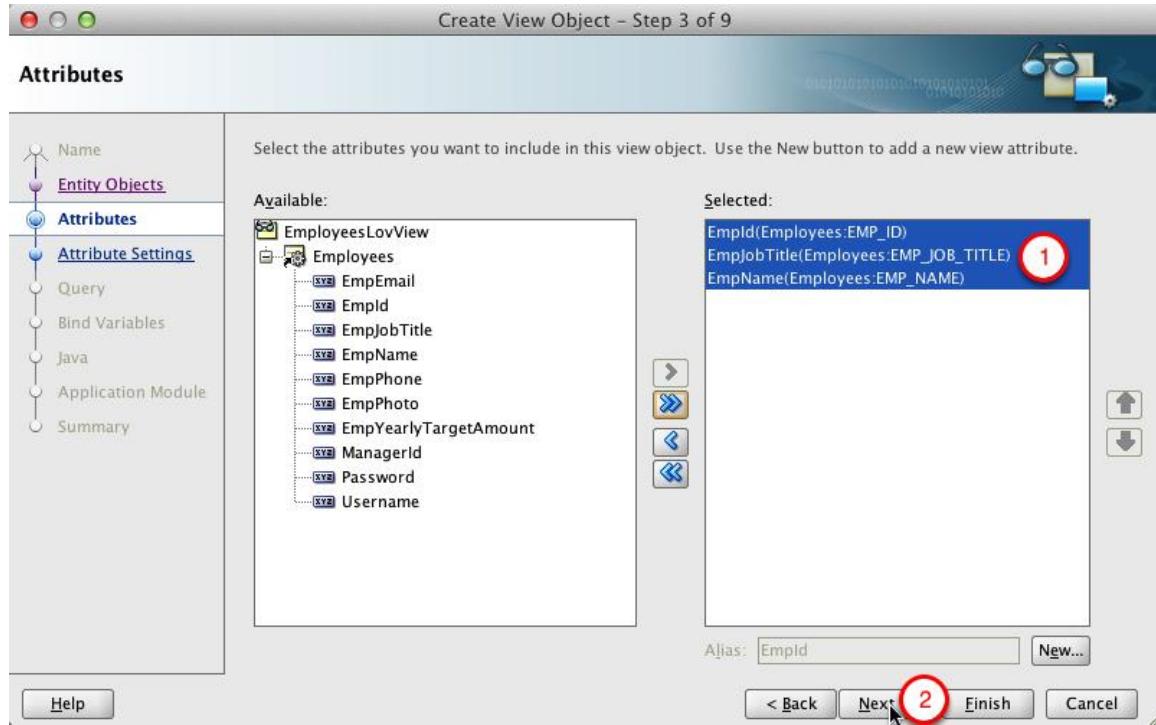
1. Set Name to EmployeesLovView
2. Click Next

Create View Object - Step 2 of 9



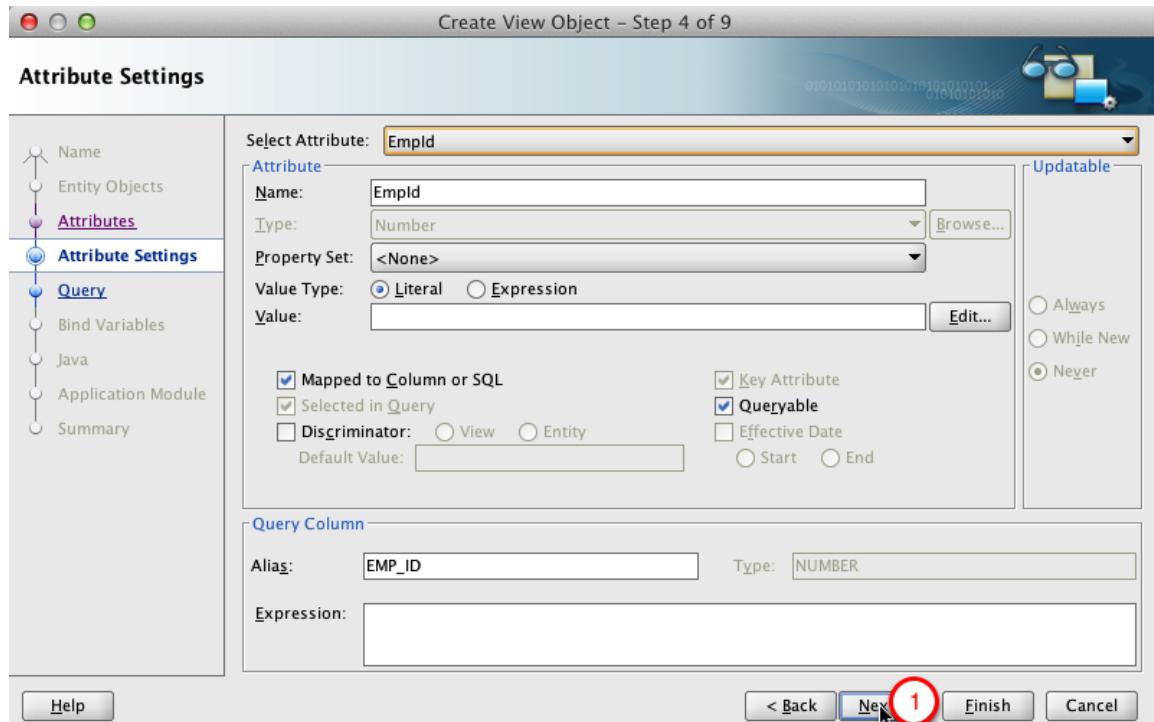
1. Select and move Employees entity to the right using Shuttle
2. Uncheck Updatable checkbox
3. Click Next

Create View Object - Step 3 of 9



1. Select EmpId, EmpJobTitle and EmpName attributes and slide them to the right using shuttle
2. Click Next

Create View Object - Step 4 of 9



1. Click Next

Create View Object - Step 5 of 9

Query

By default, the SELECT list and FROM clause are automatically maintained. To override this mechanism, select Expert Mode.

Generated Statement

```
SELECT Employees.EMP_ID,
       Employees.EMP_JOB_TITLE,
       Employees.EMP_NAME
  FROM EMPLOYEES Employees
 ORDER BY EMP_NAME
```

Query Clauses

Where:

Order By: **EMP_NAME** 1

Binding Style: Oracle Named

SQL Mode: Normal

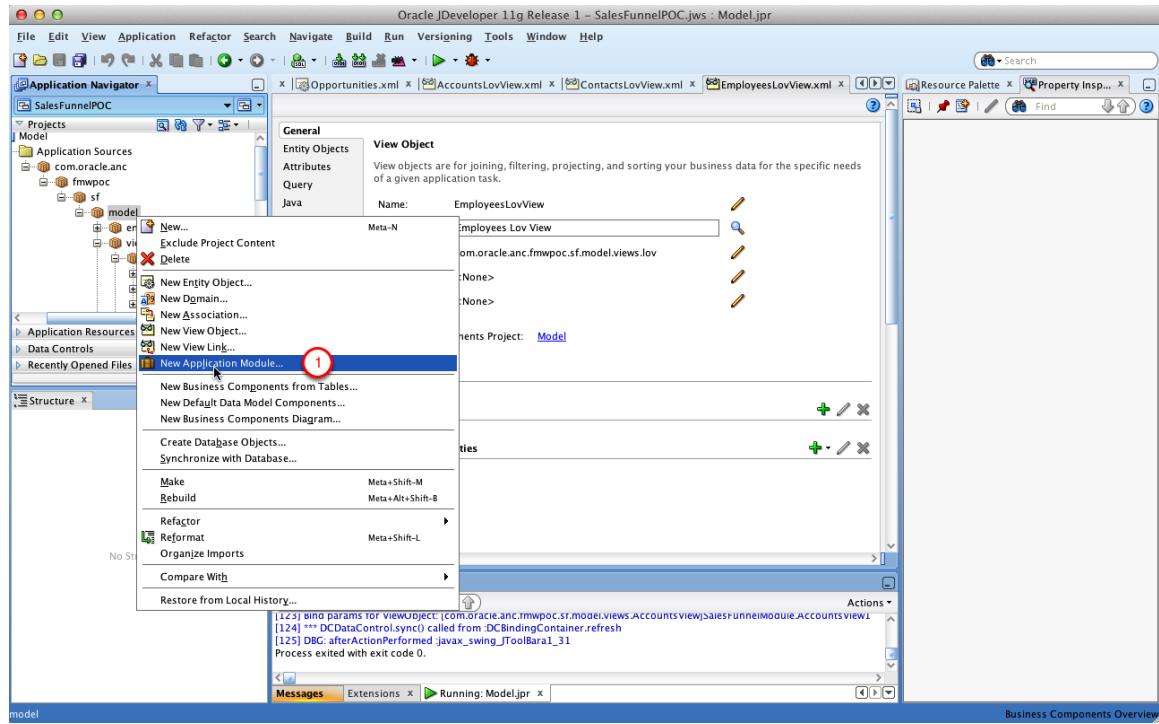
Finish 2

```
SELECT Employees.EMP_ID,
       Employees.EMP_JOB_TITLE,
       Employees.EMP_NAME
  FROM EMPLOYEES Employees
 ORDER BY EMP_NAME
```

1. Set Order By to EMP_NAME
2. Click Finish

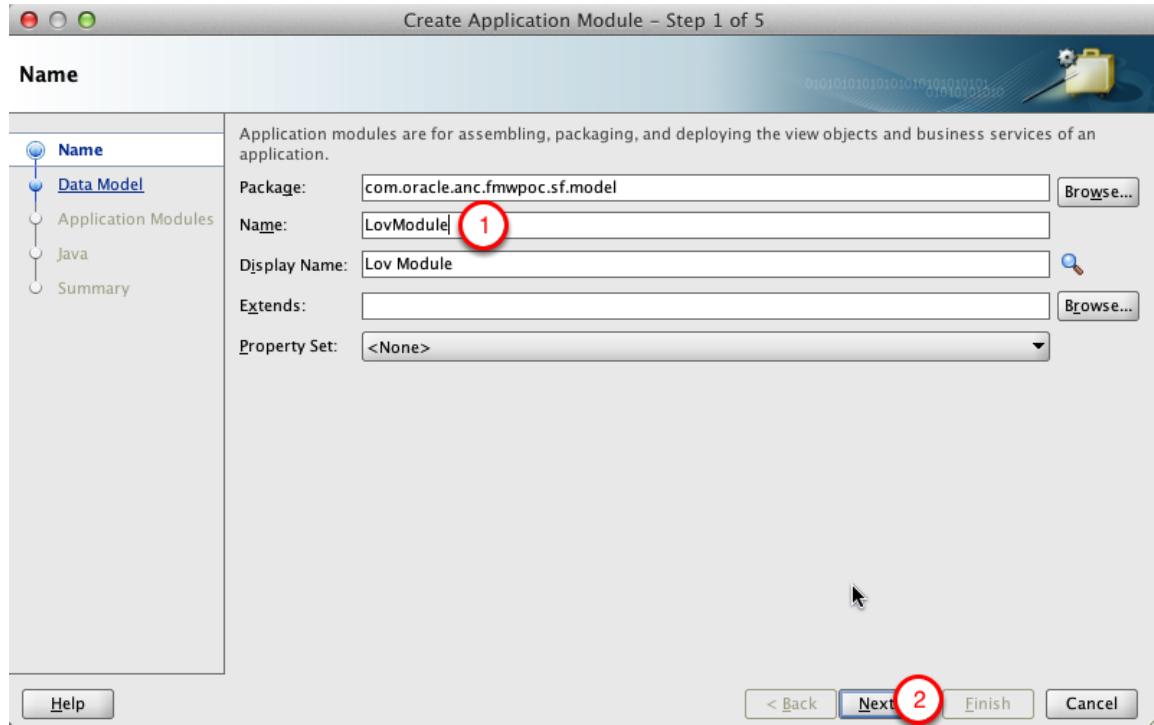
10. Shared LOV AM

Create new application module



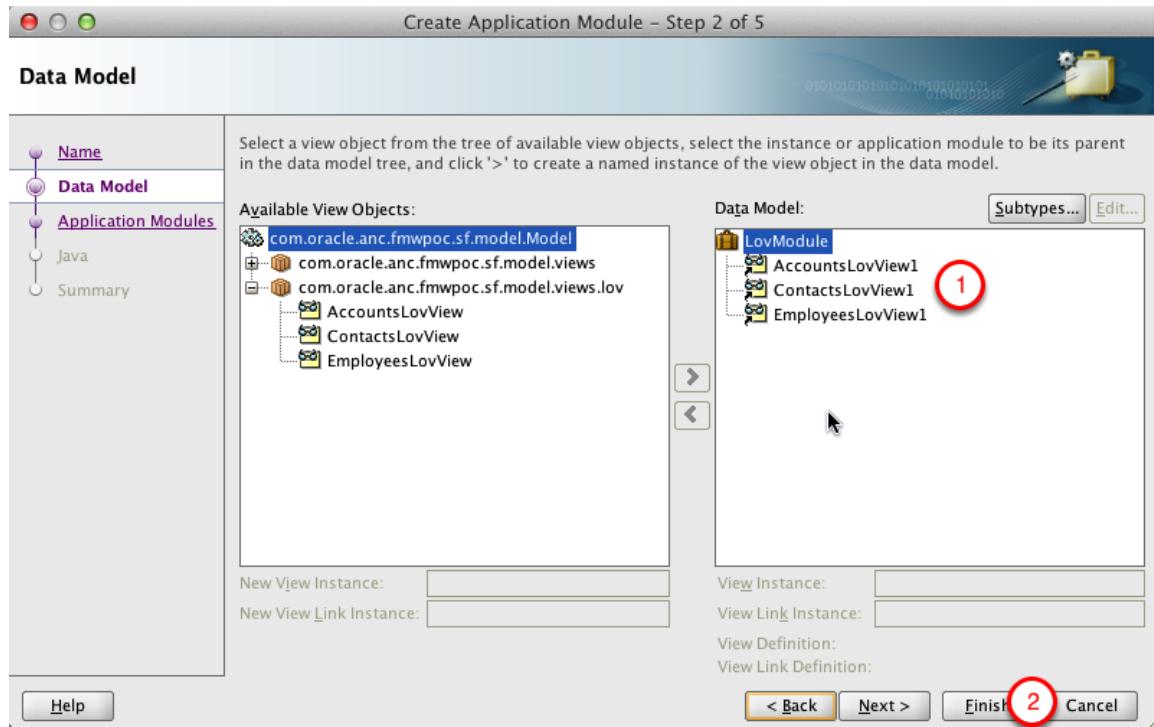
1. Right-click model package and select New Application Module in the popup menu

Create Application Module - Step 1 of 5



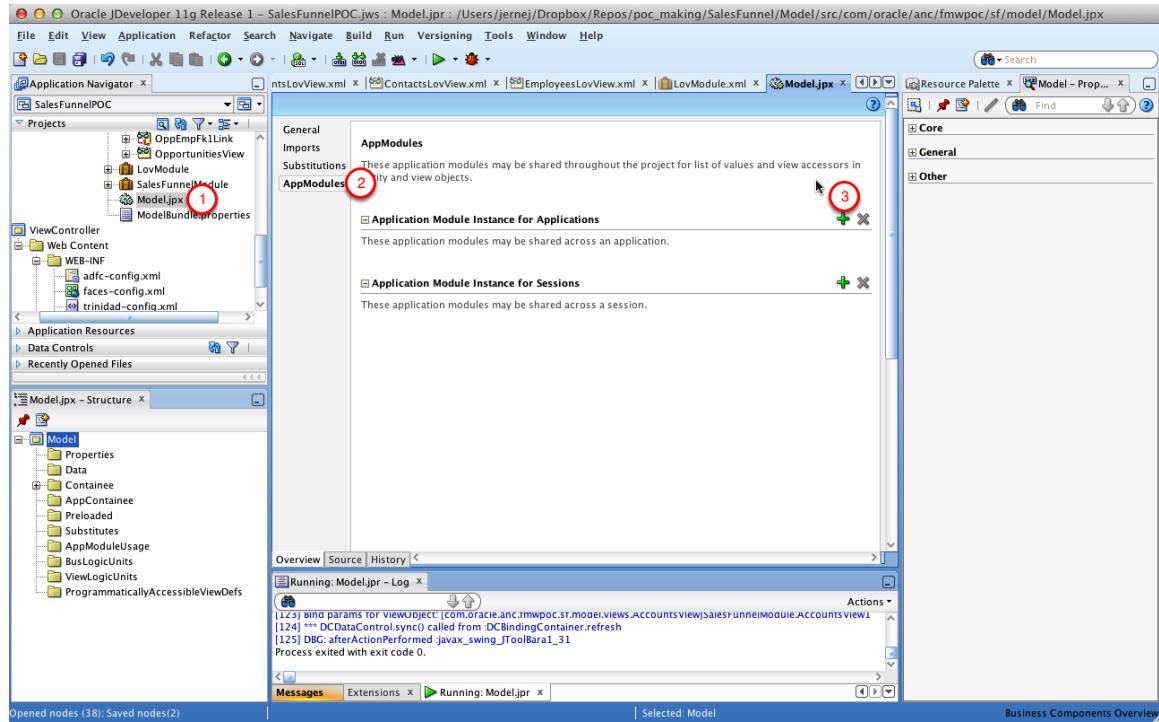
1. Set Name to LovModule
2. Click Next

Create Application Module - Step 2 of 5



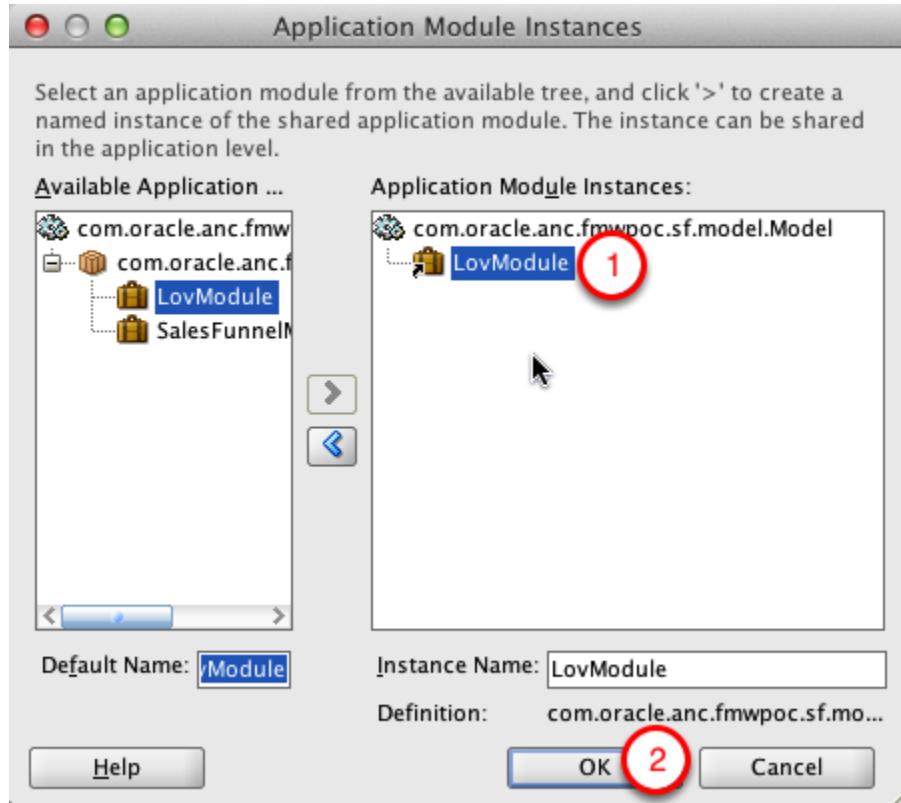
1. In the model.view.lov select AccountLovView, ContactsLovView and EmployeesLovView and slide them to the right
2. Click Finish

Configure shared application module instance



1. Open Model.jpx
2. Open AppModule tab
3. Click + next to Application Module Instance for Sessions

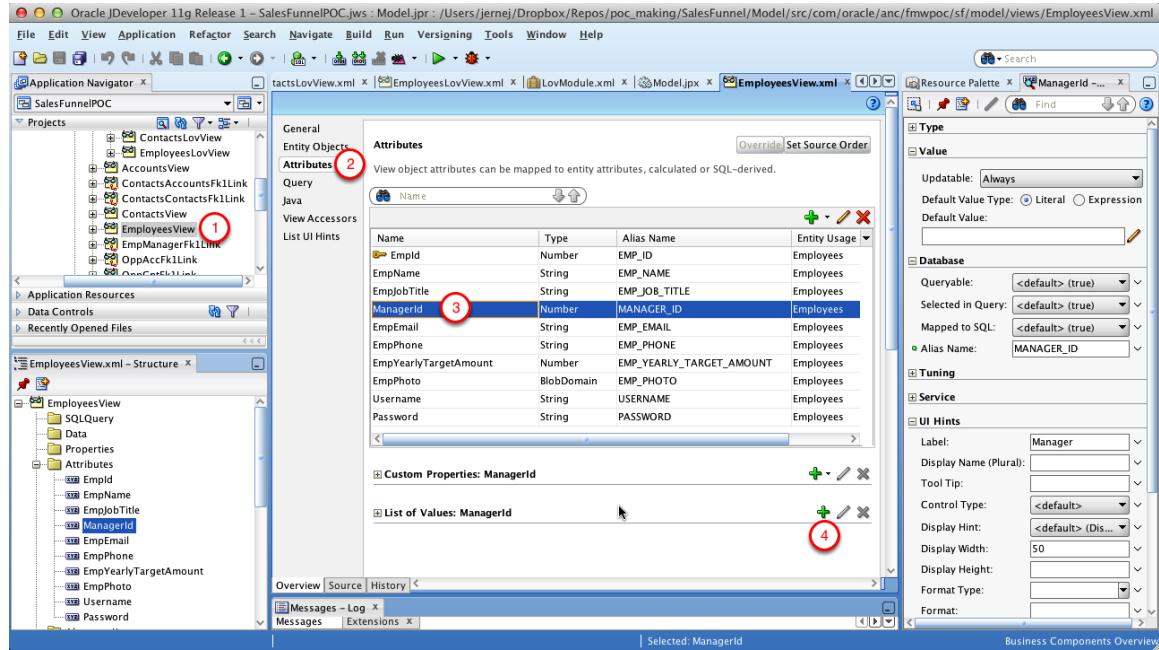
Application Module Instances



1. Select and slide LovModule to the right
2. Click OK

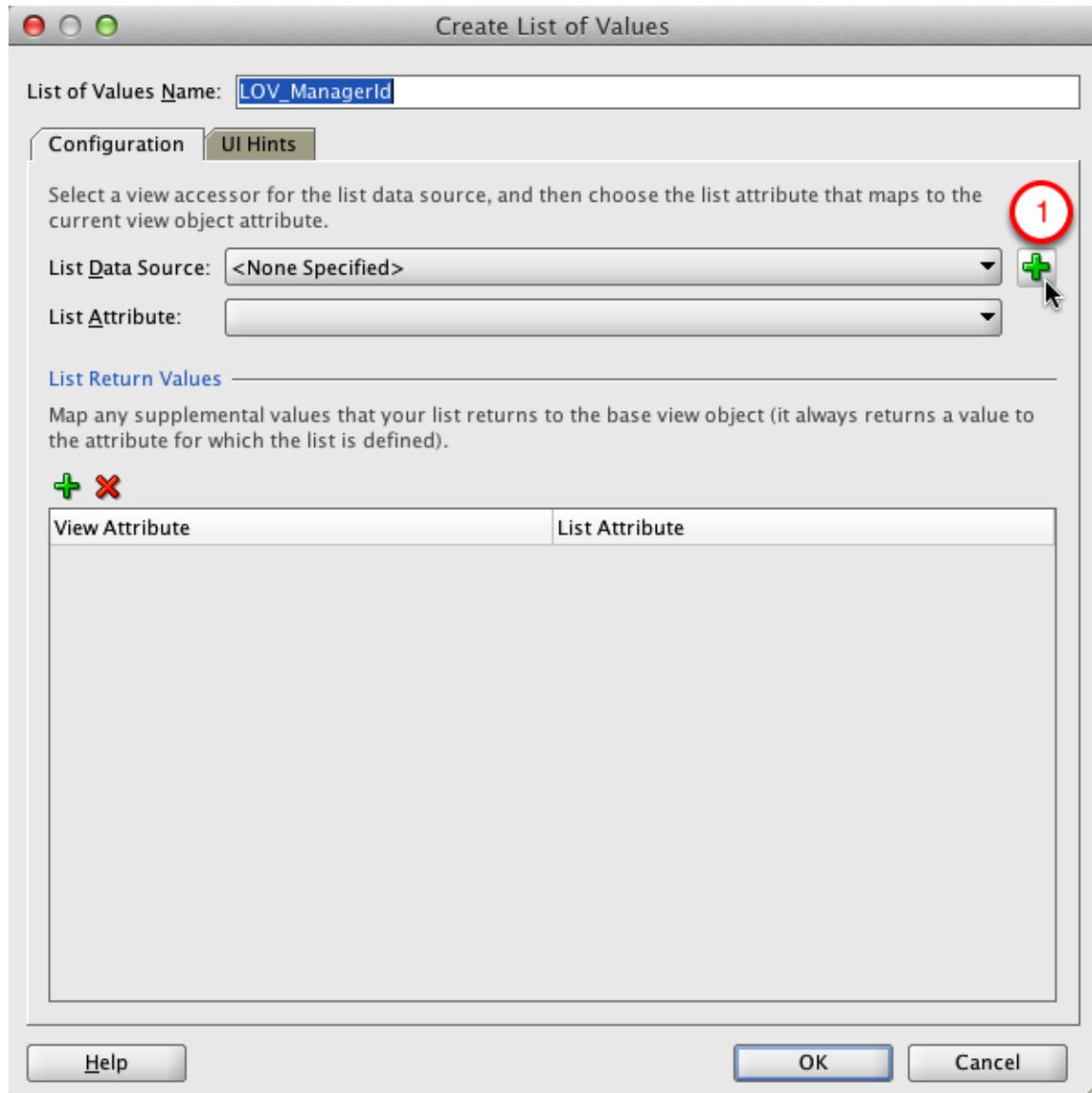
11. Employees ManagerId LOV

Select ManagerId attribute



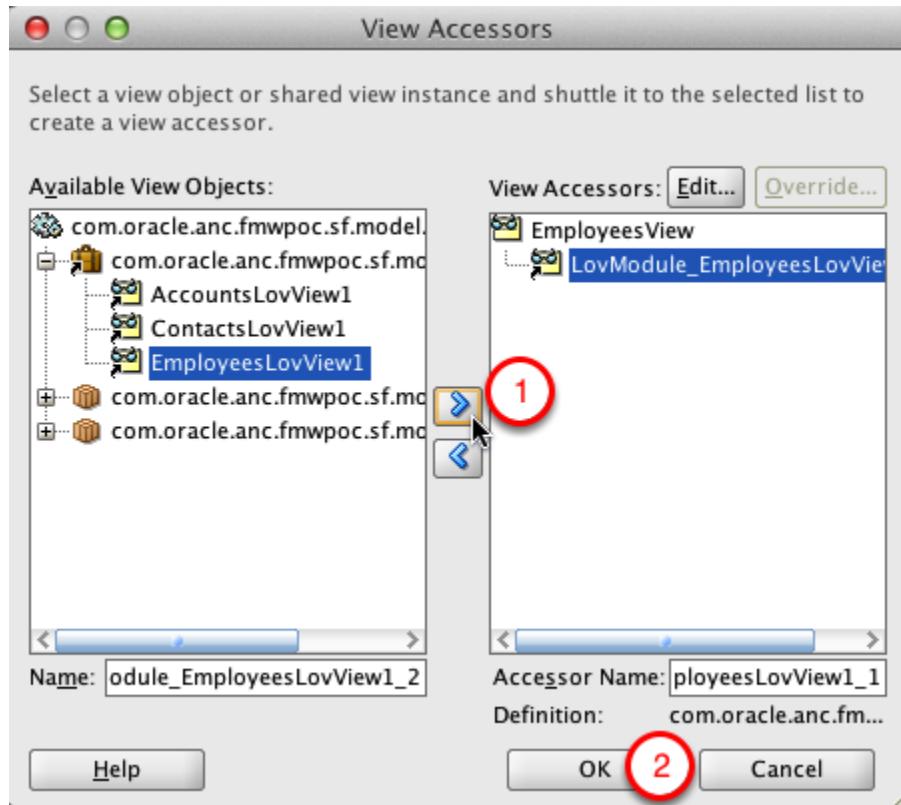
1. In Application Navigator, select and open EmployeesView (note: make sure you don't open Employees Entity instead)
2. Open Attributes tab
3. Select ManagerId attribute
4. Click the + next to List of Values

Create List of Values



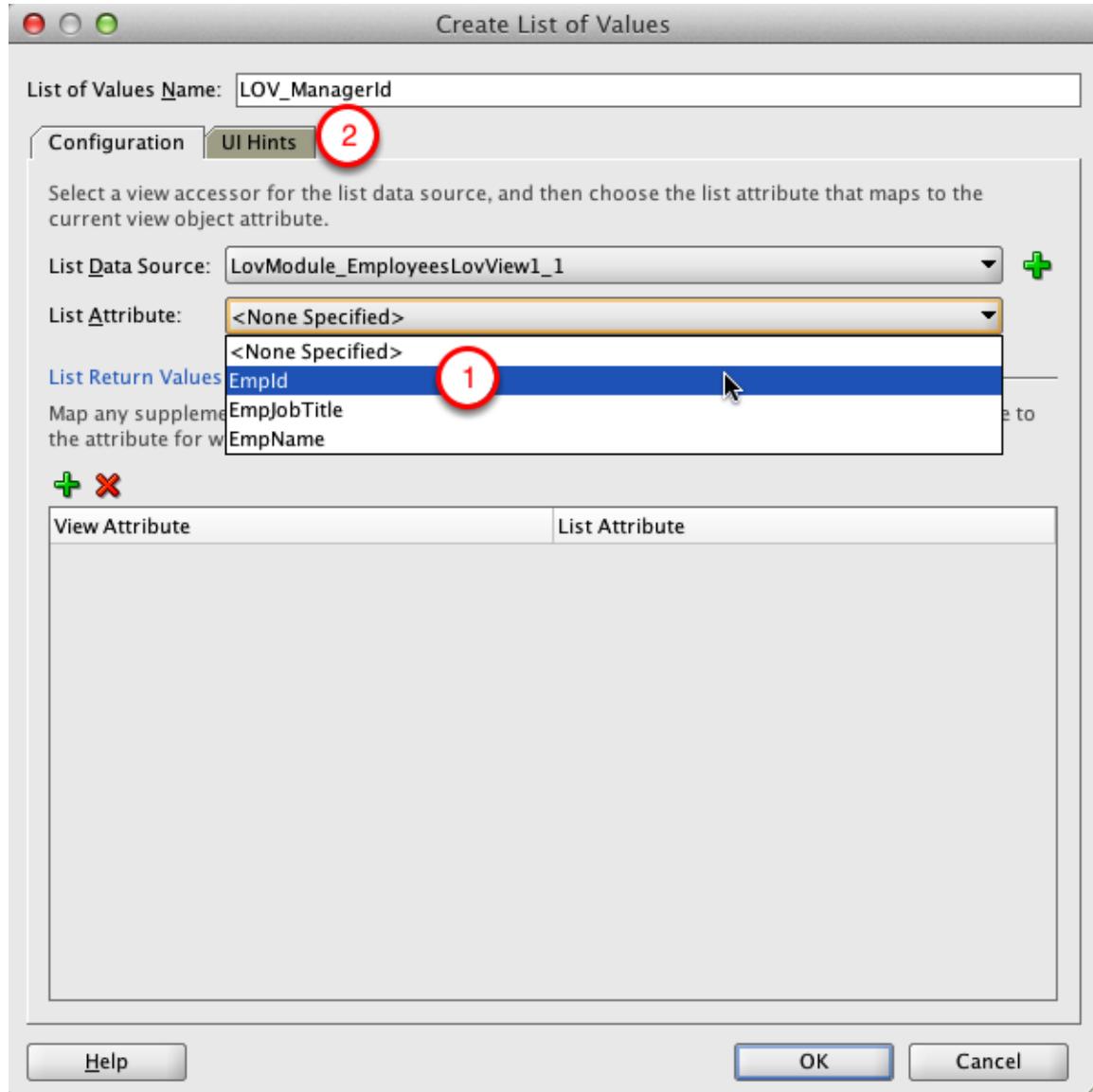
1. Click + next to List Data Source

View Accessors



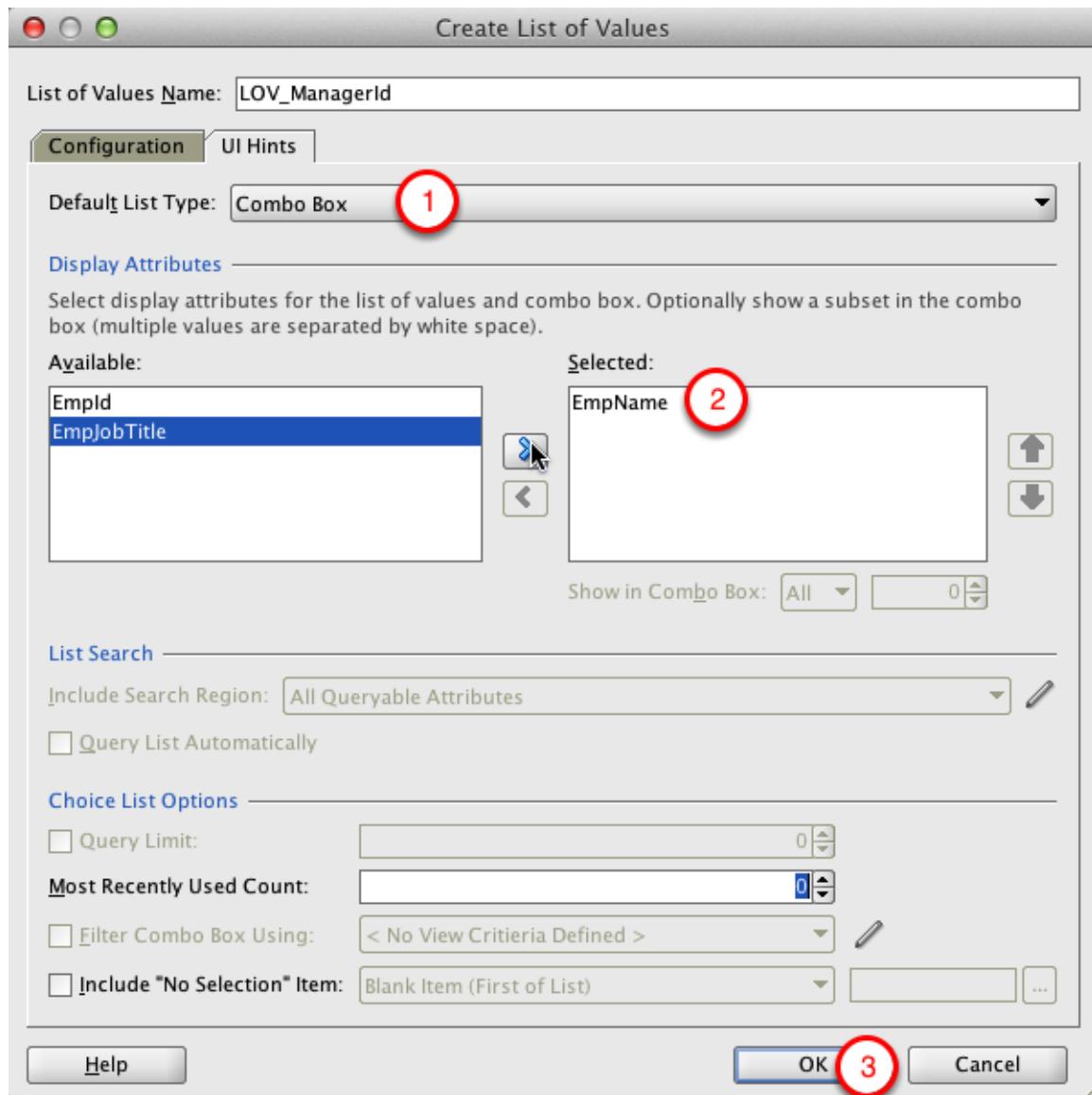
1. Select EmployeesLovView1 within the shared application module and slide it to the right.
Make sure you don't select the view from the model.views.lov package instead!
2. Click OK

Create List of Values



1. Set EmplId List Attribute
2. Open UI Hints Tab

Create List of Values

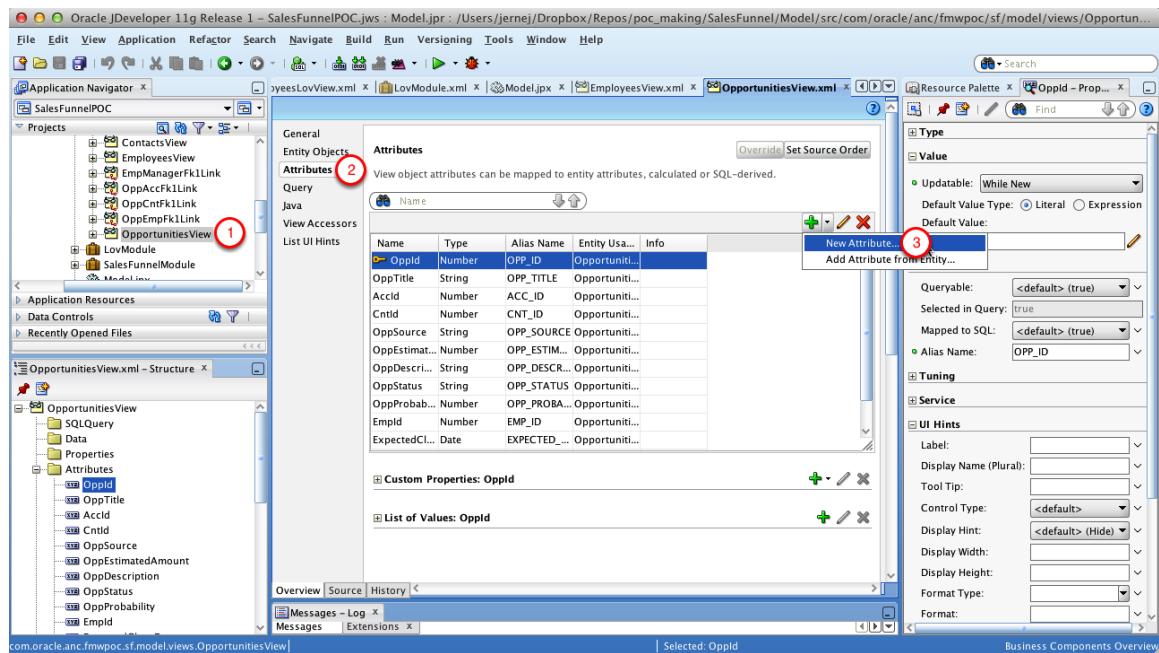


1. Set List Type to Combo Box
2. Select EmpName attribute and slide it to the right
3. Click OK

12. Opportunities Account LOV

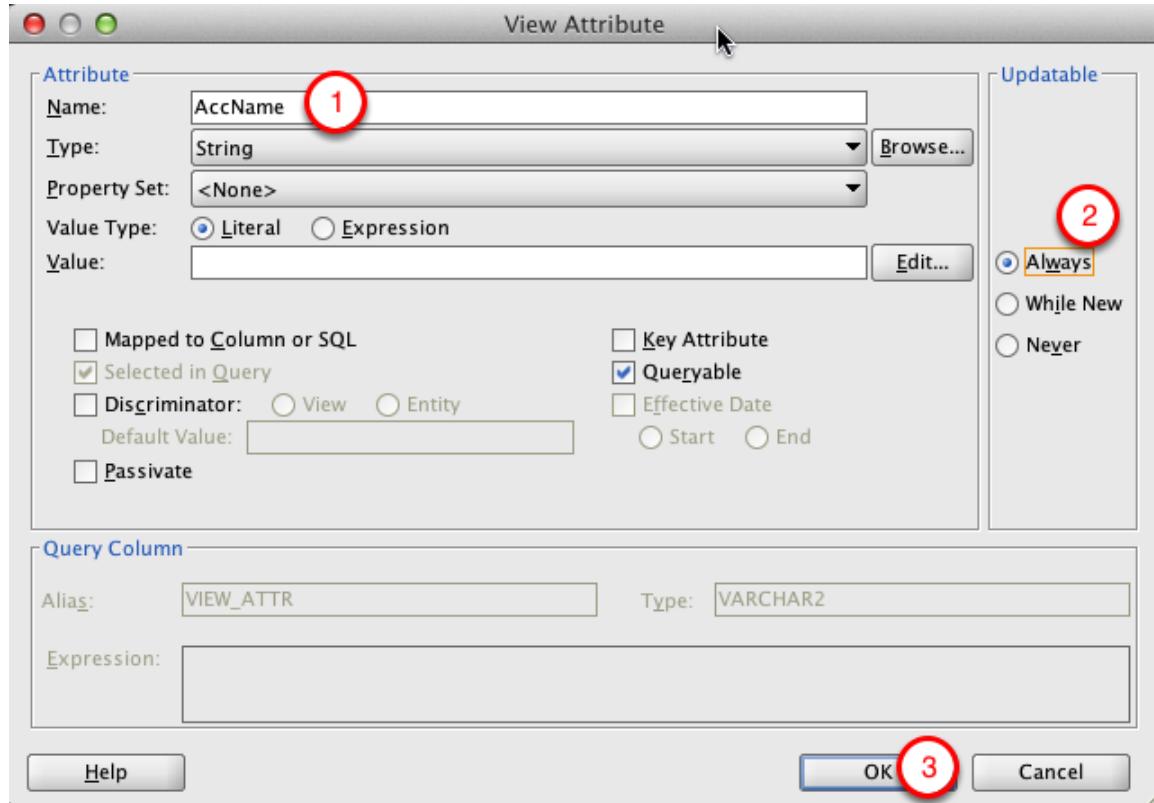
In this lesson, we will add Combo Box with List of Values lookup to the OpportunitiesView. ComboBox with List of Values has some advanced features (like search and auto complete), which don't exist for the normal Combo Box. But unlike combo box using it requires a bit different approach than simply binding it to an attribute.

Create new attribute



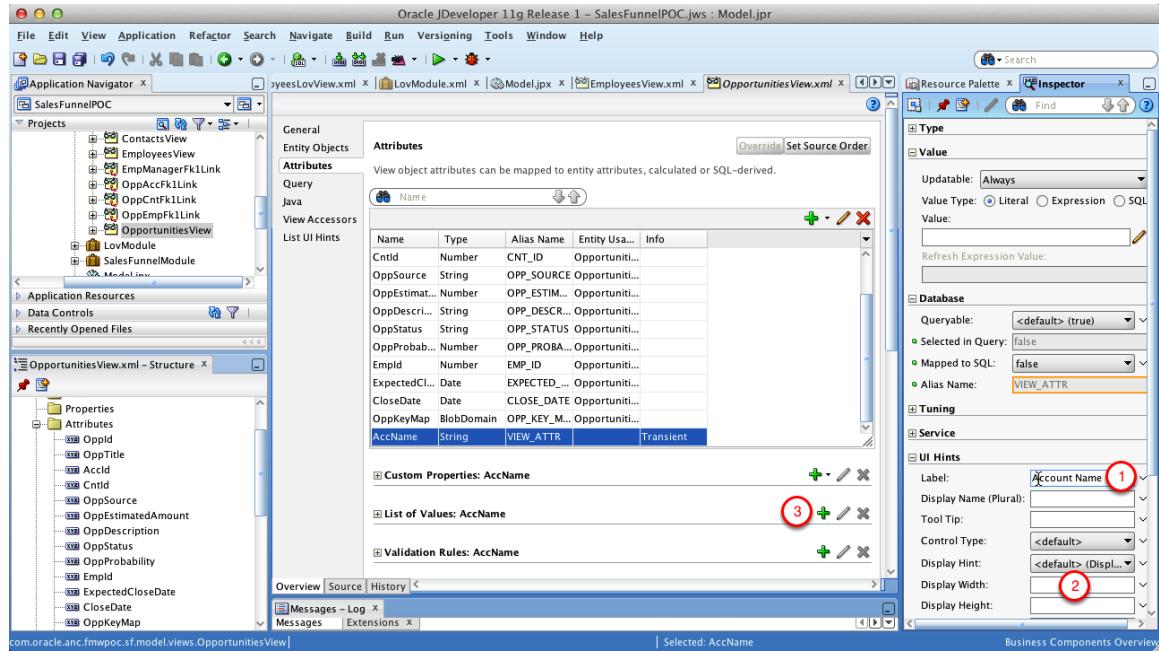
1. Locate and open OpportunitiesView
2. Open Attributes tab
3. Click the + above the attributes and select New Attribute in the popup menu

View Attribute



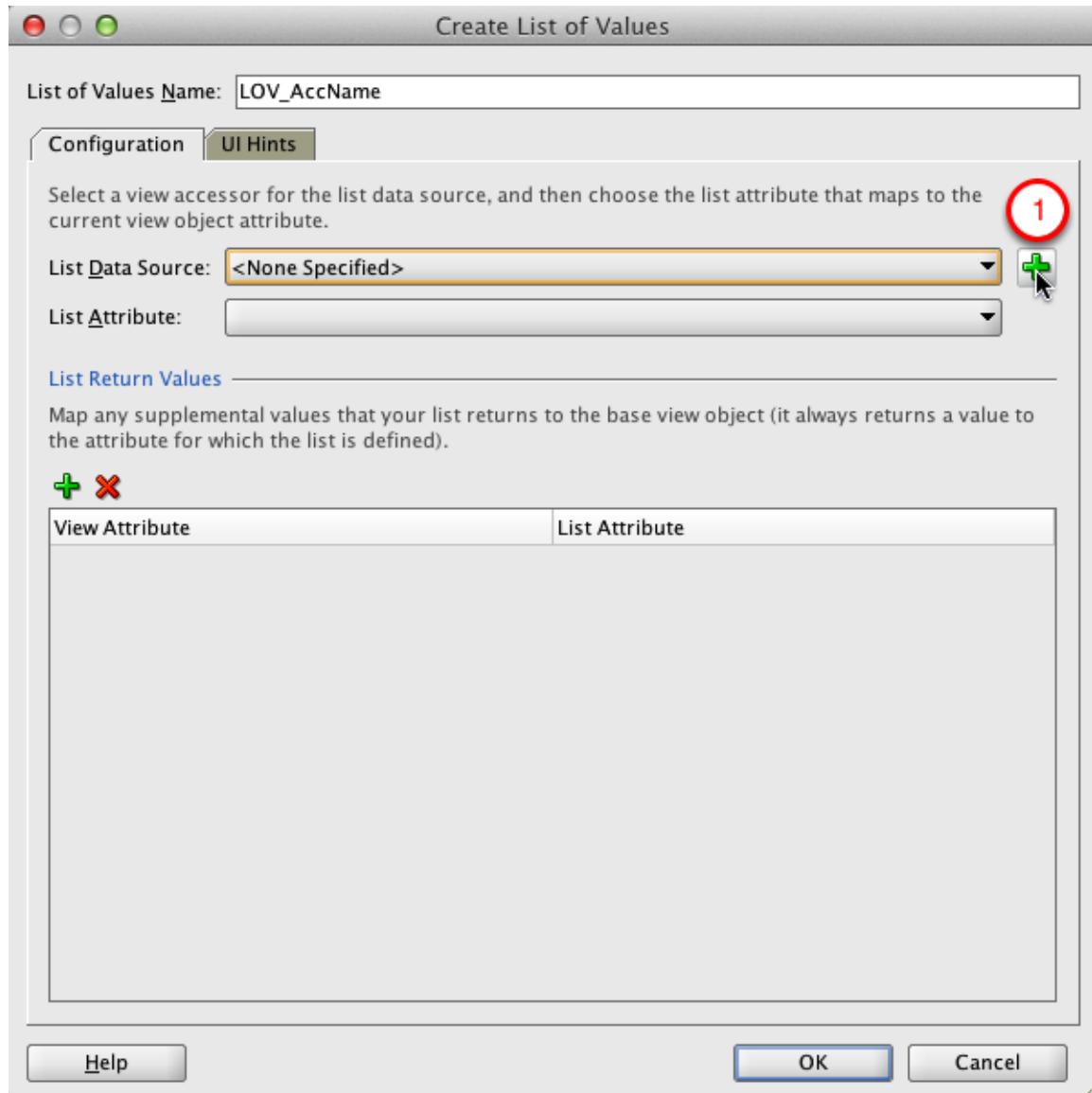
1. Set name to AccName
2. Set Updatable to Always
3. Click OK

Create List of Values



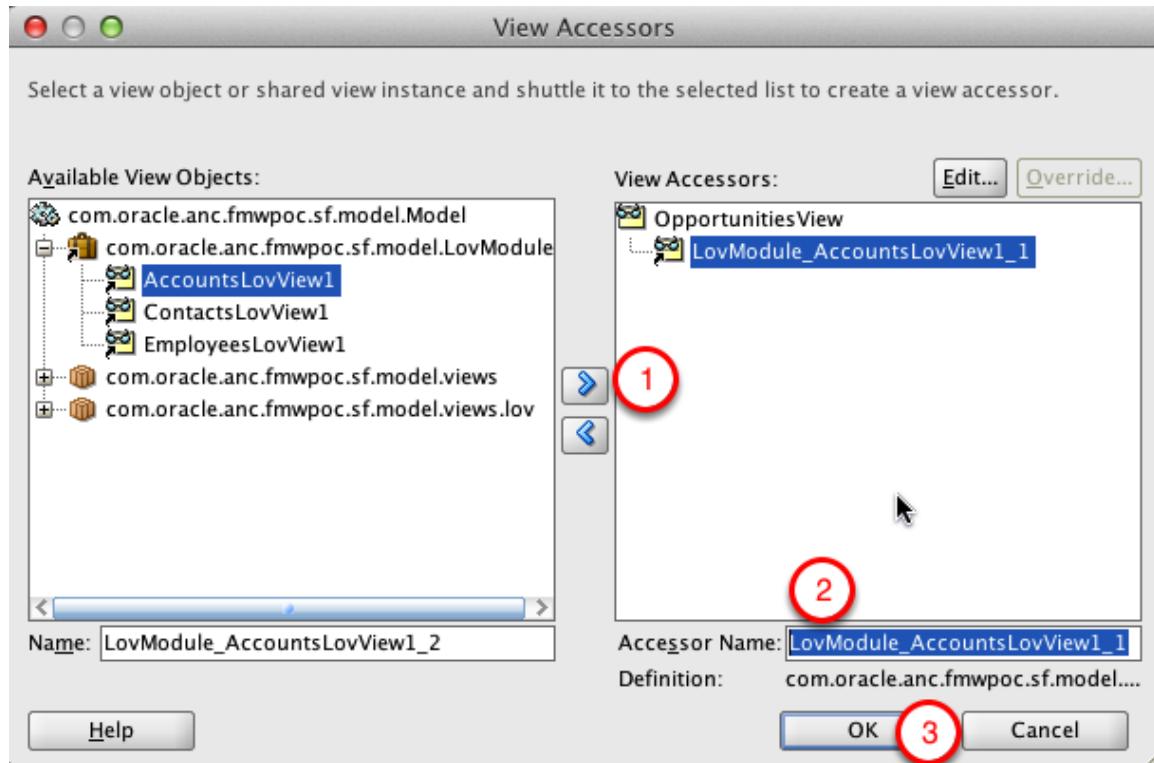
1. Set AccName's Label to Account Name
2. Set Display Width to 50
3. Click + next to List of Values

Create List of Values



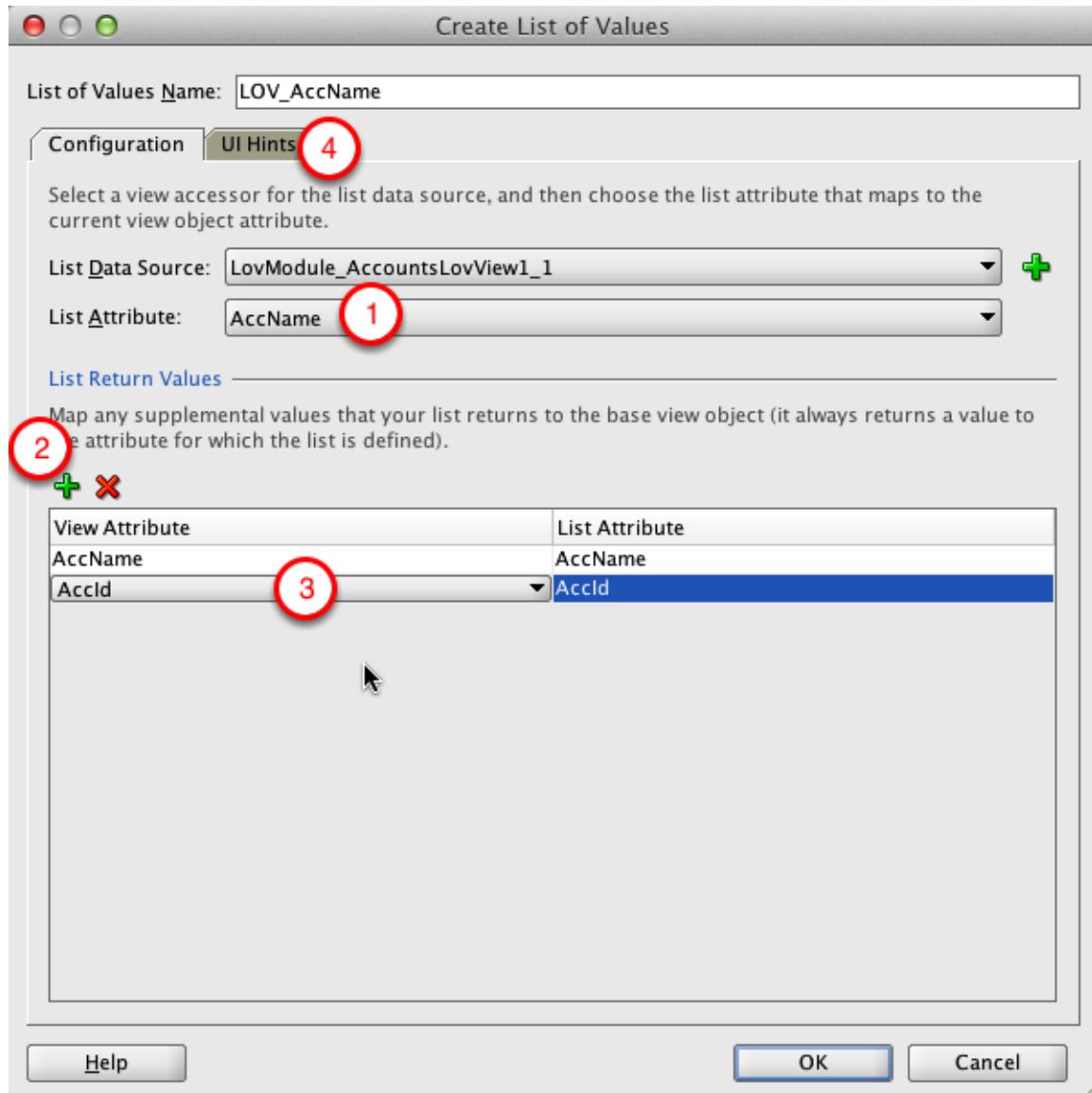
1. Click +

View Accessors



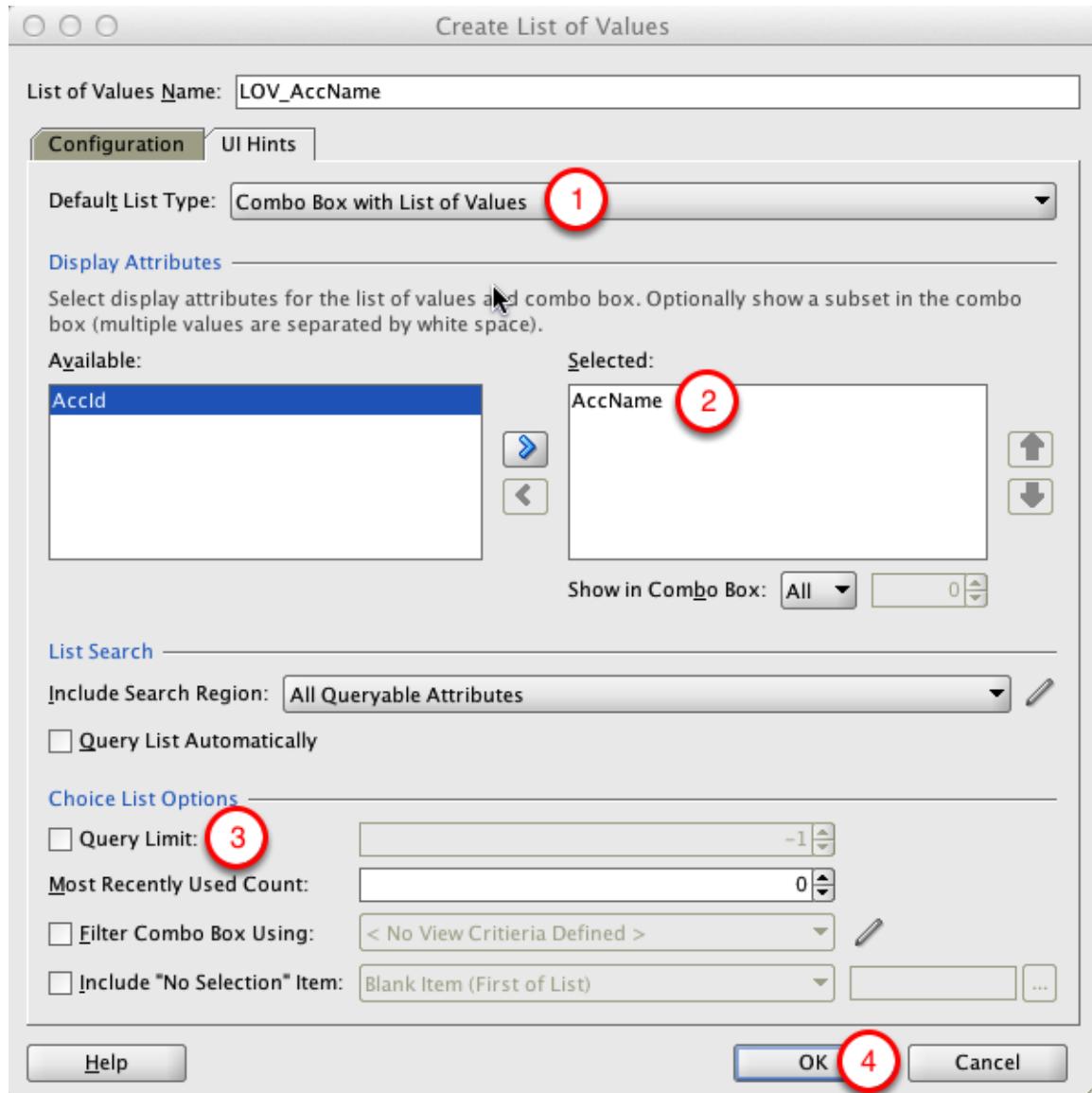
1. Select AccountsLovView1 from the shared LovModule and slide it to the right
2. Copy Accessor Name to clipboard (optionally store it in a text editor temporary)
3. Click OK

Create List of Values



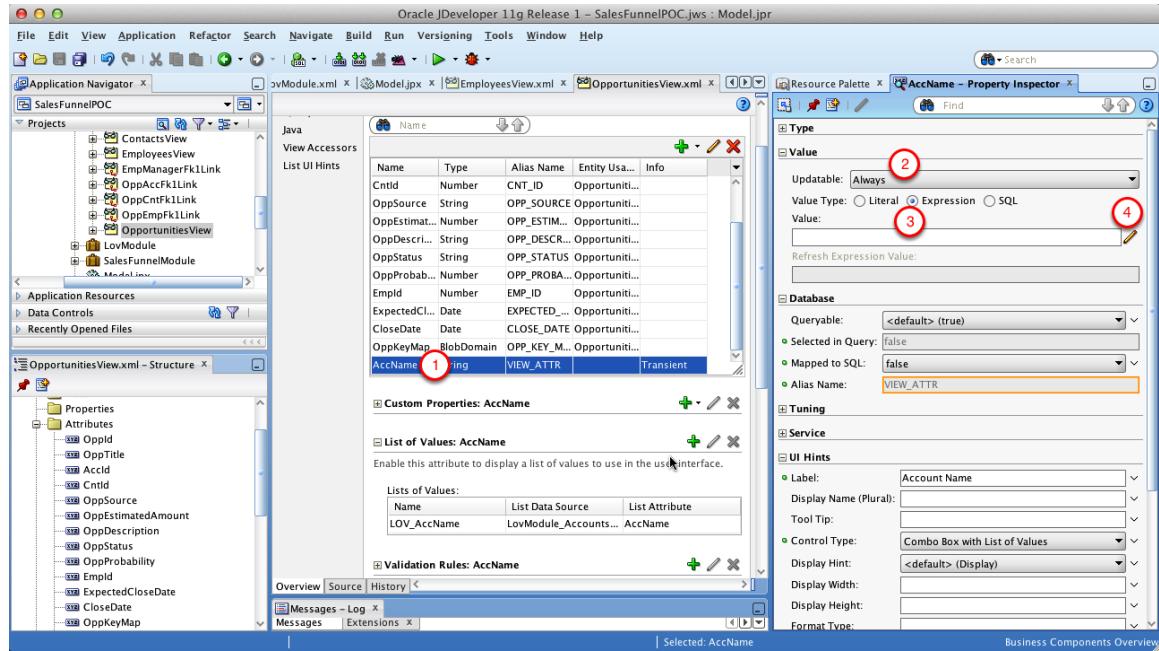
1. Set List Attribute to AccName
2. Click + above attribute list
3. Map AccId to AccId
4. Open UI Hints tab

Create List of Values



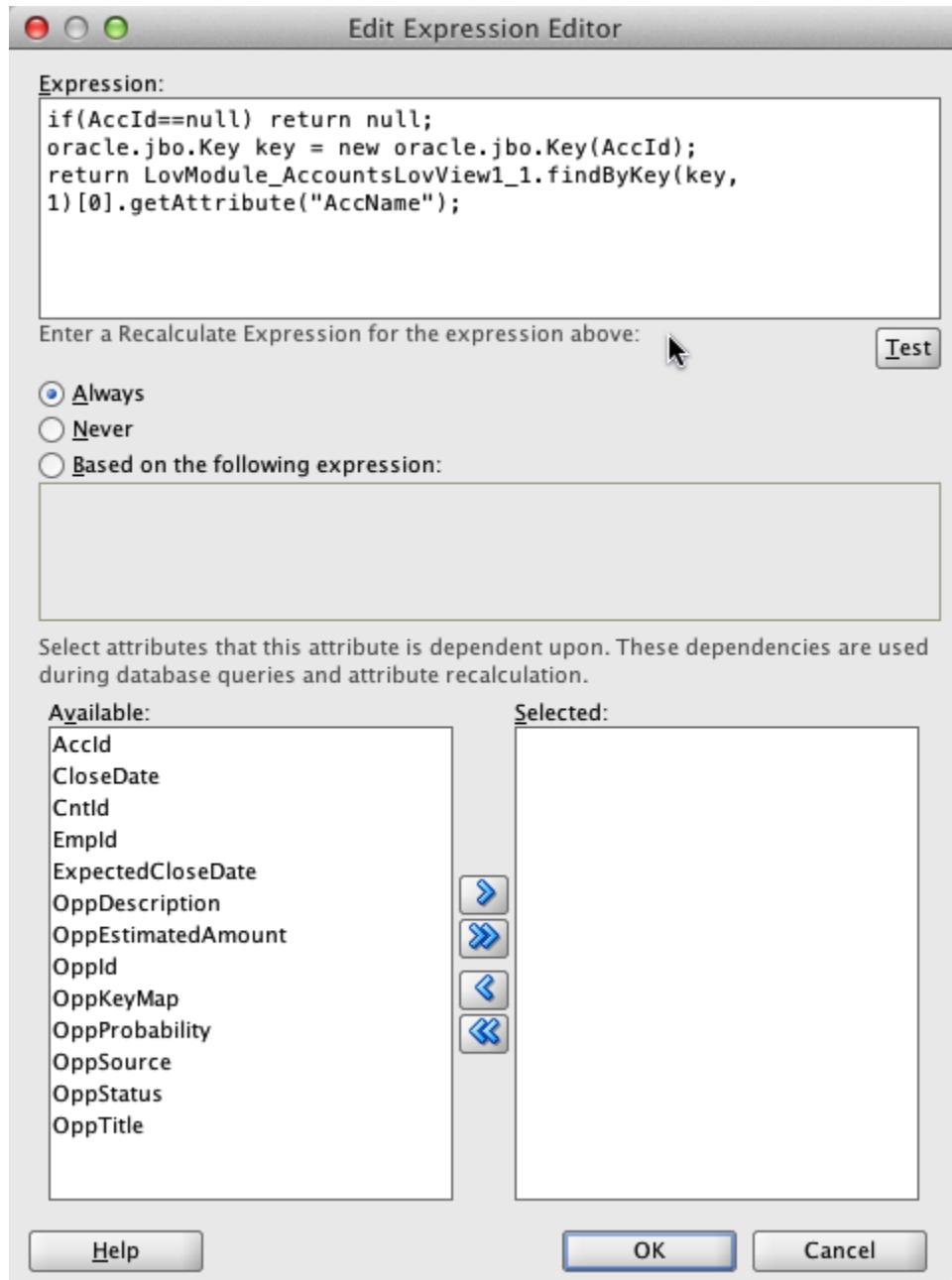
1. Set Default List Type to Combo Box with List of Values
2. Select AccName attribute and slide it to the right
3. Uncheck query limit
4. Click OK

Change AccName properties



1. Select AccName attribute
2. Set Updatable to Always
3. Set Value Type to Expression
4. Click the pencil next to Value property

Edit AccName Value Expression

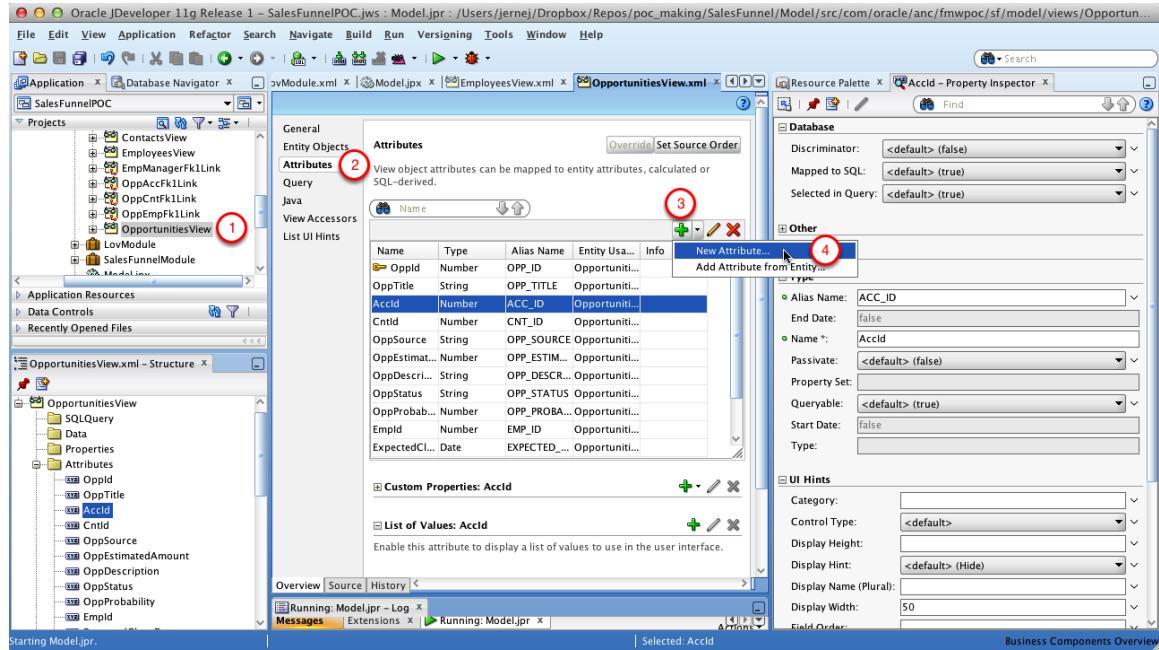


1. Copy paste the bellow expression. Make sure the LovModule_AccountsLovView1_1 matches the text you copied and stored before.

```
if(AccId==null) return null;
oracle.jbo.Key key = new oracle.jbo.Key(AccId);
return LovModule_AccountsLovView1_1.findByKey(key, 1)[0].getAttribute("AccName");
```

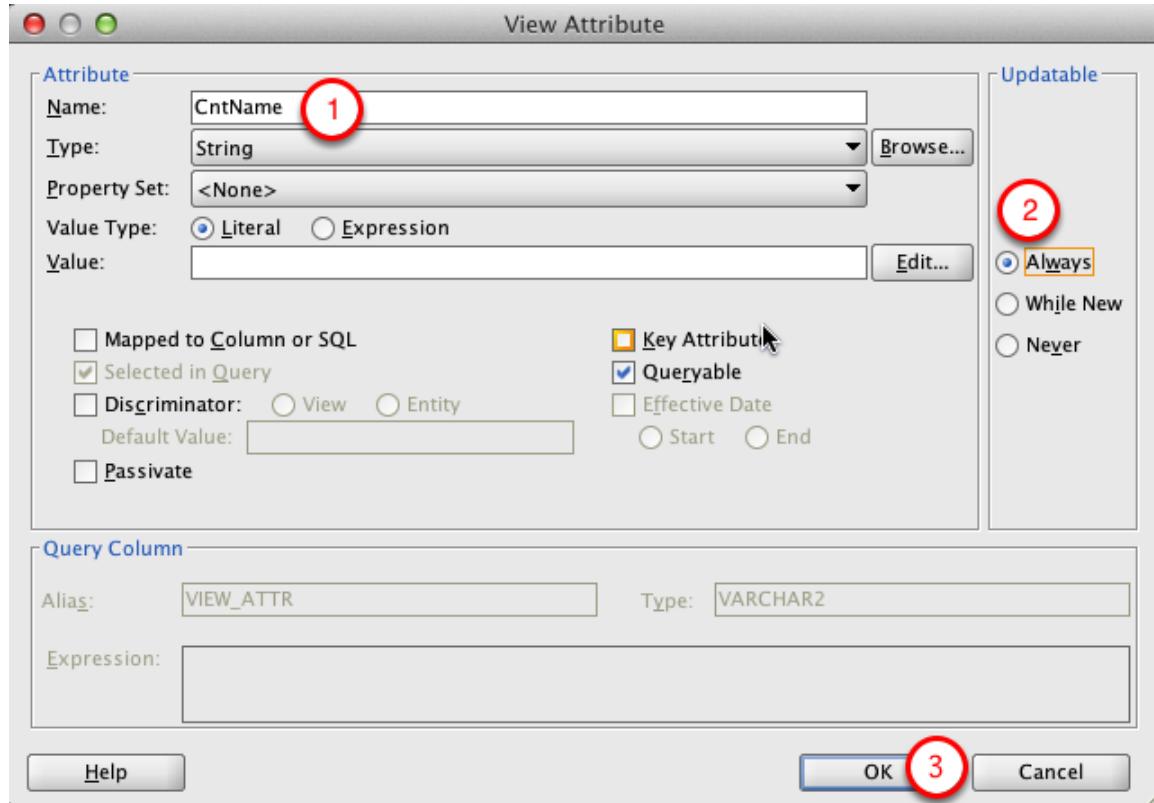
13. Opportunities Contact LOV

Create new attribute



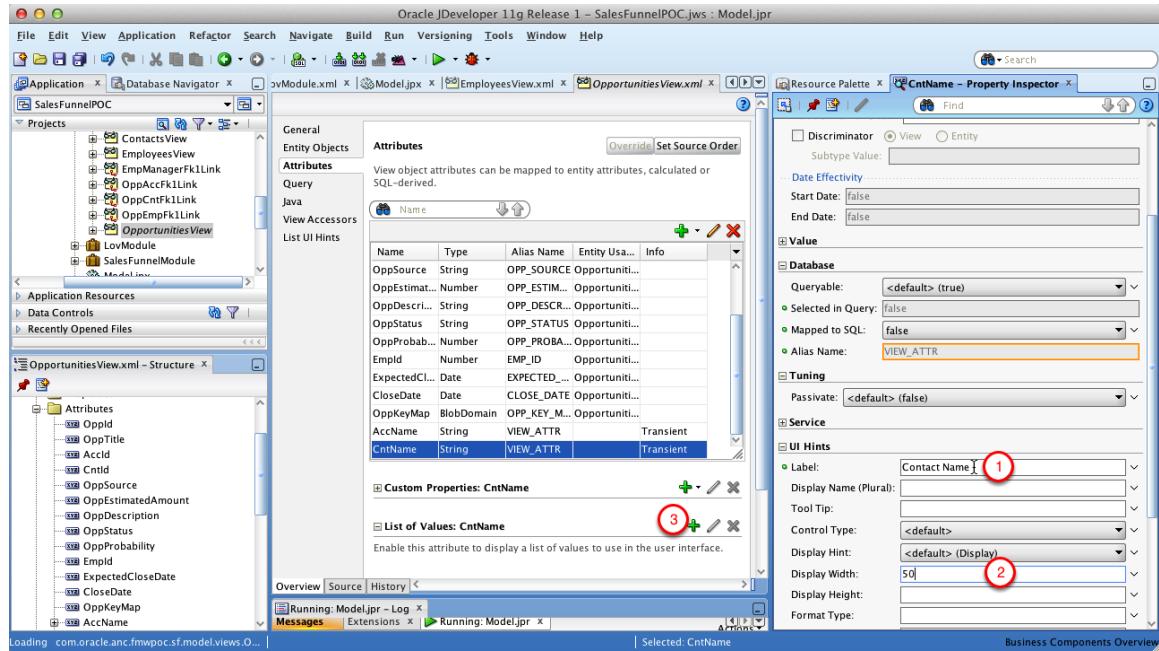
1. Locate and open OpportunitiesView
2. Open Attributes tab
3. Click the + above the attributes and select New Attribute in the popup menu

View Attribute



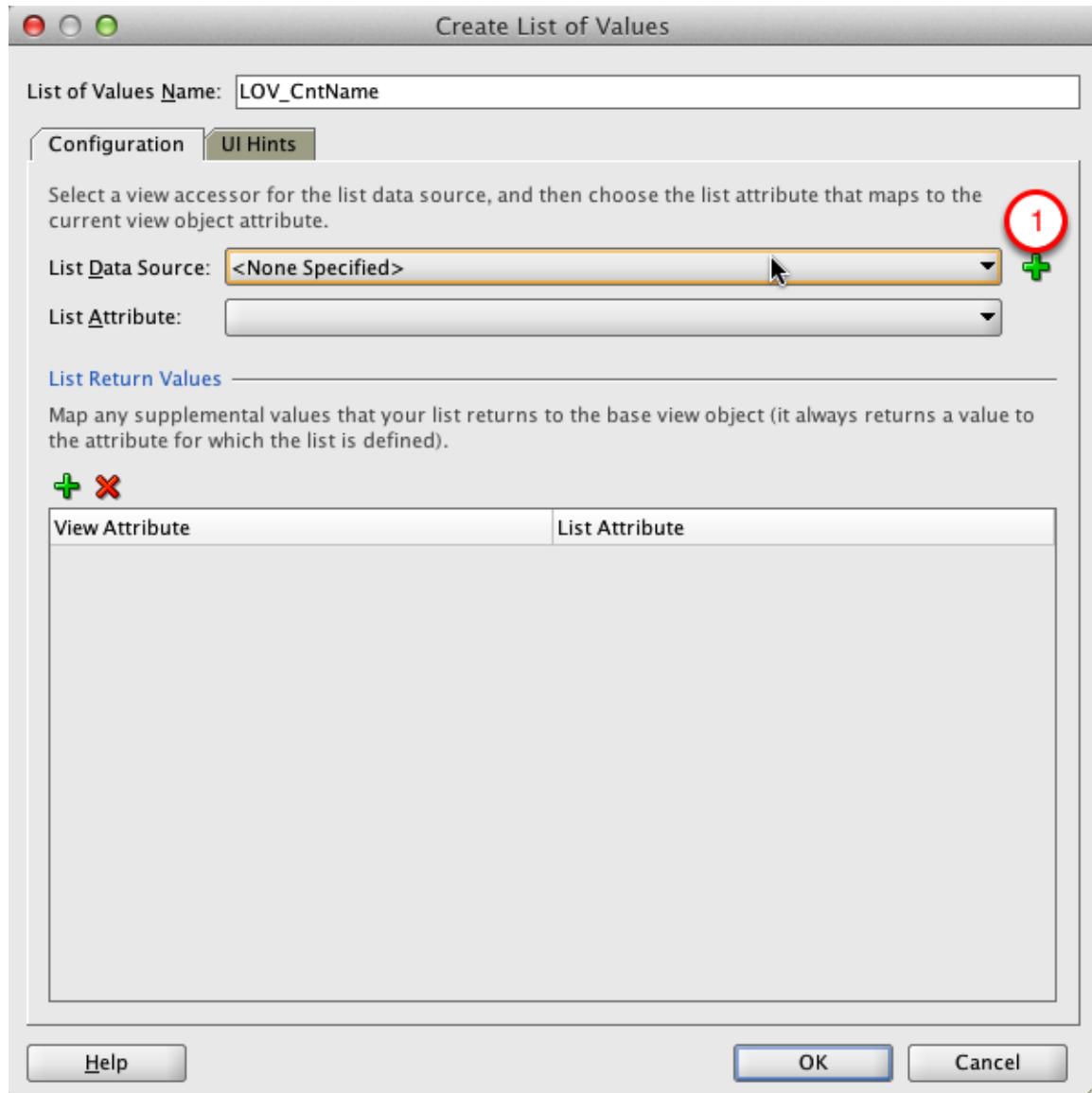
1. Set Name to CntName
2. Set Updatable to Always
3. Click OK

Set CntName properties



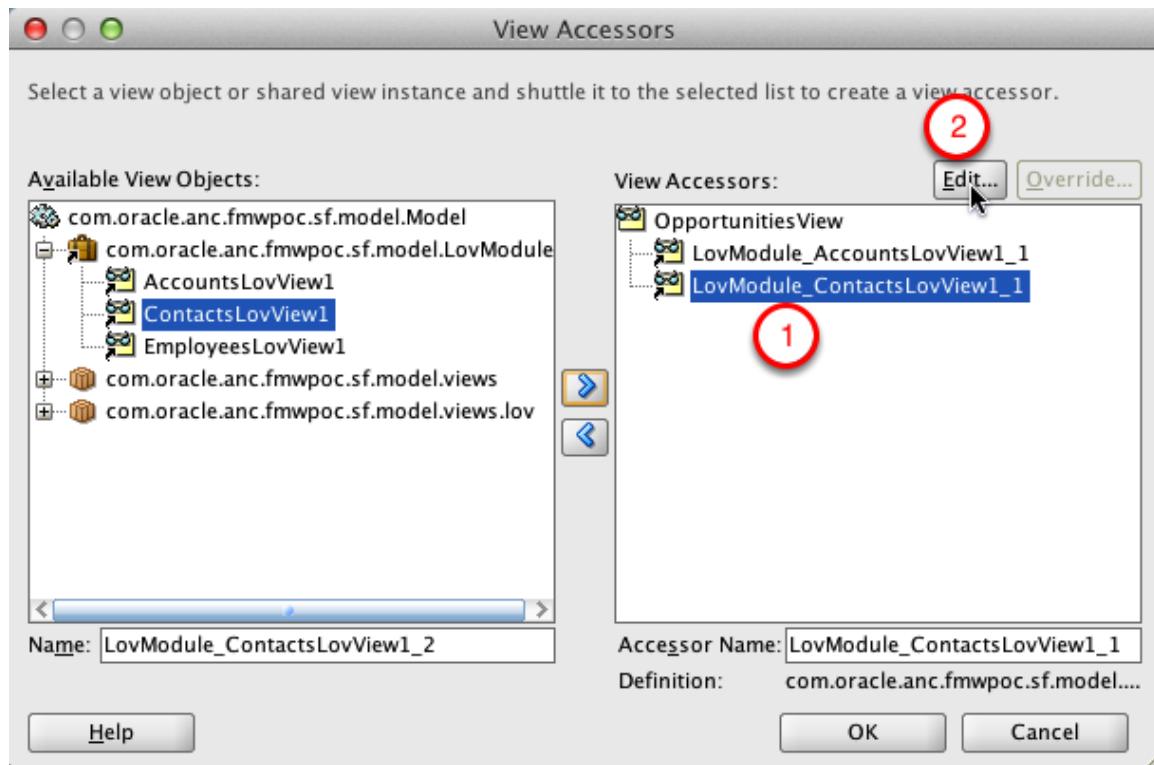
1. Set CntName's Label to "Contact Name"
2. Set Display Width to 50
3. Click + next to List of Values

Create List of Values



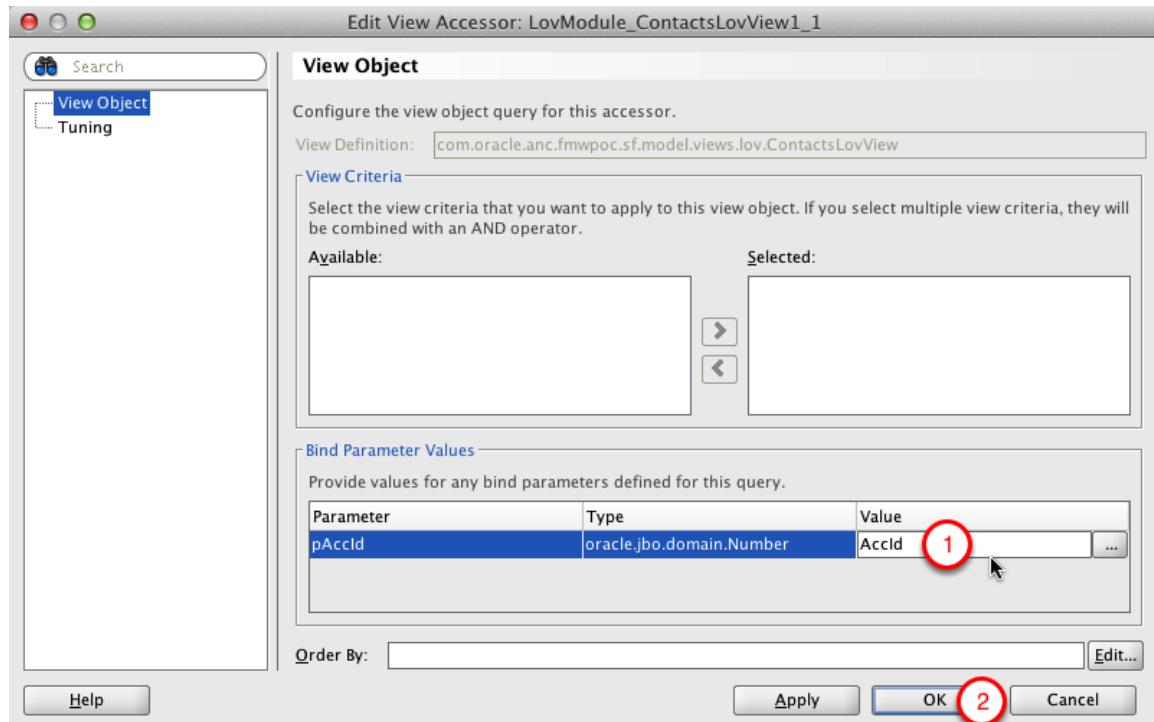
1. Click + next to List Data Source

View Accessors



1. Select ContactsLovView1 within the shared LovModule
2. Click Edit button

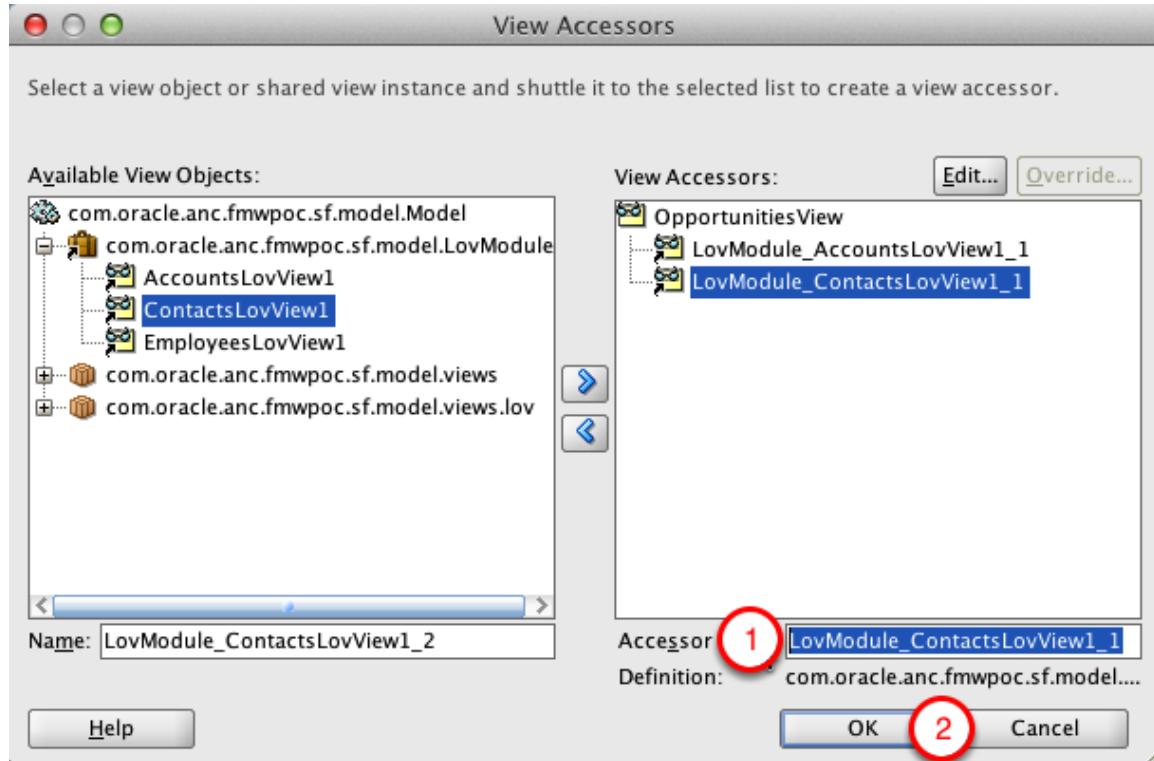
Edit View Accessor: LovModule_ContactsLovView1_1



1. Set pAcld's parameter value to Acld*
2. Click OK

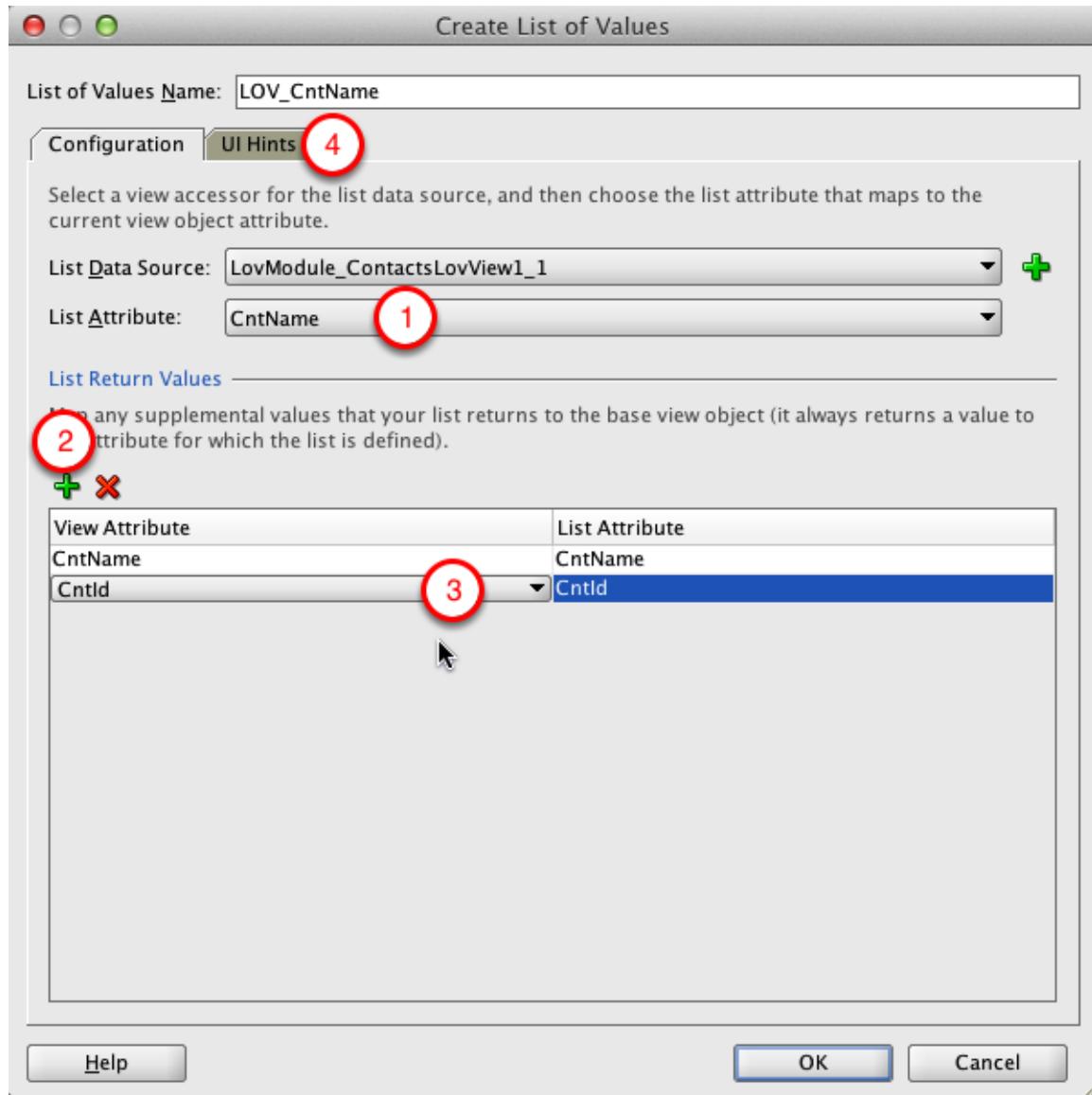
At runtime, this will populate pAcld with the current value of OpportunitiesViewRow Acld attribute. Consequently, the ContactsLovView is filtered by the Account Id selected in the Opportunity Row

View Accessors



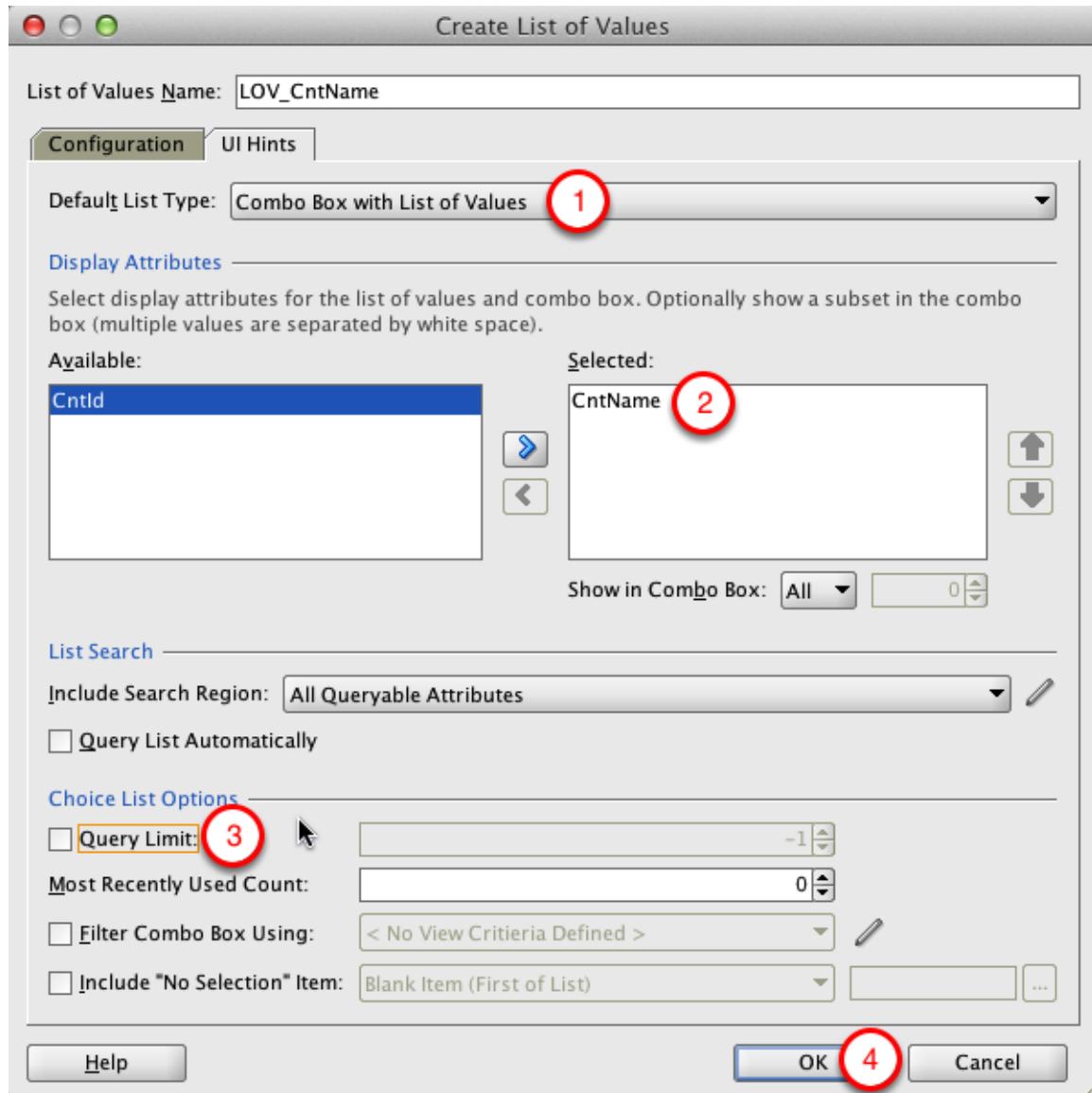
1. Copy Accessor Name and store it somewhere for future us
2. Click OK

Create List of Values



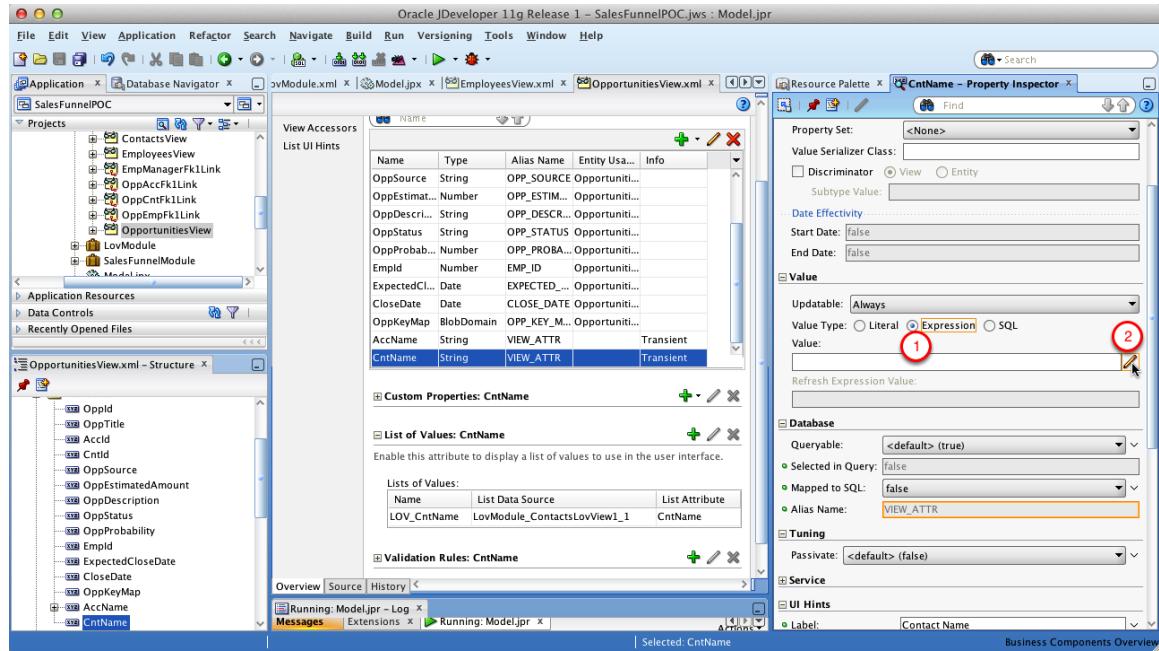
1. Select CntName
2. Click + above attribute list
3. Map CntId to CntId
4. Open UI Hints tab

Create List of Values



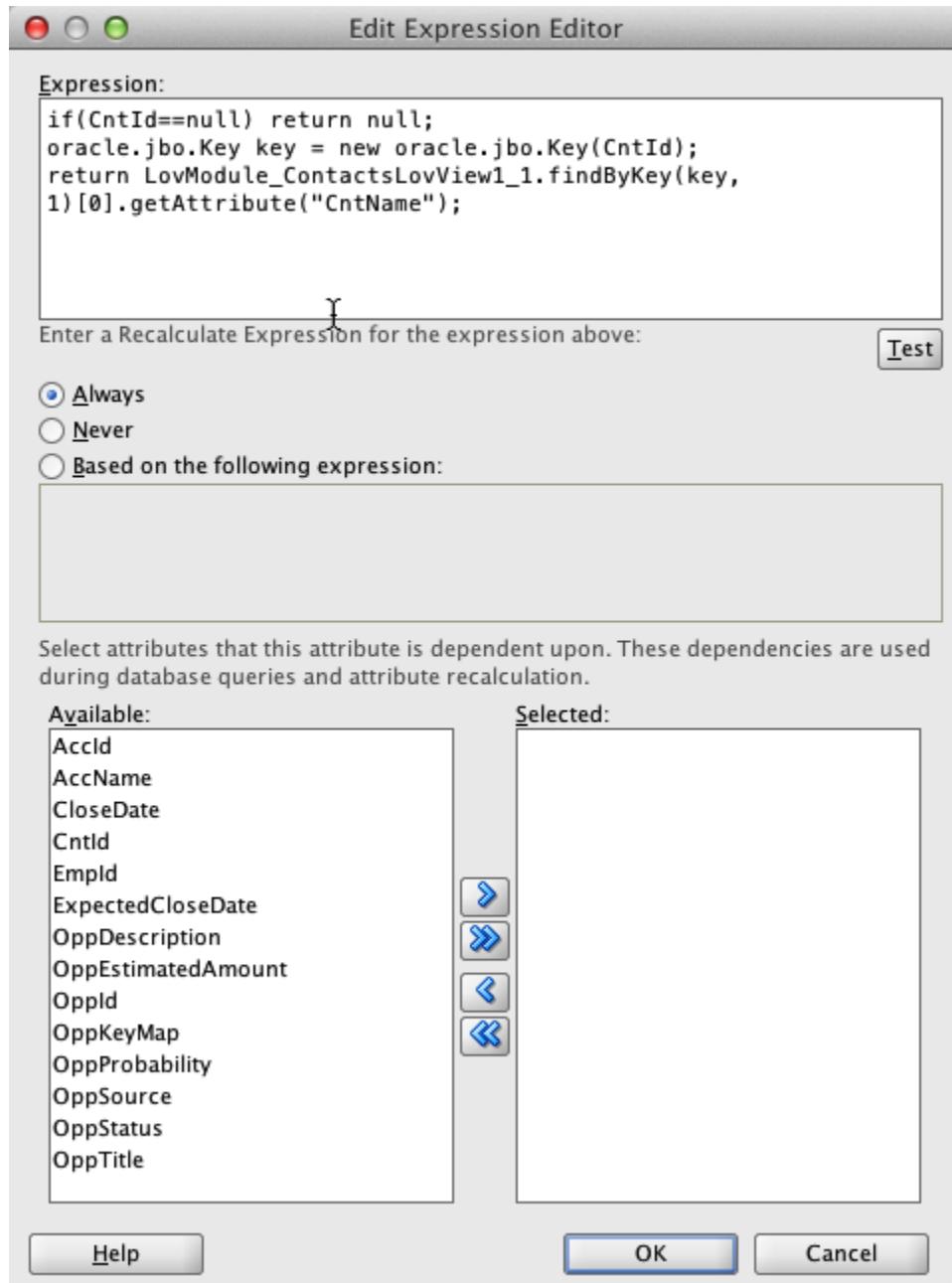
1. Select Default List Type to Combo Box with List Of Values
2. Select and slide CntName to the right
3. Uncheck Query Limit checkbox
4. Click OK

Set CntName Expression



1. Set Value Type to Expression
2. Click the pencil next to Value field

Edit CntName Value Expression



- Paste the following expression. Make sure the LovModule_ContactsLovView1_1 matches the string you copied and stored before

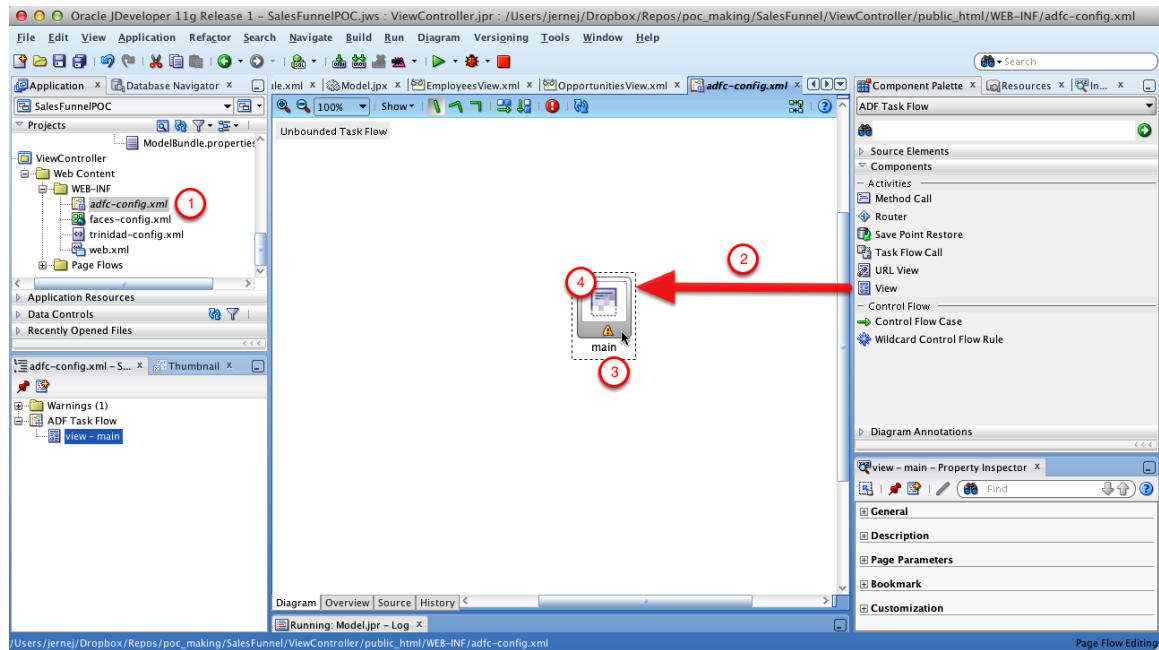
```
if(CntId==null) return null;
oracle.jbo.Key key = new oracle.jbo.Key(CntId);
return LovModule_ContactsLovView1_1.findByKey(key, 1)[0].getAttribute("CntName");
```

II. Creating User Interface and Security Foundation

14. Main Form

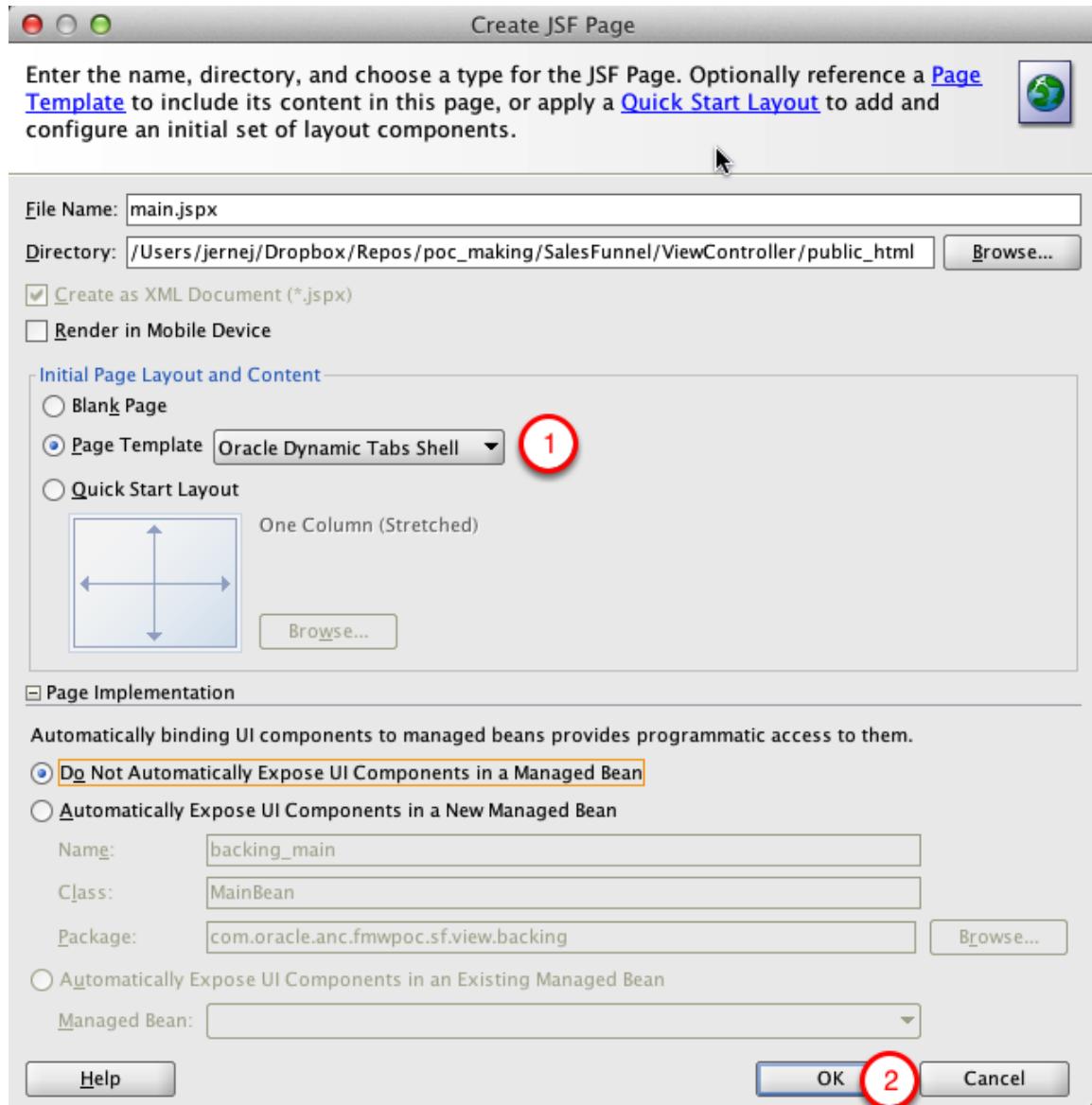
With our model ready, we'll now move to the UI.
First we'll create the main form for our application.

Add main view to the unbounded task flow



1. In ViewController project, locate and open adfc-config.xml
2. From the Component Palette drag and drop View on the Diagram
3. Set name to main
4. Double-click the icon

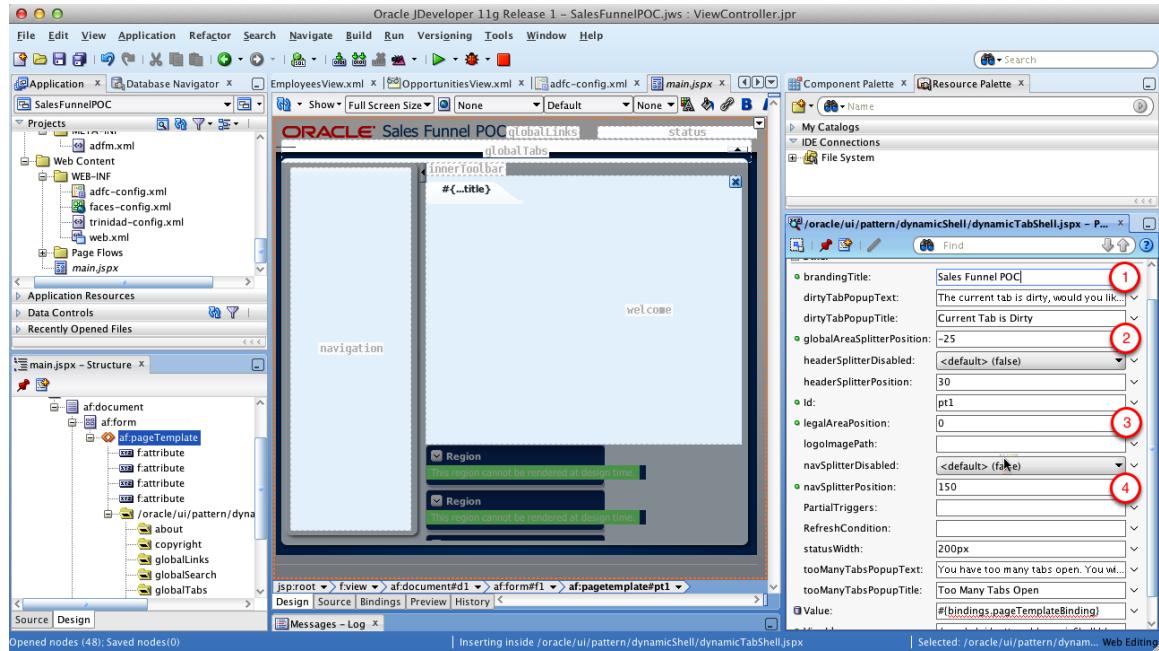
Create JSF Page



1. Set Page Template to Oracle Dynamic Tabs Shell
2. Click OK

You can read more about UI Shell Pattern here: <http://www.oracle.com/technetwork/developer-tools/adf/uishell-093084.html>

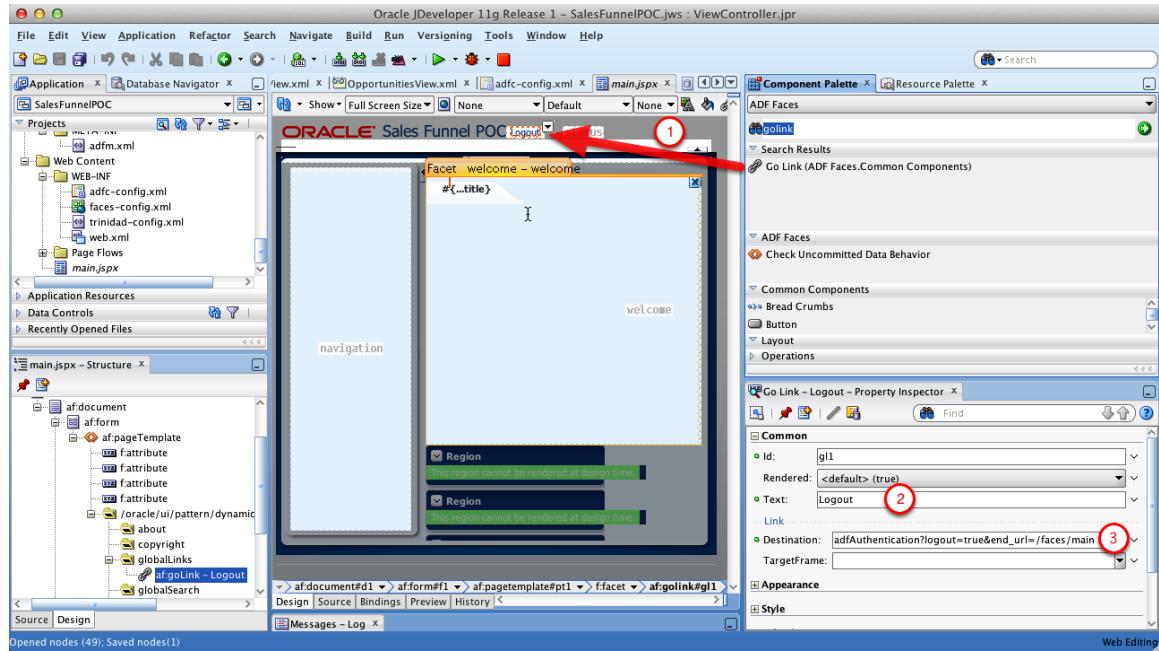
Set main form page template attributes



With pageTemplate selected in the Structure window:

1. Set brandingTitle to: Sales Funnel POC
2. Set globalAreaSplitterPosition to: -25 Gloable Tool bar
3. Set legalAreaPosition to: 0
4. Set navSplitterPosition to: 150 Navigation Area

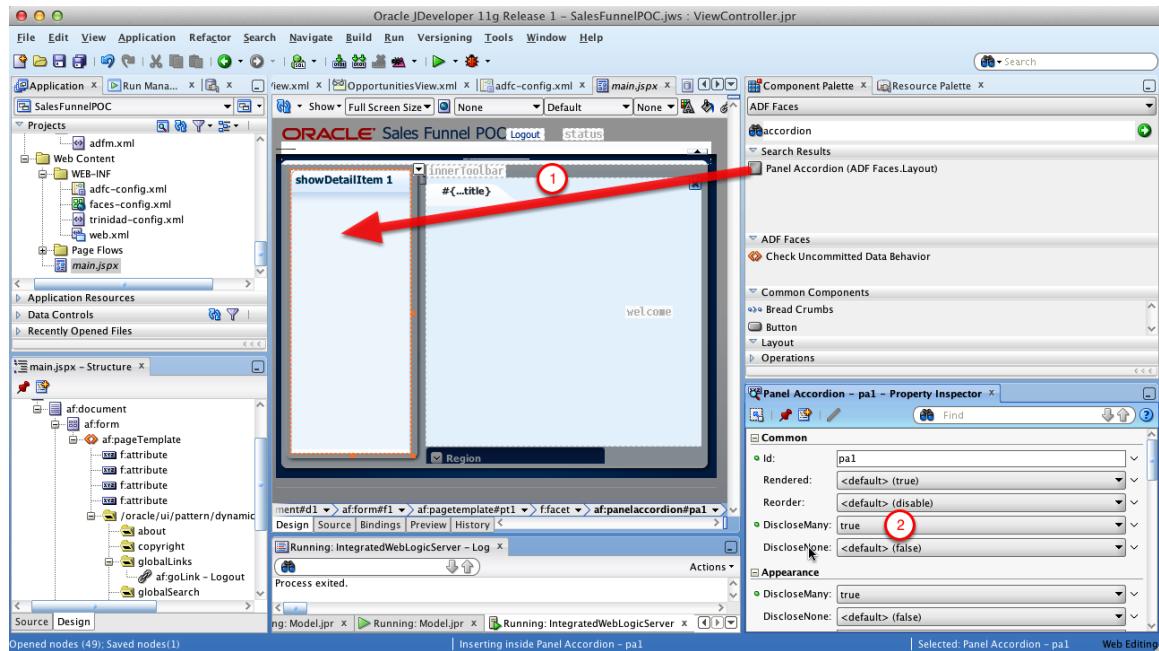
Add logout link



1. Locate Go Link in Components Palette* and Drag and drop it into globalLinks facet
2. Set Text to Logout
3. Set Destination to: `adfAuthentication?logout=true&end_url=/faces/main`

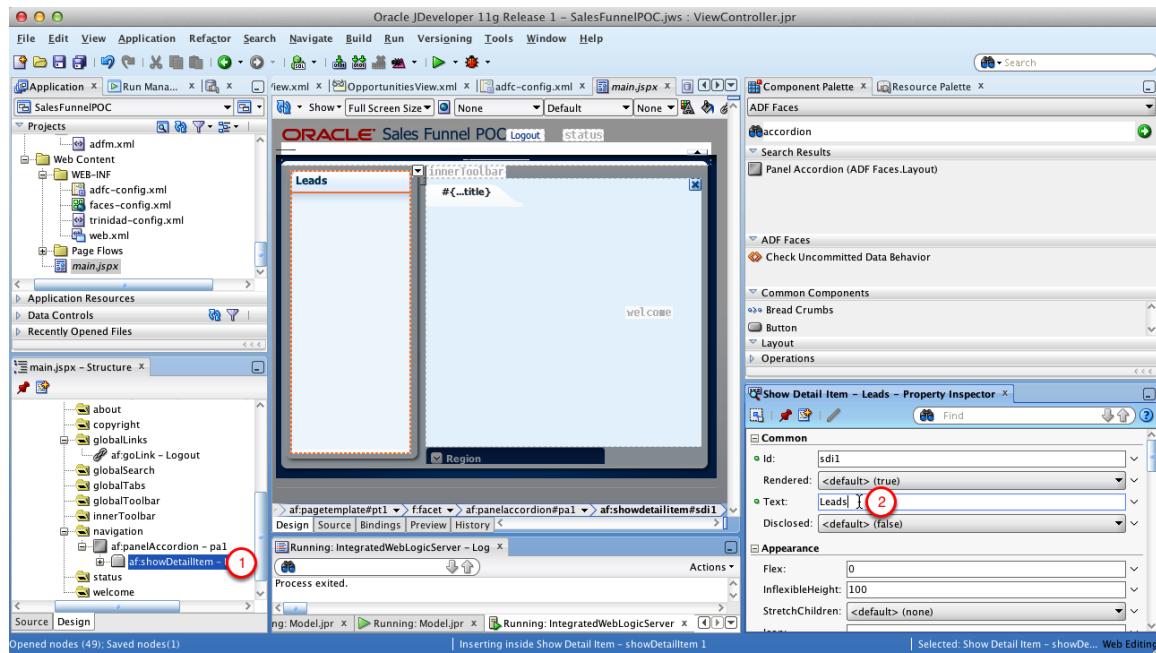
*You can use search to find components, as you can see on the screenshot

Build navigation menu



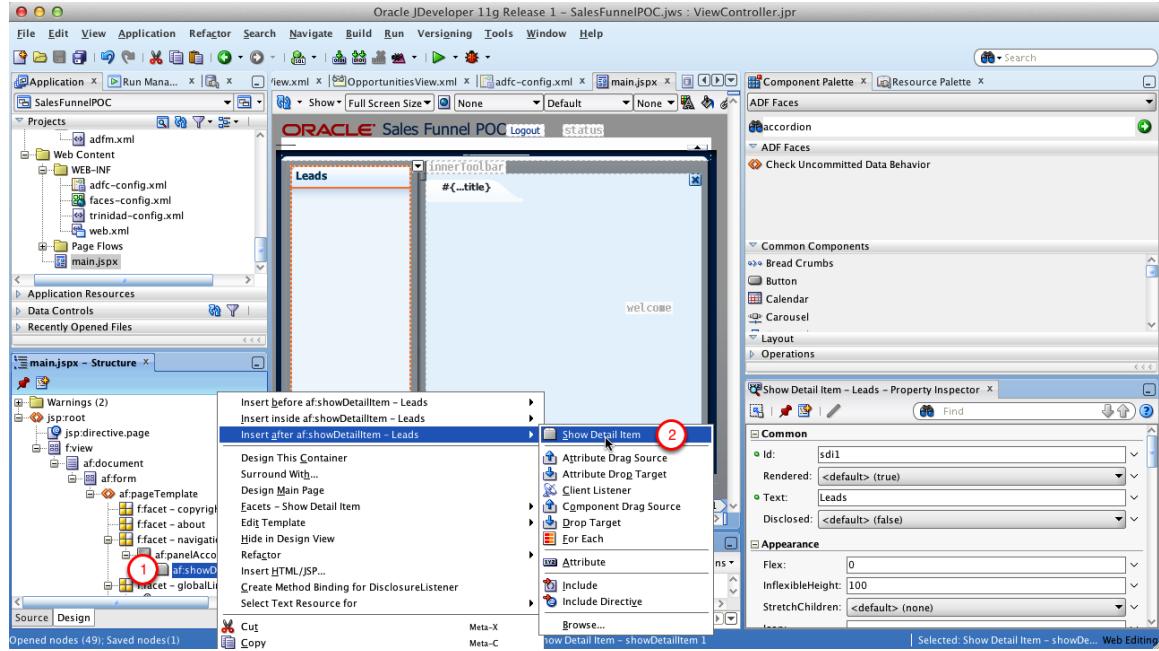
1. Find Panel Accordion in to Component Palette and Drag and drop it to the Navigation facet
2. Set DiscloseMany to true

Add navigation group



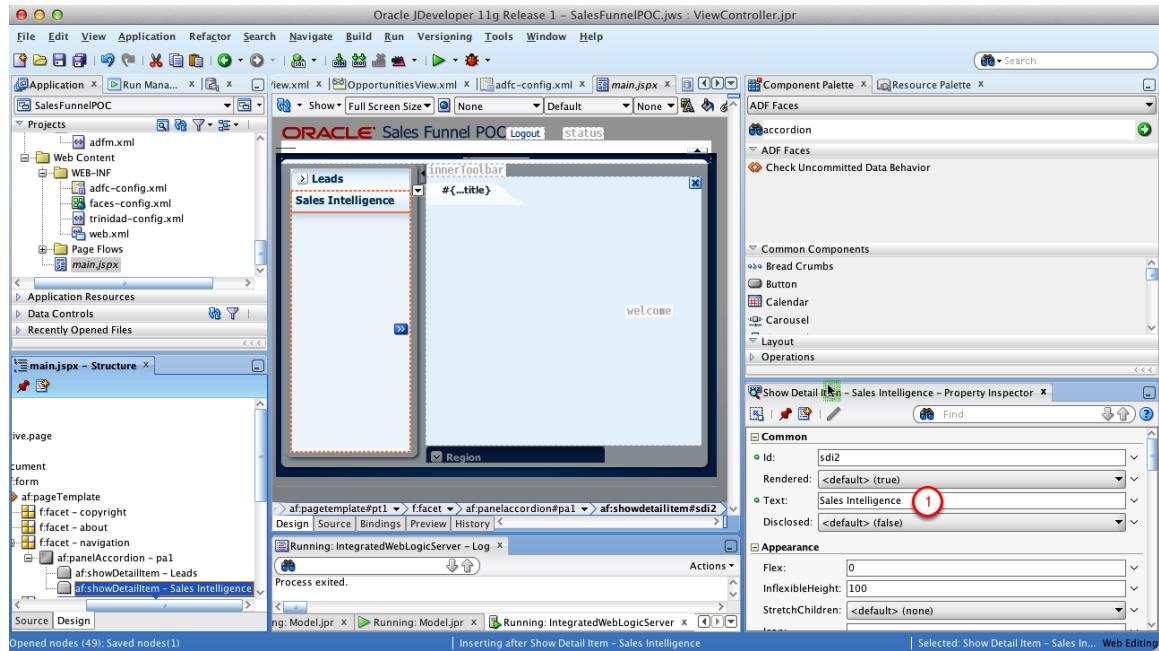
1. Select showDetailItem inside panelAccordion
2. Set Text to Leads

Add navigation group



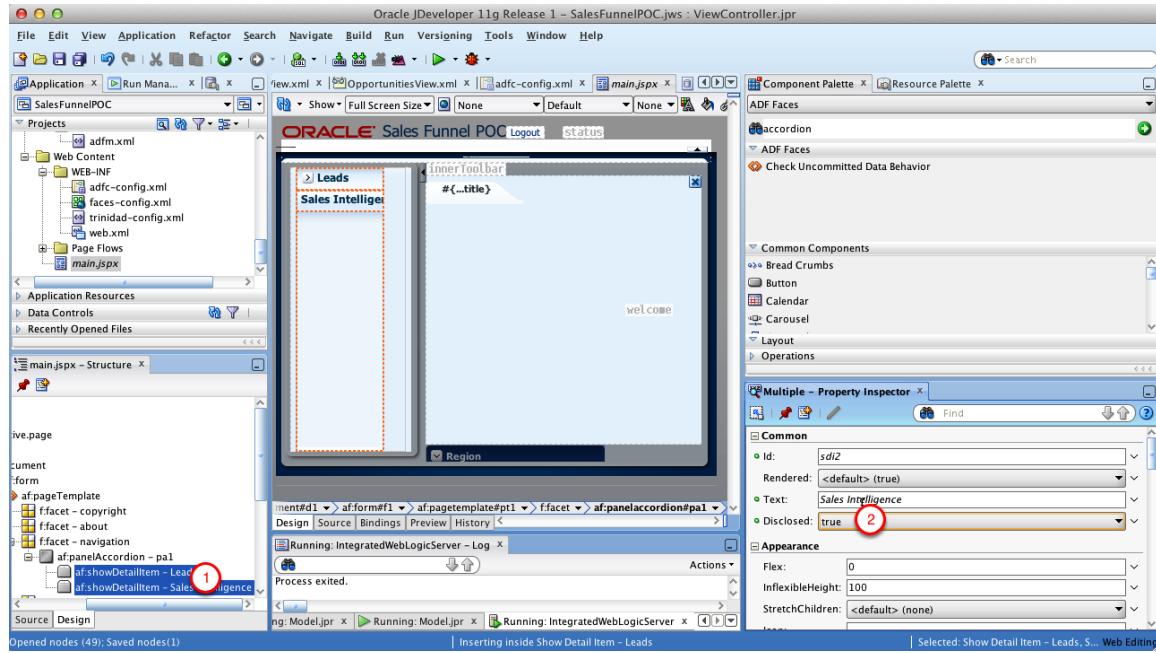
1. Right-click showDetailItem
2. Select insert after... Show Detail Item in the popup menu

Add navigation group



1. With the new showDetailItem selected, set Text to Sales Intelligence

Set Navigation group properties



1. Select both showDetailItems
2. Set Disclosed to true

15. Login

Copy Login Page

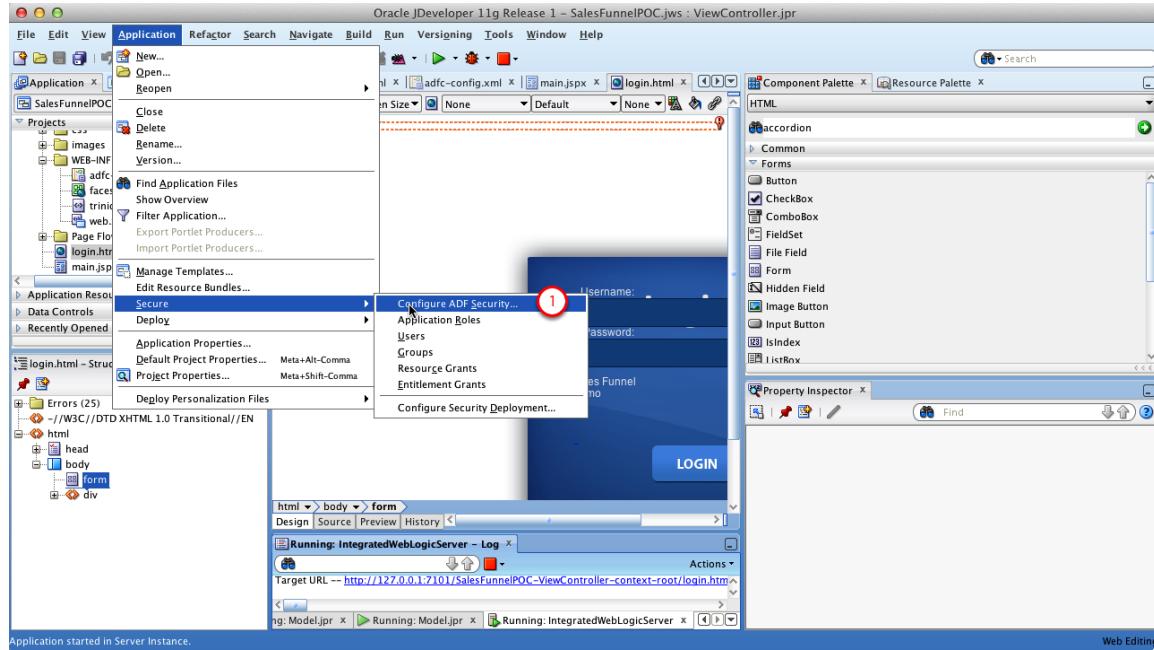
Name	Date Modified	Size	Kind
.DS_Store	Today 5:35 PM	6 KB	Document
css	Sep 4, 2011 7:10 PM	846 bytes	Folder
images	Sep 4, 2011 7:10 PM	80 KB	Folder
login.html	Sep 4, 2011 7:10 PM	3 KB	HTML Document
main.jspx	Today 4:32 PM	2 KB	Document
WEB-INF	Today 3:38 PM	291 KB	Folder

Copy paste login.html, images and css

Note: the files will be distributed to you by the trainer

16. Security

Start ADF Configuration Wizard



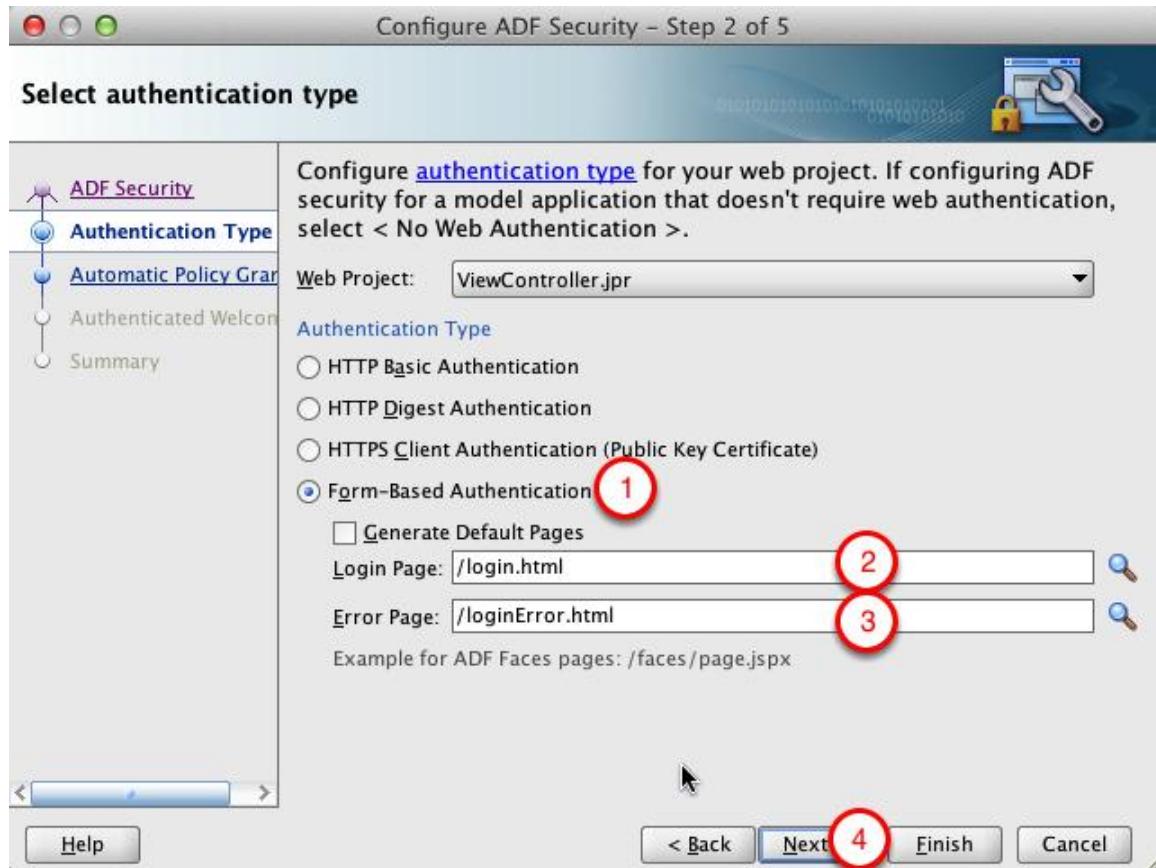
1. In the main menu click Application->Secure->Configure ADF Security

Configure ADF Security - Step 1 of 5



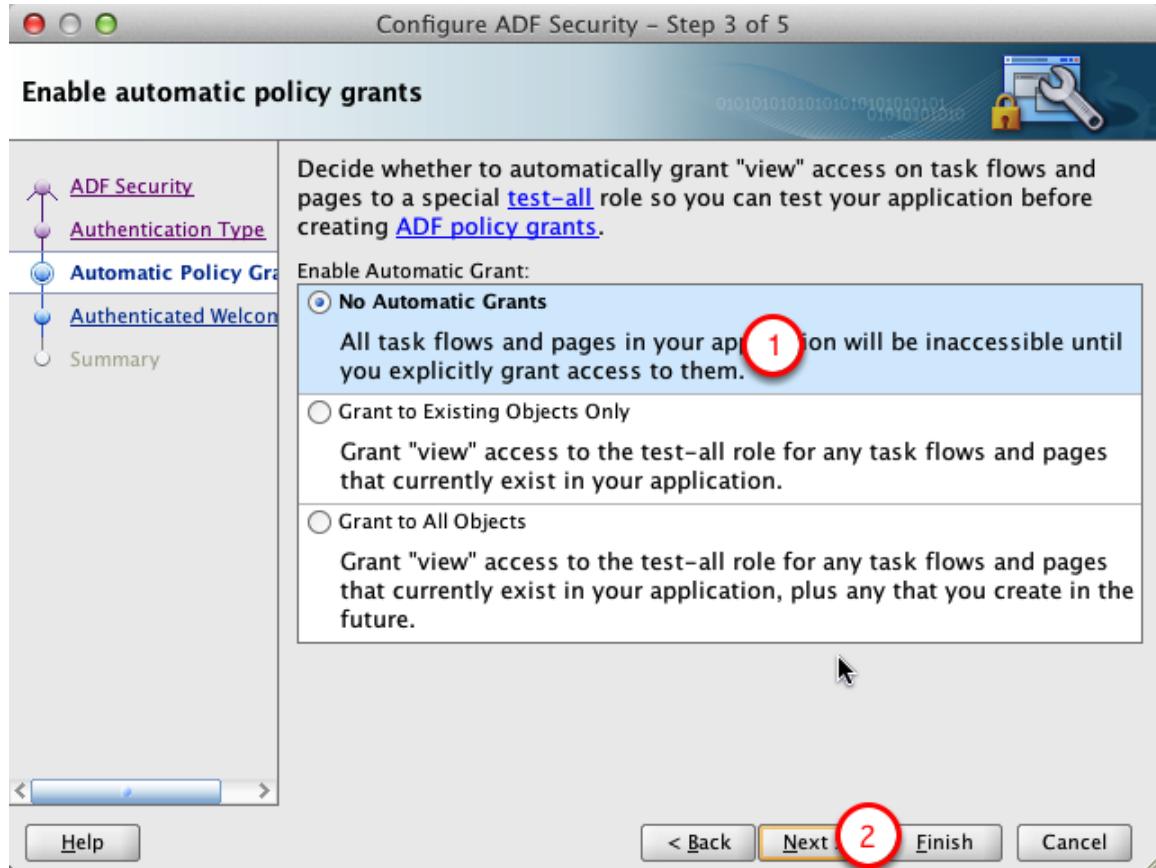
1. Select ADF Authentication and Authorization security model
2. Click Next

Configure ADF Security - Step 2 of 5



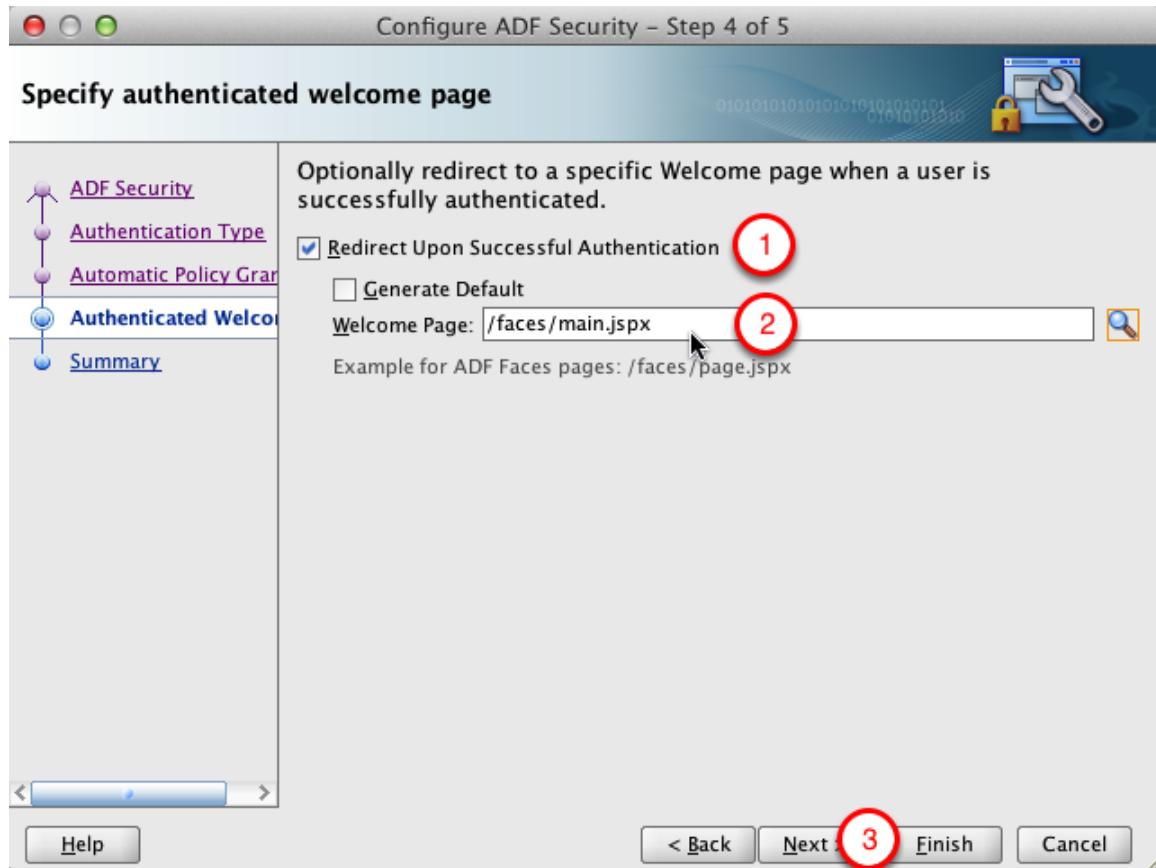
1. Select Form-Based Authentication
2. Set /login.html as Login Page
3. Set /loginError.html as Error page
4. Click Next

Configure ADF Security - Step 3 of 5



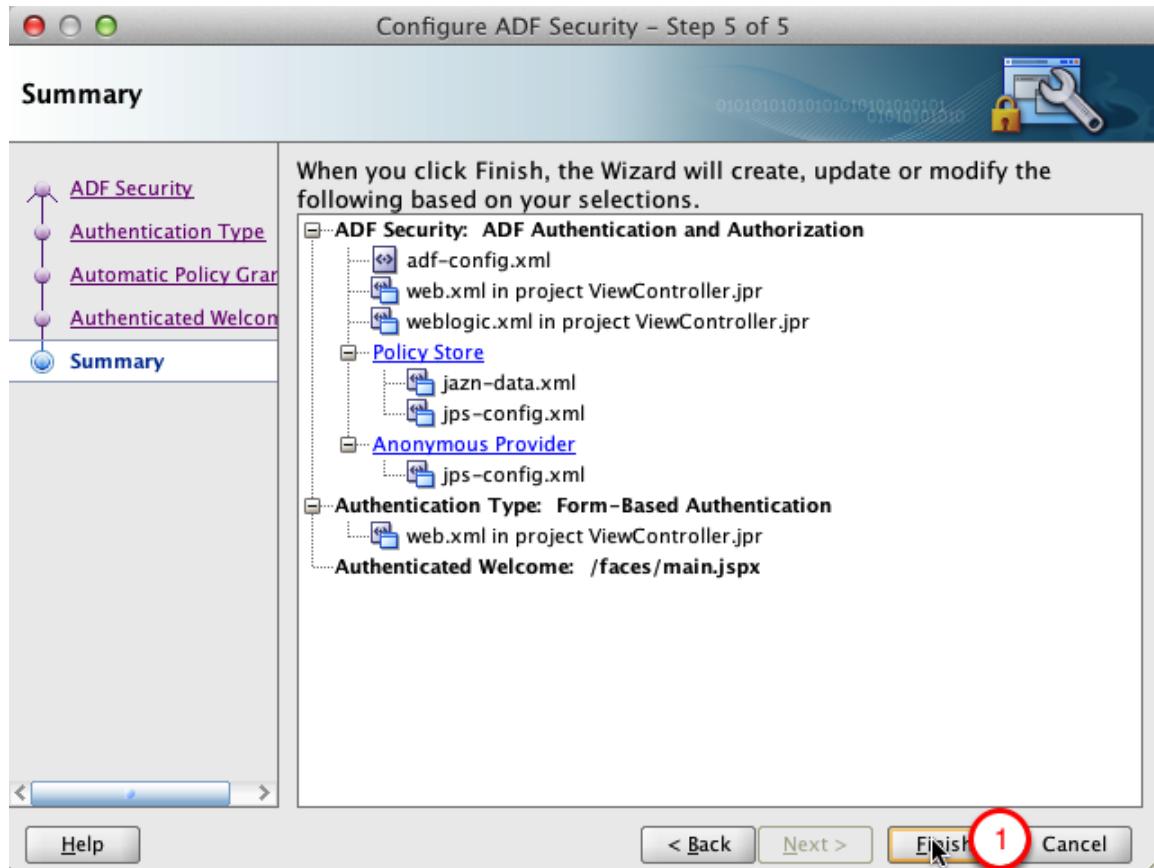
1. Select No Automatic Grants
2. Click Next

Configure ADF Security - Step 4 of 5



1. Check Redirect on Successful Authentication
2. Set /faces/main.jspx as Welcome Page
3. Click Next

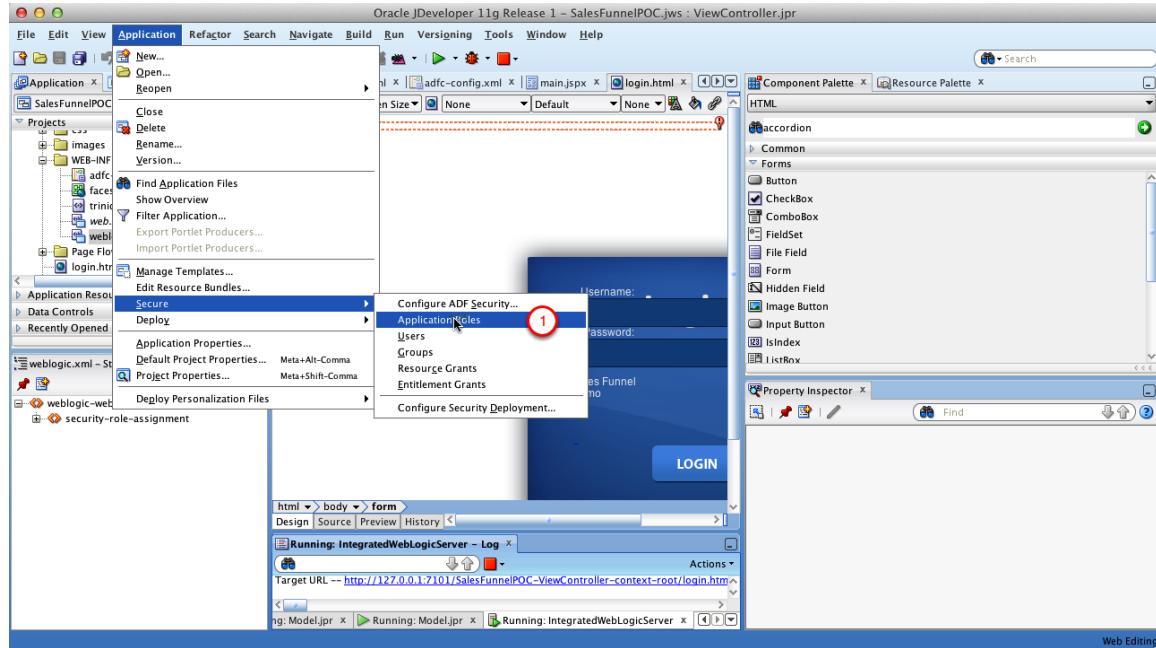
Configure ADF Security - Step 5 of 5



1. Click Finish

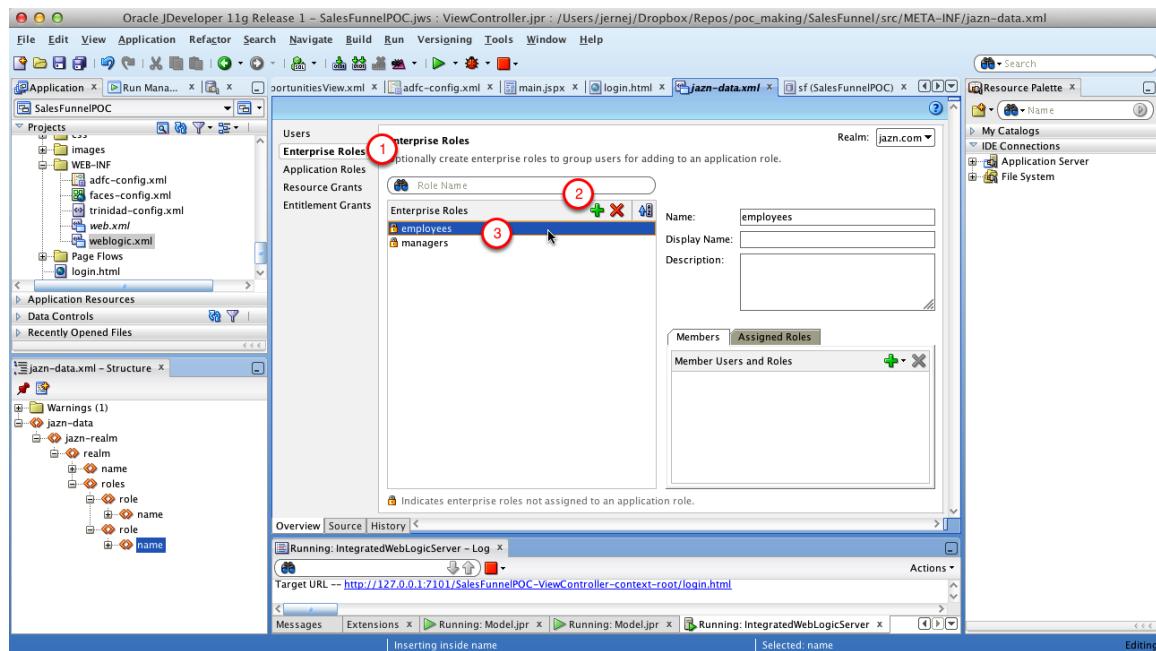
17. Roles and grants configuration

Open Security Configuration



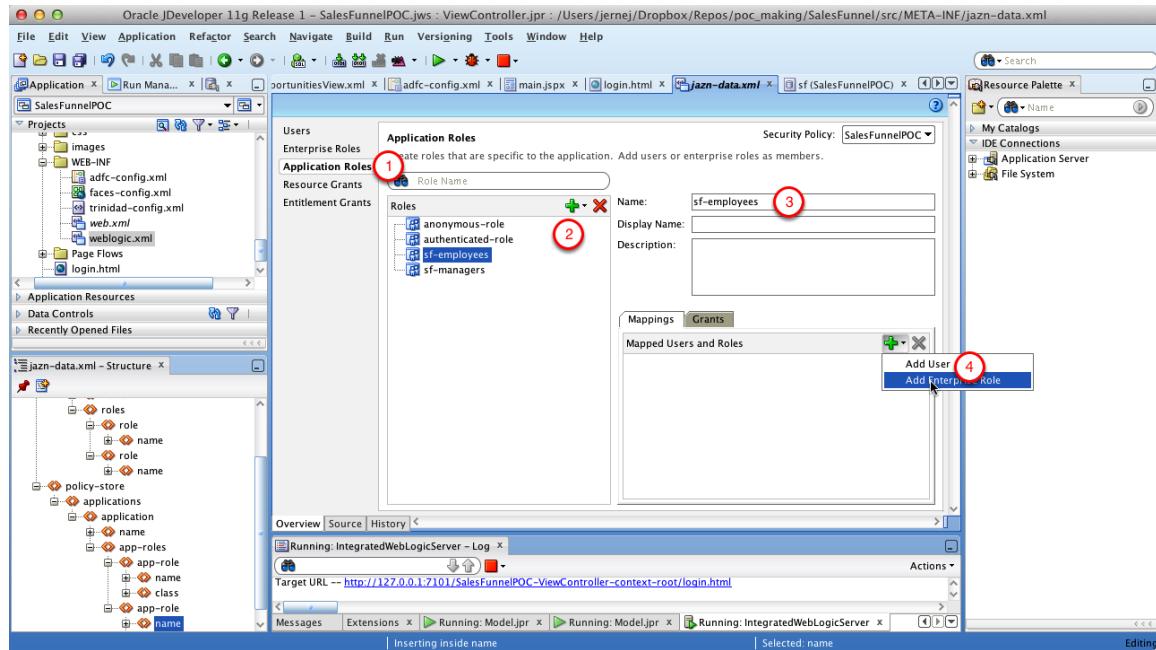
1. In the main menu select Application->Secure->Application Roles

Add Enterprise Roles



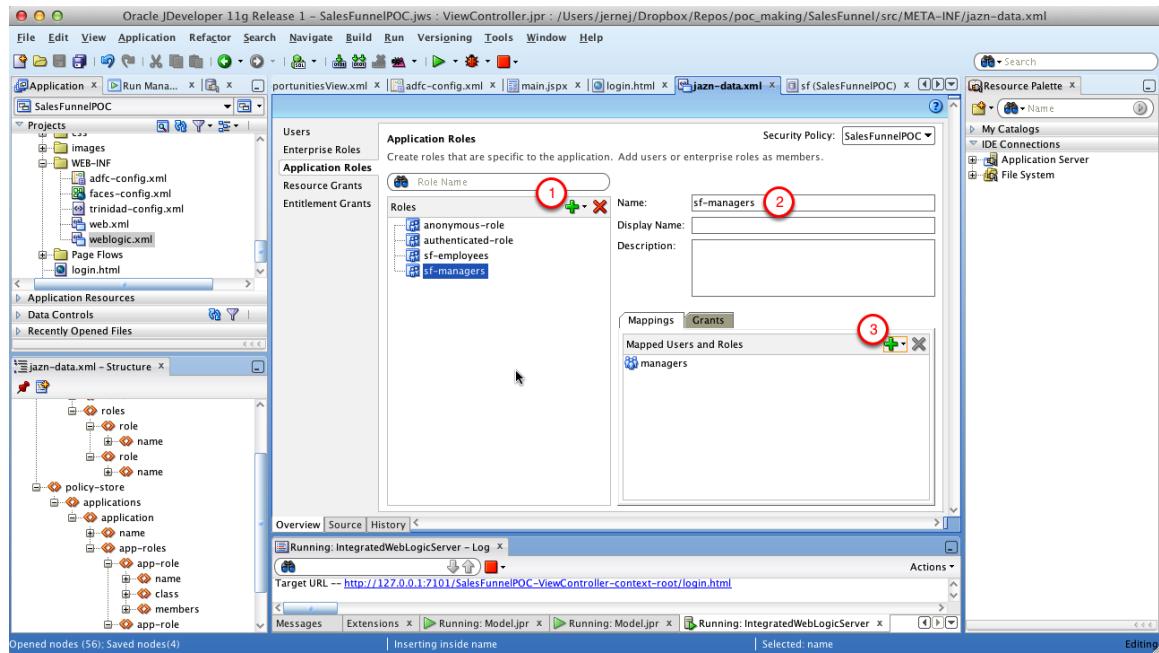
1. Open Enterprise Roles tab
2. Click +
3. Add employees and managers roles

Add sf-employees application role



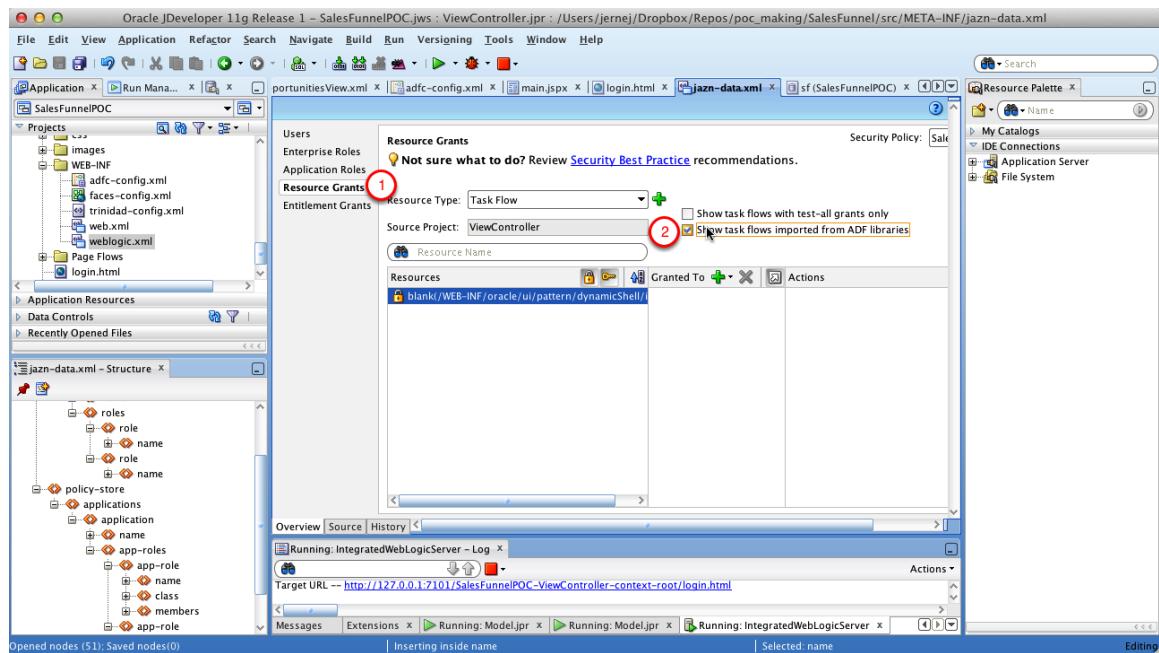
1. Open Application Roles tab
2. Click + to create new Role
3. Name it sf-employees
4. Click + in Mappings and select Add Enterprise Role
5. Add employees enterprise role

Add sf-managers application role



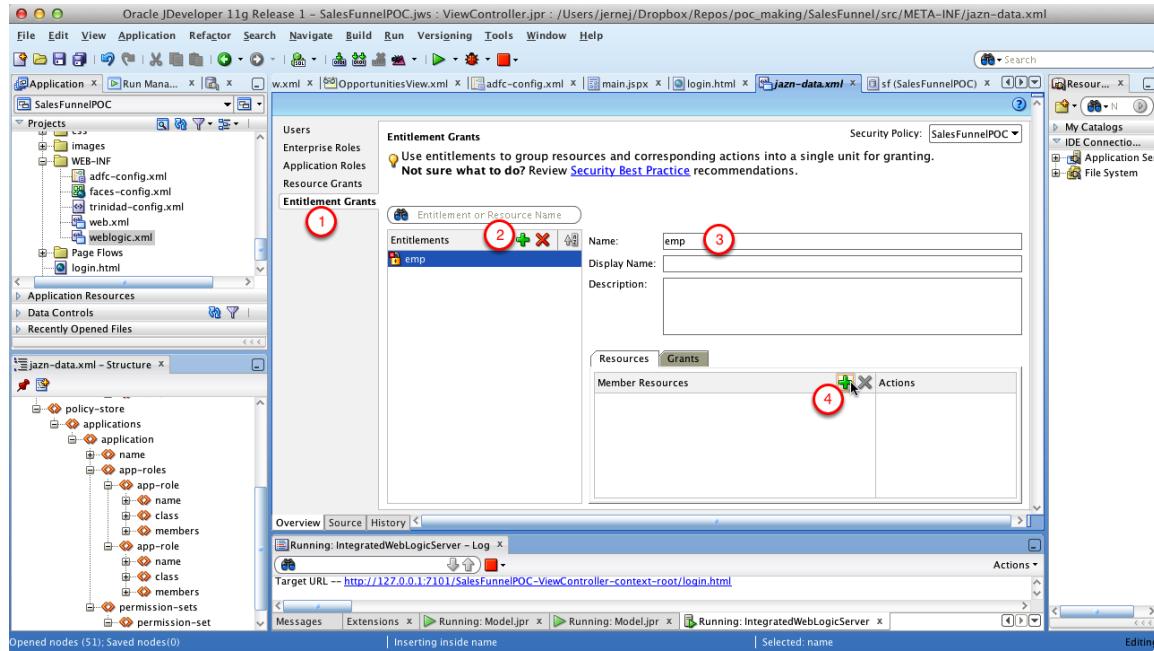
1. Create new application role
2. Name it sf-managers
3. Add managers enterprise role

Configure resource grants



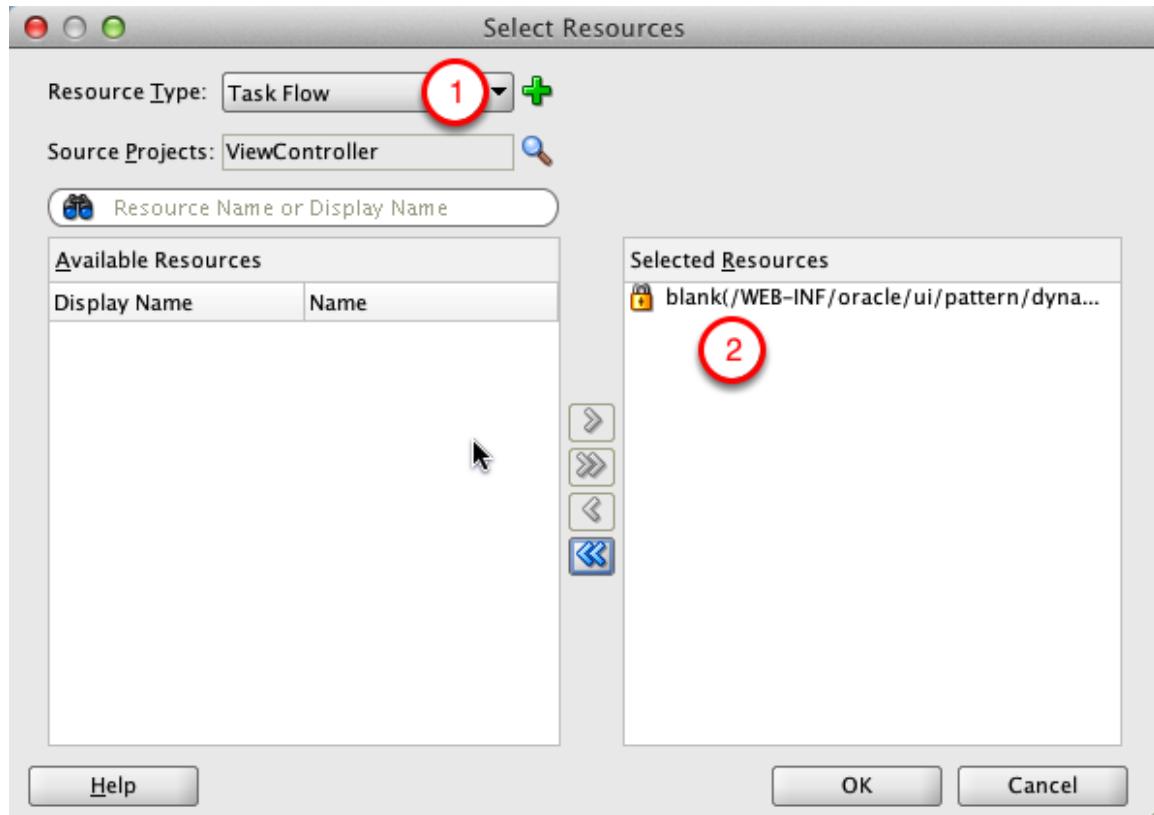
1. Open Resource Grants tab
2. Check Show task flows imported from ADF libraries tab

Add resources to emp grant



1. Open Entitlement Grants tab
2. Click +
3. Name the entitlement "emp"
4. Click + in resources tab

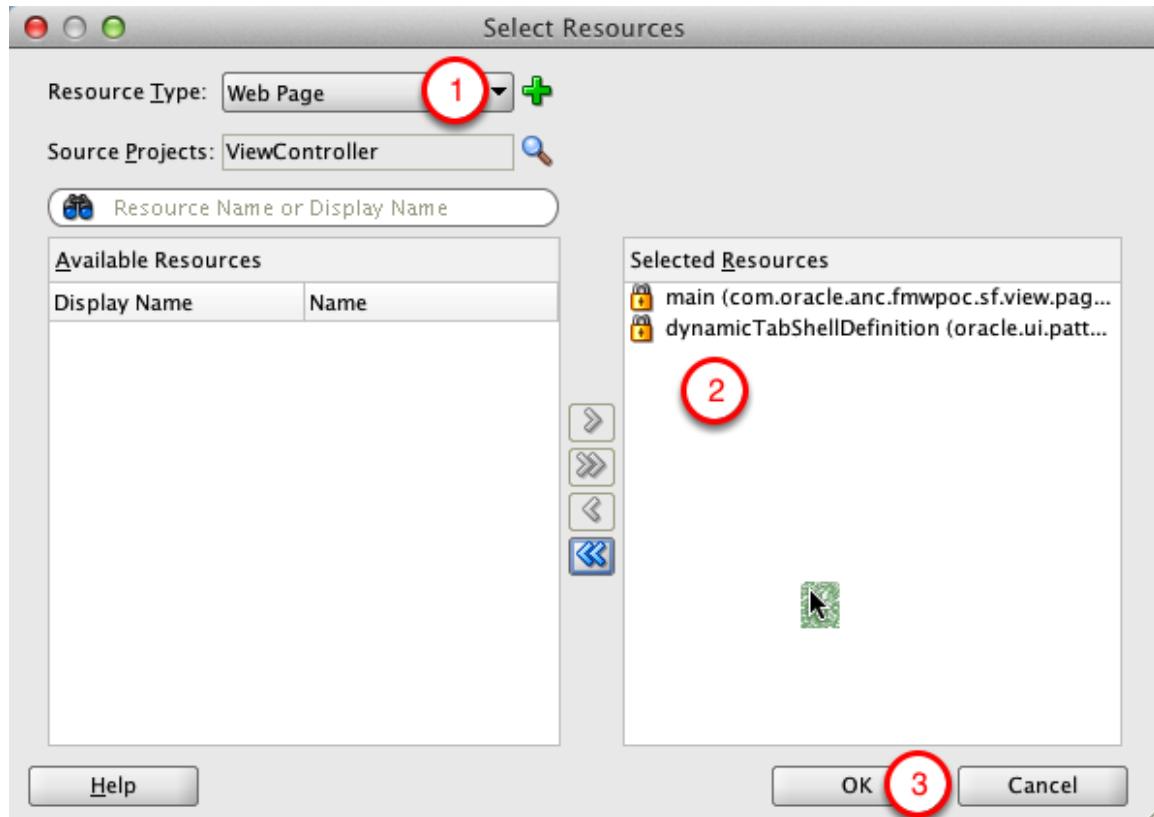
Select Resources



1. Set resource Type to Task Flow
2. Select and slide blank to the right

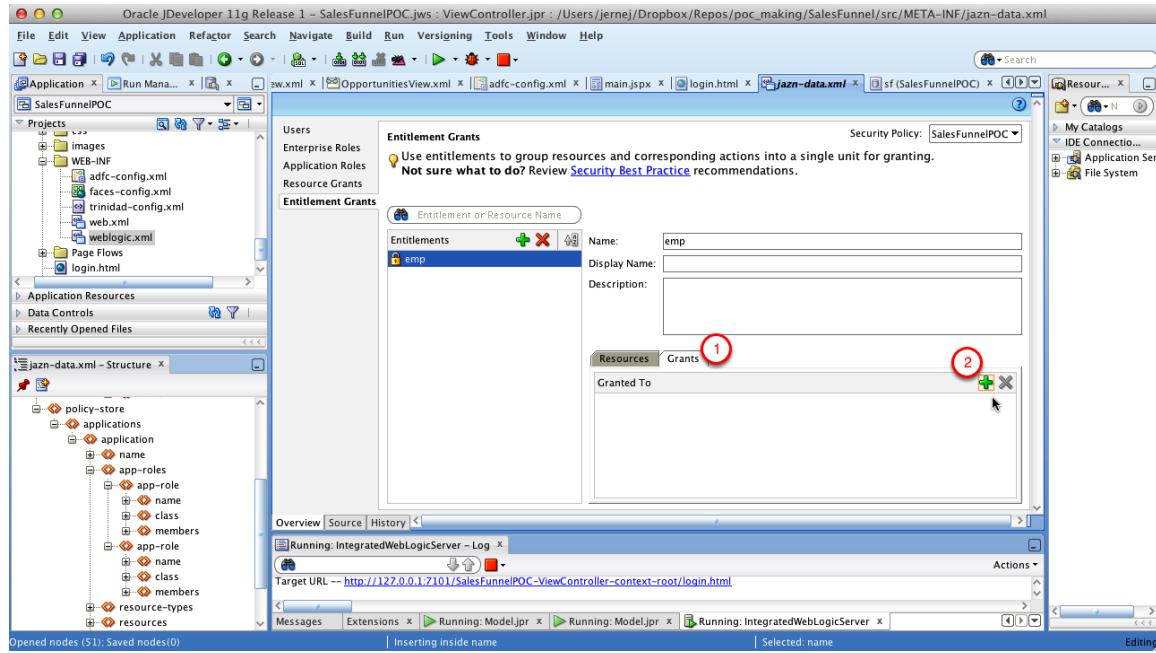
Note: blank is a special task flow from the UI Shell pattern and has to be included in order for UI Shell to work.

Select Resources



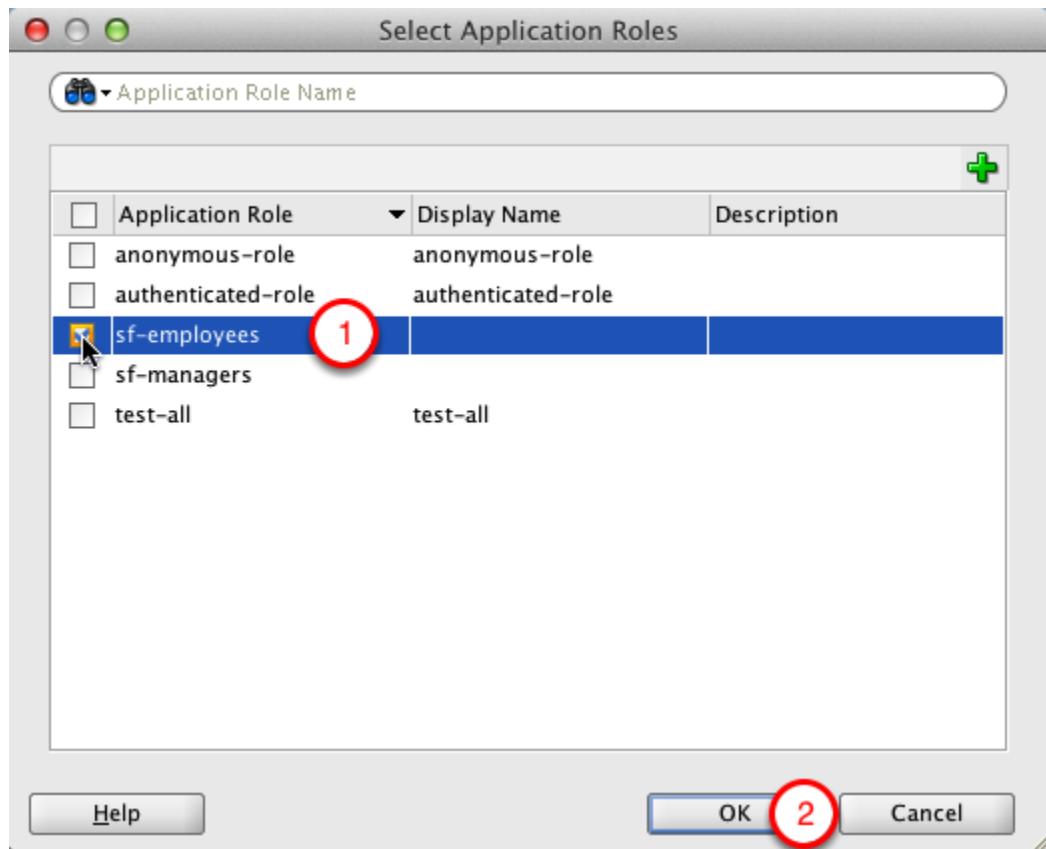
1. Change Resource Type to Web page
2. Slide main and dynamicTabShellDefinition to the right
3. Click OK

Add members to emp grant



1. Open Grants tab
2. Click +

Select Application Roles



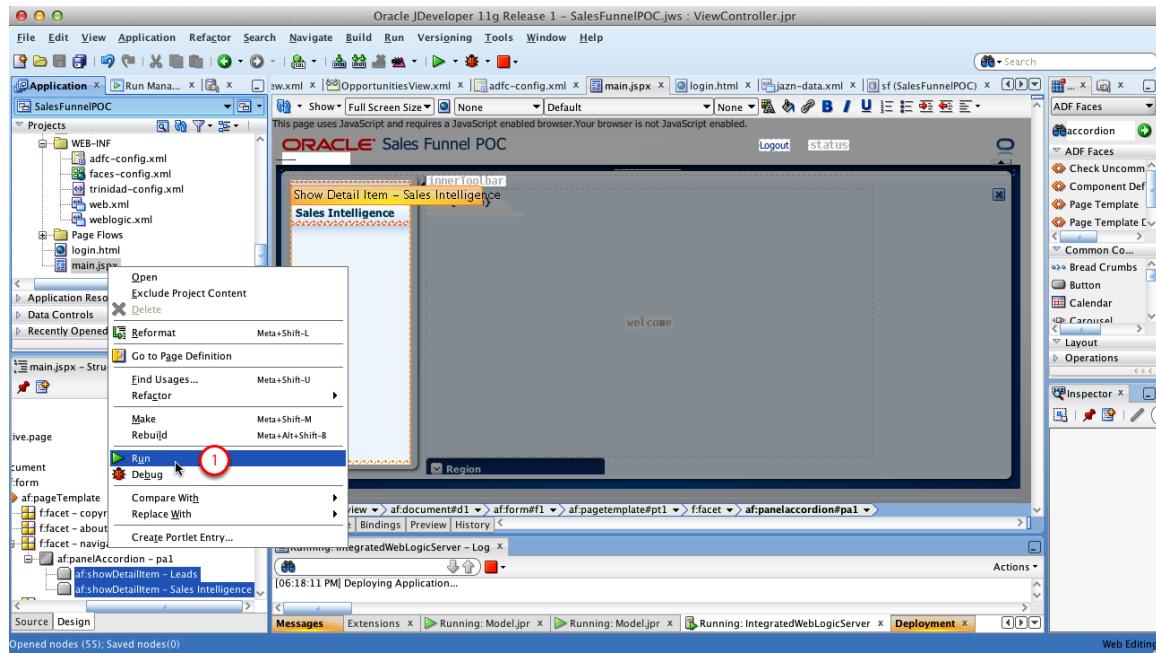
1. Check sf-employees role and click OK

18. WLS security configuration

Since our application will use usernames and passwords stored in the database we need to configure the server to support that.

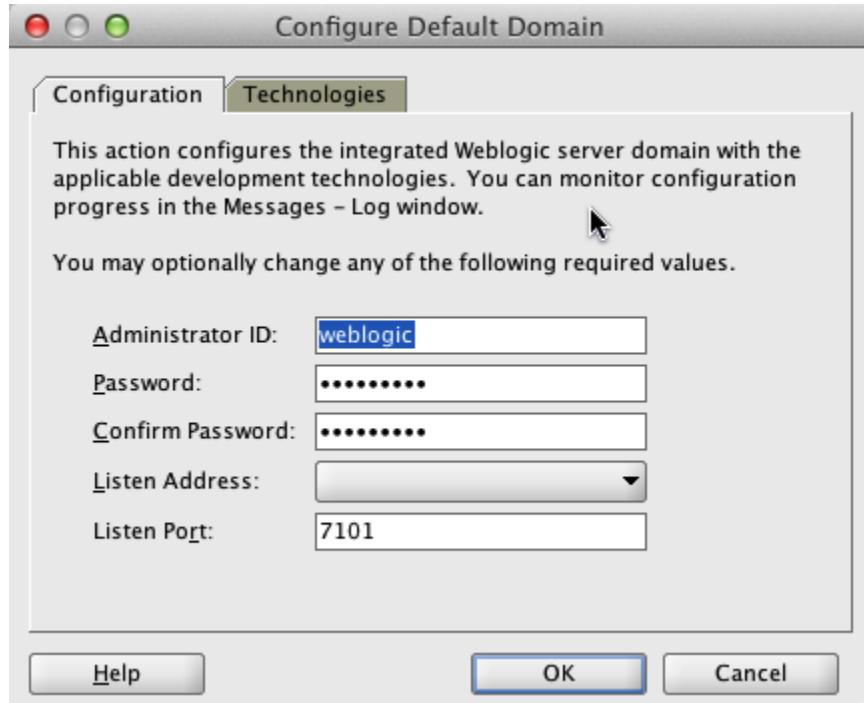
The application itself is authenticator agnostic and it relies completely on the application server to handle authentication.

Start the integrated WLS application server



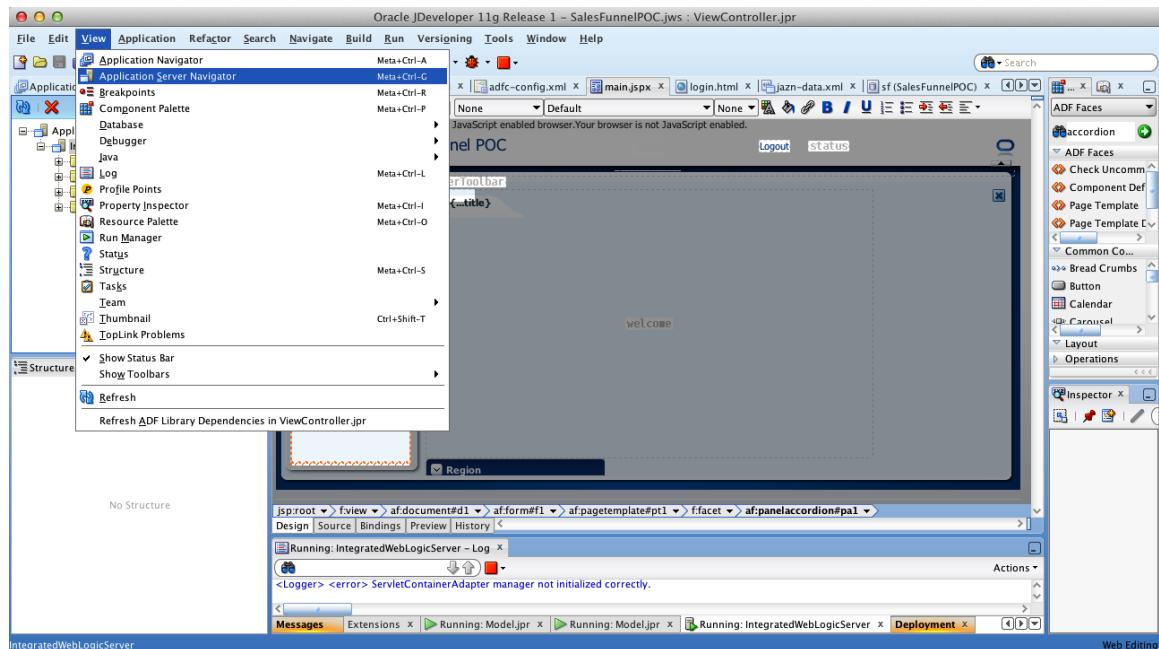
1. Right-click main.jspx and click Run in the popup menu

Configure Default Domain



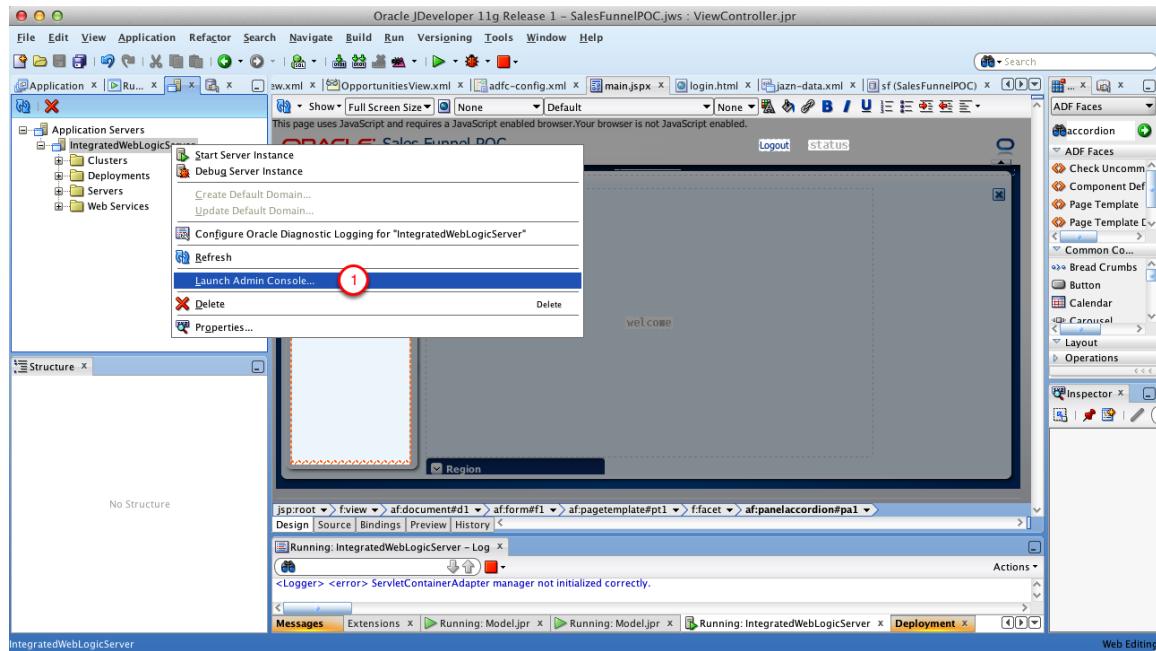
1. If you are starting web logic for the first time, it will ask for default configuration. Enter password and remember it. Leave other options to default values.
2. Click OK

Open Application Server Navigator



1. In the menu, click View->Application Server Navigator

Launch Admin Console



1. Expand application servers, right click on IntegratedWeblogicServer and click Launch Admin Console

Oracle WebLogic Server Administration Console



1. Enter credentials and click Login

Open Data Sources configuration

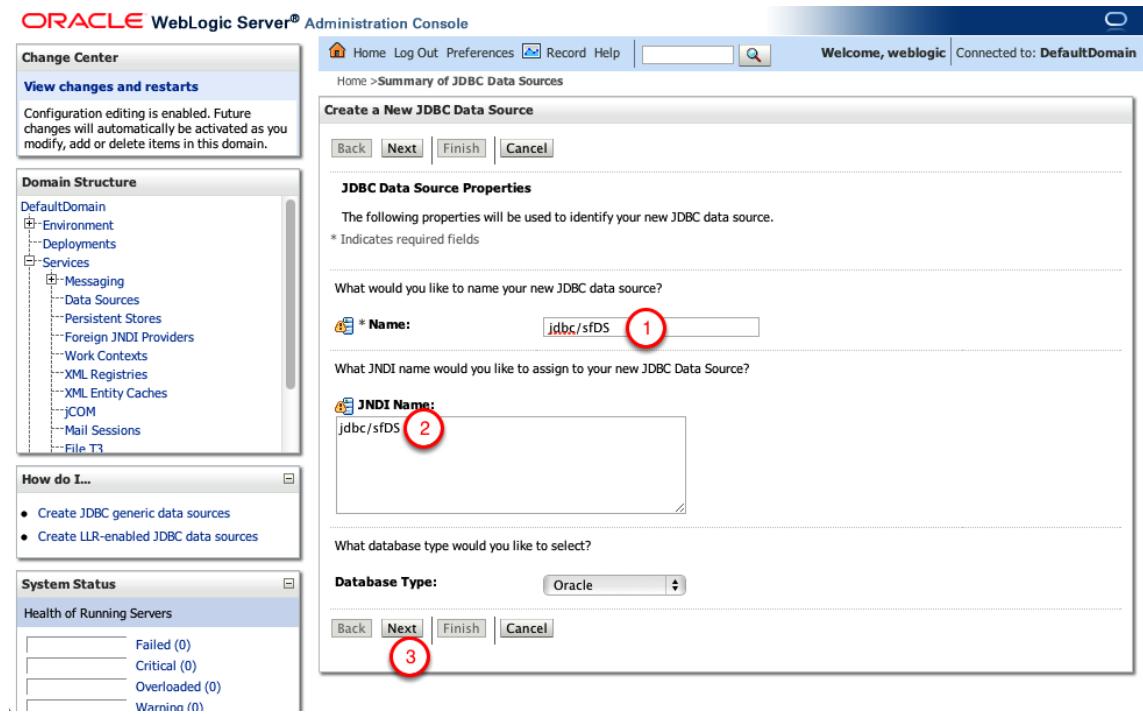
1. Expand Services and open Data Sources

Create new Data Source

The screenshot shows the Oracle WebLogic Server Administration Console interface. The title bar reads "ORACLE WebLogic Server® Administration Console". The main content area is titled "Summary of JDBC Data Sources". Below it, there are tabs for "Configuration" and "Monitoring". A message states: "A JDBC data source is an object bound to the JNDI tree that provides database connectivity through a pool of JDBC connections. Applications can look up a data source on the JNDI tree and then borrow a database connection from a data source." Another message says: "This page summarizes the JDBC data source objects that have been created in this domain." Below these messages is a table titled "Data Sources (Filtered - More Columns Exist)". The table has columns for "Type", "JNDI Name", and "Targets". There are three rows: "Generic Data Source" (highlighted with a red circle), "GridLink Data Source", and "Multi Data Source". A tooltip for the "New" button says: "Creates a new JDBC data source." At the bottom of the table, a message says: "There are no items to display". On the left side of the screen, there is a "Domain Structure" tree view under "DefaultDomain" which includes "Environment", "Deployments", "Services", and "Messaging" with "Data Sources" selected. Below the tree is a "How do I..." section with links for creating various types of data sources. A "System Status" section at the bottom shows "Health of Running Servers".

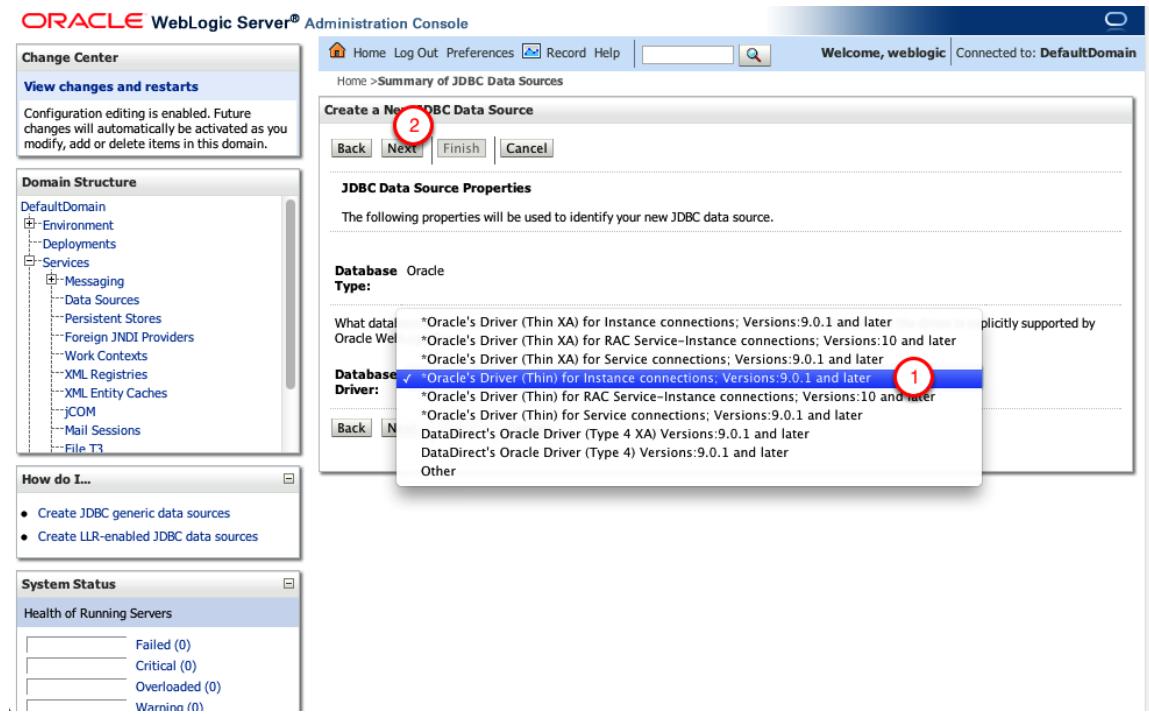
1. Click New -> Generic Data Source

Configure New Data Source



1. Enter "jdbc/sfDS" as Name
2. Enter "jdbc/sfDS" as JNDI Name
3. Click Next

Configure New Data Source



1. Select Oracle Driver for Instance connections
2. Click Next

Configure New Data Source

ORACLE WebLogic Server® Administration Console

Change Center

View changes and restarts

Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Domain Structure

- DefaultDomain
 - + Environment
 - Deployments
 - Services
 - + Messaging
 - Data Sources
 - Persistent Stores
 - Foreign JNDI Providers
 - Work Contexts
 - XML Registries
 - XML Entity Caches
 - jCOM
 - Mail Sessions
 - File T3

How do I...

- Create JDBC generic data sources
- Create LLR-enabled JDBC data sources

System Status

Health of Running Servers

- Failed (0)
- Critical (0)
- Overloaded (0)
- Warning (0)

Create a New JDBC Data Source

Back | **Next** | Finish | Cancel

Transaction Options

You have selected non-XA JDBC driver to create database connection in your new data source.

Does this data source support global transactions? If yes, please choose the transaction protocol for this data source.

Supports Global Transactions

Select this option if you want to enable non-XA JDBC connections from the data source to participate in global transactions using the *Logging Last Resource* (LLR) transaction optimization. Recommended in place of Emulate Two-Phase Commit.

Logging Last Resource

Select this option if you want to enable non-XA JDBC connections from the data source to emulate participation in global transactions using JTA. Select this option only if your application can tolerate heuristic conditions.

Emulate Two-Phase Commit

Select this option if you want to enable non-XA JDBC connections from the data source to participate in global transactions using the one-phase commit transaction processing. With this option, no other resources can participate in the global transaction.

One-Phase Commit

Back | **Next** | Finish | Cancel

1

1. Click Next

Configure New Data Source database connection

The screenshot shows the Oracle WebLogic Server Administration Console. On the left, there's a sidebar with 'Change Center' and 'View changes and restarts'. Below that is the 'Domain Structure' tree, which includes 'DefaultDomain' with 'Environment', 'Deployments', 'Services' (which has 'Messaging', 'Data Sources', 'Persistent Stores', etc.), and 'File T3'. There's also a 'How do I...' section with links to 'Create JDBC generic data sources' and 'Create LLR-enabled JDBC data sources'. On the right, the main area is titled 'Create a New JDBC Data Source' under 'Connection Properties'. It asks for the database name ('Database Name: xe'), host name ('Host Name: localhost'), port ('Port: 1521'), database user name ('Database User Name: sf'), password ('Password: sf'), and confirm password ('Confirm Password: sf'). The 'Next' button is highlighted.

Enter your database connection properties. If you are using Oracle Xe you can enter:

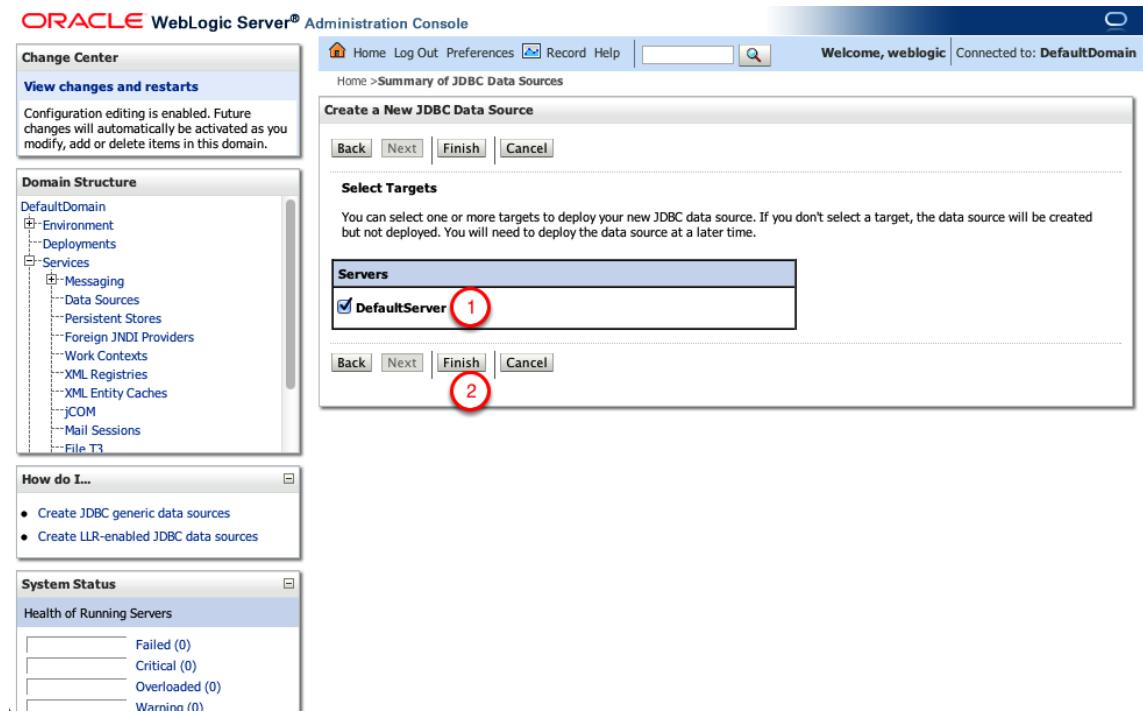
1. Database Name: xe
2. Host Name: localhost
3. Database User Name: sf
4. Password: sf
5. Confirm Password: sf
6. Click Next

Test Data Source configuration

The screenshot shows the Oracle WebLogic Server Administration Console. The left sidebar has a 'Domain Structure' tree with 'Data Sources' selected. A 'How do I...' panel lists 'Create JDBC generic data sources' and 'Create LLR-enabled JDBC data sources'. The main area shows the 'Create a New JDBC Data Source' wizard. Step 1: 'Test Configuration' is selected. Step 2: A message says 'Connection test succeeded.' Step 3: A red circle highlights the 'Next' button. The URL is set to 'jdbc:oracle:thin:@local'.

1. Click Test Configuration
2. Make sure it is OK
3. Click Next

Deploy Data Source



1. Check the checkbox next to DefaultServer
2. Click Finish

Open default Security Realm

The screenshot shows the Oracle WebLogic Server Administration Console interface. The title bar reads "ORACLE WebLogic Server® Administration Console". The top navigation bar includes links for Home, Log Out, Preferences, Record, Help, and a search bar. The status bar at the top right says "Welcome, weblogic | Connected to: DefaultDomain".

The main content area is titled "Summary of Security Realms". It contains a brief description of what a security realm is and a note that only one can be set as the default. Below this is a table titled "Realms (Filtered - More Columns Exist)". The table has two columns: "Name" and "Default Realm". There is one entry: "myrealm" with "true" in the "Default Realm" column. Buttons for "New" and "Delete" are visible at the bottom of the table.

On the left side, there is a "Domain Structure" tree view under "DefaultDomain" with nodes: Environment, Deployments, Services, Security Realms (which is circled with '1'), Interoperability, and Diagnostics.

A sidebar on the left lists "How do I..." with options: Configure new security realms, Delete security realms, and Change the default security realm.

At the bottom left, there is a "System Status" section titled "Health of Running Servers" which shows Failed (0), Critical (0), and Overloaded (0).

1. Click Security Realms
2. Click myrealm

Create new Authentication Provider

The screenshot shows the Oracle WebLogic Server Administration Console interface. The main title bar reads "ORACLE WebLogic Server® Administration Console". The top navigation bar includes "Home", "Log Out", "Preferences", "Record", "Help", a search bar, and the text "Welcome, weblogic". Below the navigation is a breadcrumb trail: "Home >Summary of JDBC Data Sources >Summary of Security Realms >myrealm >Providers".

The left sidebar contains a "Change Center" section with "View changes and restarts" and a "Domain Structure" tree view under "DefaultDomain" which includes Environment, Deployments, Services, Security Realms, Interoperability, and Diagnostics.

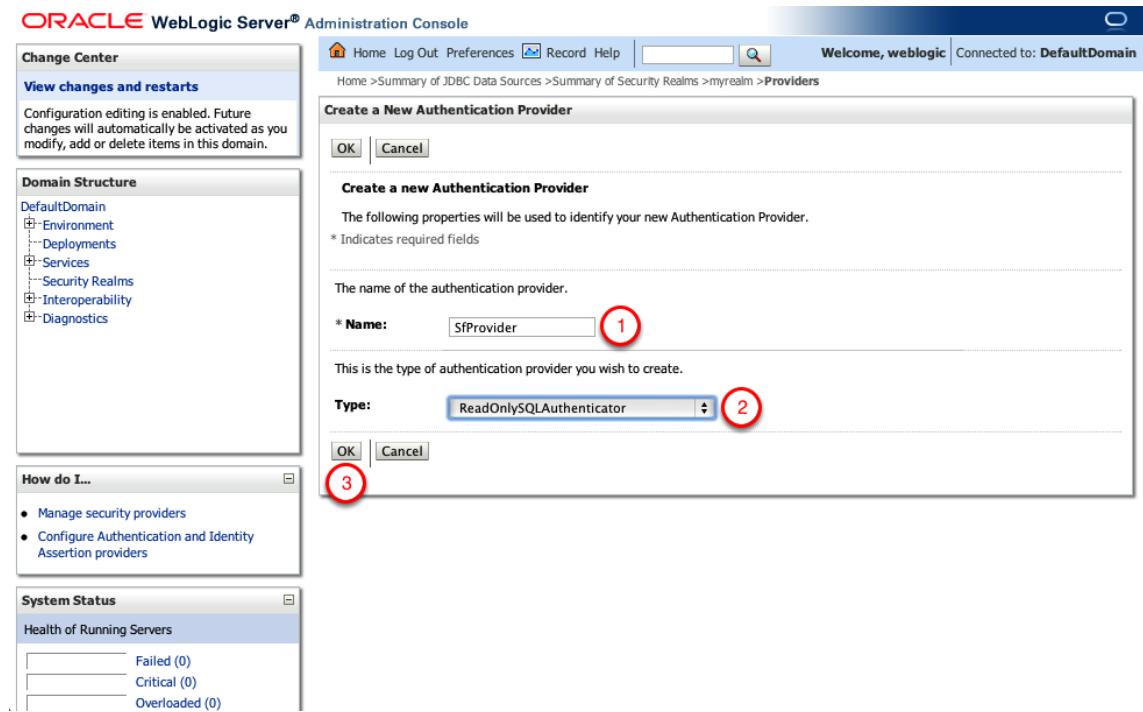
The central content area is titled "Settings for myrealm" and has tabs for Configuration, Users and Groups, Roles and Policies, Credential Mappings, **Providers**, and Migration. The "Providers" tab is highlighted and circled with a red number "1". Below the tabs is a descriptive text about authentication providers.

A "Customize this table" link leads to the "Authentication Providers" table. The table has columns for Name, Description, and Version. It lists two entries: "DefaultAuthenticator" (WebLogic Authentication Provider, Version 1.0) and "DefaultIdentityAsserter" (WebLogic Identity Assertion provider, Version 1.0). A "New" button is located at the top of the table, circled with a red number "2".

The bottom right corner of the table area shows a pagination message: "Showing 1 to 2 of 2 Previous | Next".

1. Open Providers tab
2. Click New

Configure new Authentication Provider



1. Set Name to SfProvider
2. Set Type to ReadOnlySQLAuthenticator
3. Click OK

Open SfProvider configuration

The screenshot shows the Oracle WebLogic Server Administration Console interface. The top navigation bar includes links for Home, Log Out, Preferences, Record, Help, and a search bar. The main title is "Welcome, weblogic" and it indicates "Connected to: DefaultDomain". The current path is "Home > Summary of JDBC Data Sources > Summary of Security Realms > myrealm > Providers".

Change Center panel:

- View changes and restarts:** Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Domain Structure panel:

- DefaultDomain
 - + Environment
 - Deployments
 - + Services
 - + Security Realms
 - + Interoperability
 - + Diagnostics

How do I... panel:

- Configure Authentication and Identity Assertion providers
- Configure the Password Validation provider
- Manage security providers
- Set the JAAS control flag
- Re-order Authentication providers

System Status panel:

- Health of Running Servers

Settings for myrealm panel:

- Providers tab is selected.
- Authentication sub-tab is selected.
- Buttons: New, Delete, Reorder.
- Text: An Authentication provider allows WebLogic Server to establish trust by validating a user. You must have one Authentication provider in a security realm, and you can configure multiple Authentication providers in a security realm. Different types of Authentication providers are designed to access different data stores, such as LDAP servers or DBMS. You can also configure a Realm Adapter Authentication provider that allows you to work with users and groups from previous releases of WebLogic Server.
- Customize this table link.

Authentication Providers table:

<input type="checkbox"/>	Name	Description	Version
<input type="checkbox"/>	DefaultAuthenticator	WebLogic Authentication Provider	1.0
<input type="checkbox"/>	DefaultIdentityAssertion	WebLogic Identity Assertion provider	1.0
<input type="checkbox"/>	SfProvider 1	Provider that performs DBMS authentication	1.0

Buttons: New, Delete, Reorder.

- Click on SfProvider

SfProvider Control Flag

The screenshot shows the Oracle WebLogic Server Administration Console interface. The left sidebar includes 'Change Center' with 'View changes and restarts' information, 'Domain Structure' listing 'DefaultDomain' with sub-nodes like 'Environment', 'Services', and 'Security Realms', and a 'How do I...' section with links for authentication, JAAS control flags, and security providers. The main content area is titled 'Settings for SfProvider' under 'Configuration' > 'Common'. It displays basic provider information and a 'Control Flag' dropdown menu. The 'Control Flag' dropdown menu has four options: 'REQUIRED', 'REQUISITE', 'SUFFICIENT' (which is highlighted with a red circle labeled '1'), and 'OPTIONAL'. Below the dropdown is a 'Save' button, which is also highlighted with a red circle labeled '2'.

1. Set Control Flag to: SUFFICIENT
2. Click Save

SfProvider specifics

The screenshot shows the Oracle WebLogic Server Administration Console. The left sidebar includes a 'Change Center' section with a message about configuration editing, a 'Domain Structure' tree, a 'How do I...' section with links for authentication providers and security providers, and a 'System Status' section showing server health. The main content area is titled 'Settings for SfProvider' under the 'Provider Specific' tab. Step 1 is highlighted with a red circle around the checked checkbox for 'Plaintext Passwords Enabled'. Step 2 is highlighted with a red circle around the 'Data Source Name' input field containing 'jdbc/sfDS'. Step 3 is highlighted with a red circle around the 'Save' button. Step 4 is highlighted with a red circle around the 'Providers >SfProvider' link in the breadcrumb navigation bar at the top right.

1. Enable Plaintext Passwords
2. Enter "jdbc/sfDS" as Data Source Name
3. Click Save
4. Open Providers page in the breadcrumbs

Open DefaultAuthenticator configuration

The screenshot shows the Oracle WebLogic Server Administration Console interface. The top navigation bar includes links for Home, Log Out, Preferences, Record, Help, and a search bar. The main title is "Welcome, weblogic" and it indicates "Connected to: DefaultDomain". Below the title, the path is "Home >Summary of JDBC Data Sources >Summary of Security Realms >myrealm >Providers >SfProvider >Providers". The left sidebar has sections for Change Center (with a note about configuration editing), Domain Structure (listing Environment, Deployments, Services, Security Realms, Interoperability, and Diagnostics), How do I... (with a list of tasks like Configure Authentication and Identity Assertion providers, Configure the Password Validation provider, Manage security providers, Set the JAAS control flag, and Re-order Authentication providers), and System Status (Health of Running Servers). The main content area is titled "Settings for myrealm" and focuses on "Providers". Under "Authentication", the "DefaultAuthenticator" provider is selected and highlighted with a red circle containing the number "1". A tooltip for "DefaultAuthenticator" states: "An Authentication provider allows Weblogic Server to establish trust by validating a user. You must have one Authentication provider in a security realm, and you can configure multiple Authentication providers in a security realm. Different types of Authentication providers are designed to access different data stores, such as LDAP servers or DBMS. You can also configure a Realm Adapter Authentication provider that allows you to work with users and groups from previous releases of WebLogic Server." Below this, there is a table titled "Authentication Providers" showing three entries:

<input type="checkbox"/>	Name	Description	Version
<input type="checkbox"/>	DefaultAuthenticator 1	WebLogic Authentication Provider	1.0
<input type="checkbox"/>	DefaultIdentityAssertioner	WebLogic Identity Assertion provider	1.0
<input type="checkbox"/>	SfProvider	Provider that performs DBMS authentication	1.0

1. Click DefaultAuthenticator

Change DefaultAuthenticator's Control Flag

The screenshot shows the Oracle WebLogic Server Administration Console. In the center, there is a 'Settings for DefaultAuthenticator' page under the 'Configuration' tab. On the right, there is a 'Control Flag' dropdown menu with four options: REQUIRED, REQUISITE, SUFFICIENT, and OPTIONAL. The 'SUFFICIENT' option is highlighted with a red circle and labeled '1'. Below the dropdown is a 'Save' button, which is also circled in red and labeled '2'. To the left of the main content area, there is a 'Domain Structure' tree and a 'How do I...' help section.

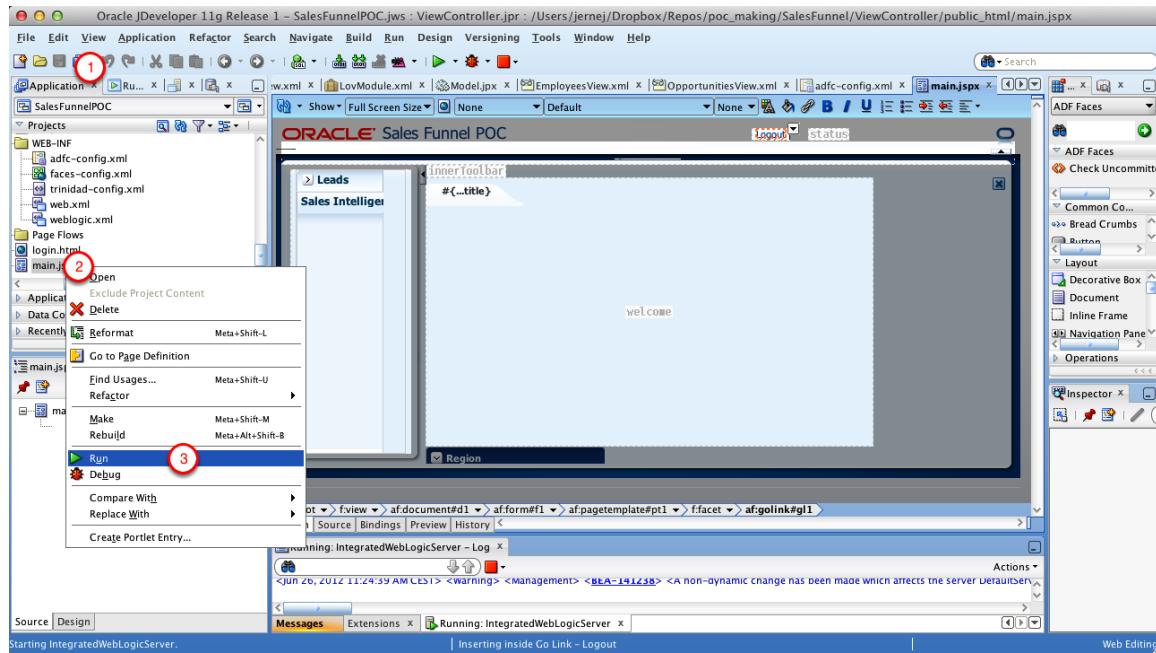
1. Set Control Flag to SUFFICIENT
2. Click Save

Stop WebLogic Server

The screenshot shows the Oracle JDeveloper 11g interface. The central workspace displays the 'IntegratedWebLogicServer' application. The toolbar at the top has a red circle labeled '1' over the 'Stop' icon. The bottom status bar shows the message 'Starting IntegratedWebLogicServer.' The interface includes various toolbars, menus, and panels typical of a Java development environment.

1. Click Stop Icon and stop Integrated WebLogic server

Run main.jspx



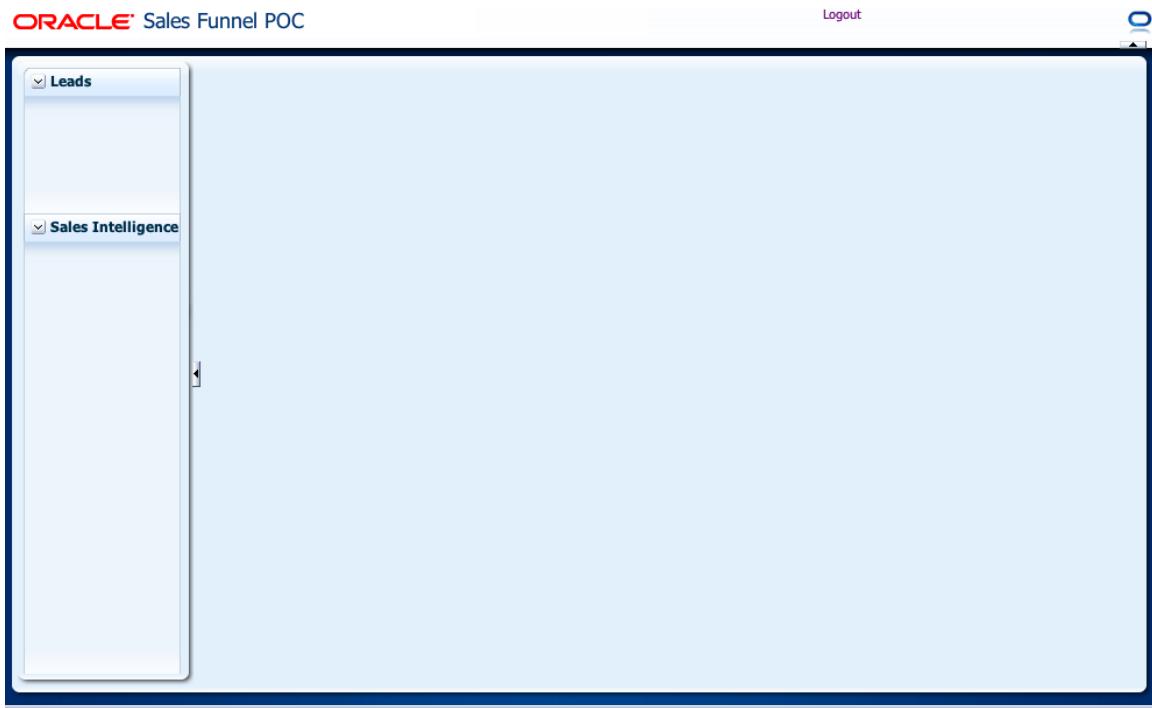
1. Open Application Navigator
2. Right-click main.jspx
3. Click Run in the popup menu

Test Login



1. Enter "demetris" username
2. Enter "password" password
3. Click Login

Main form loads

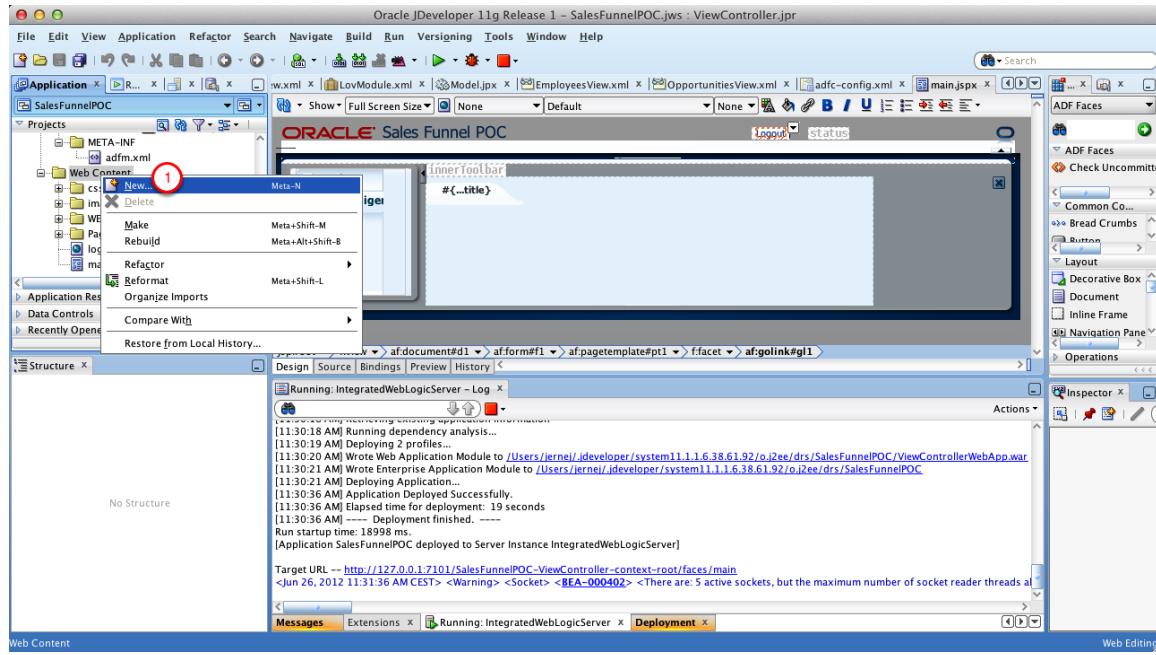


Main page loads

19. Page Fragment template

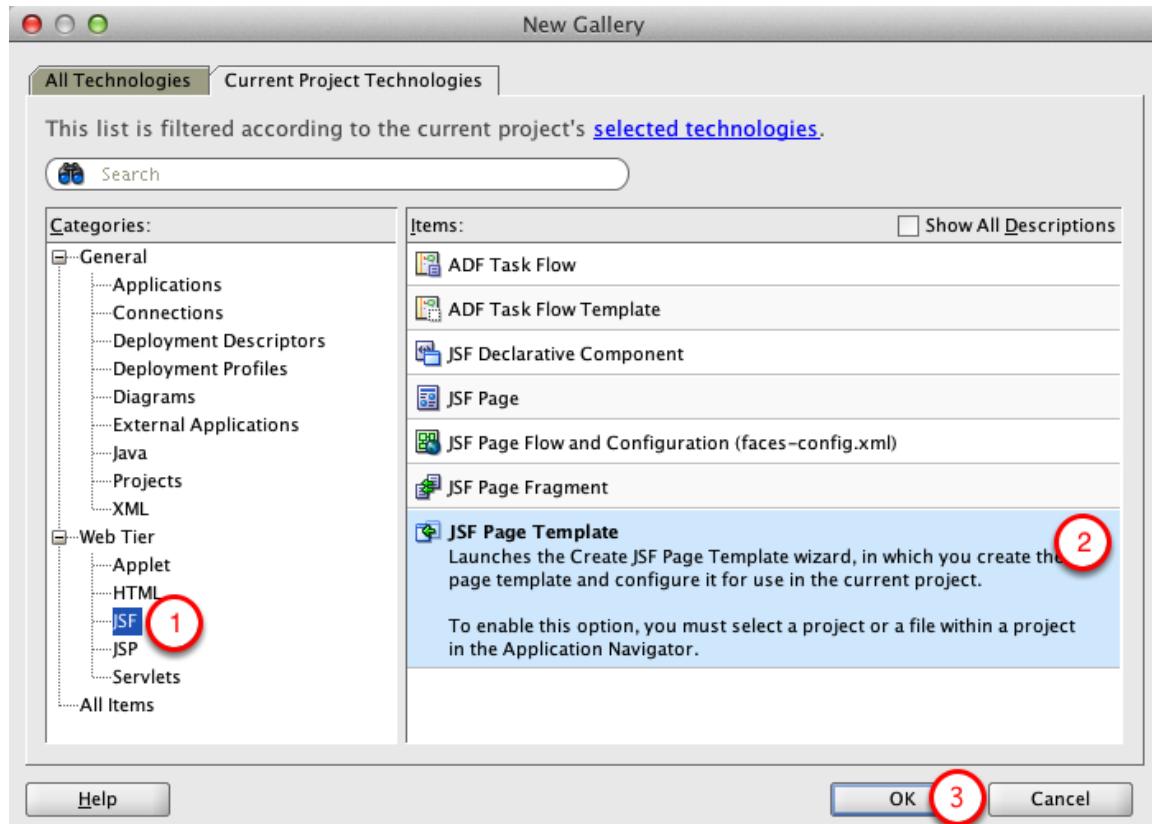
Since many pages will share similar layout, we'll create a template for them

Create New template



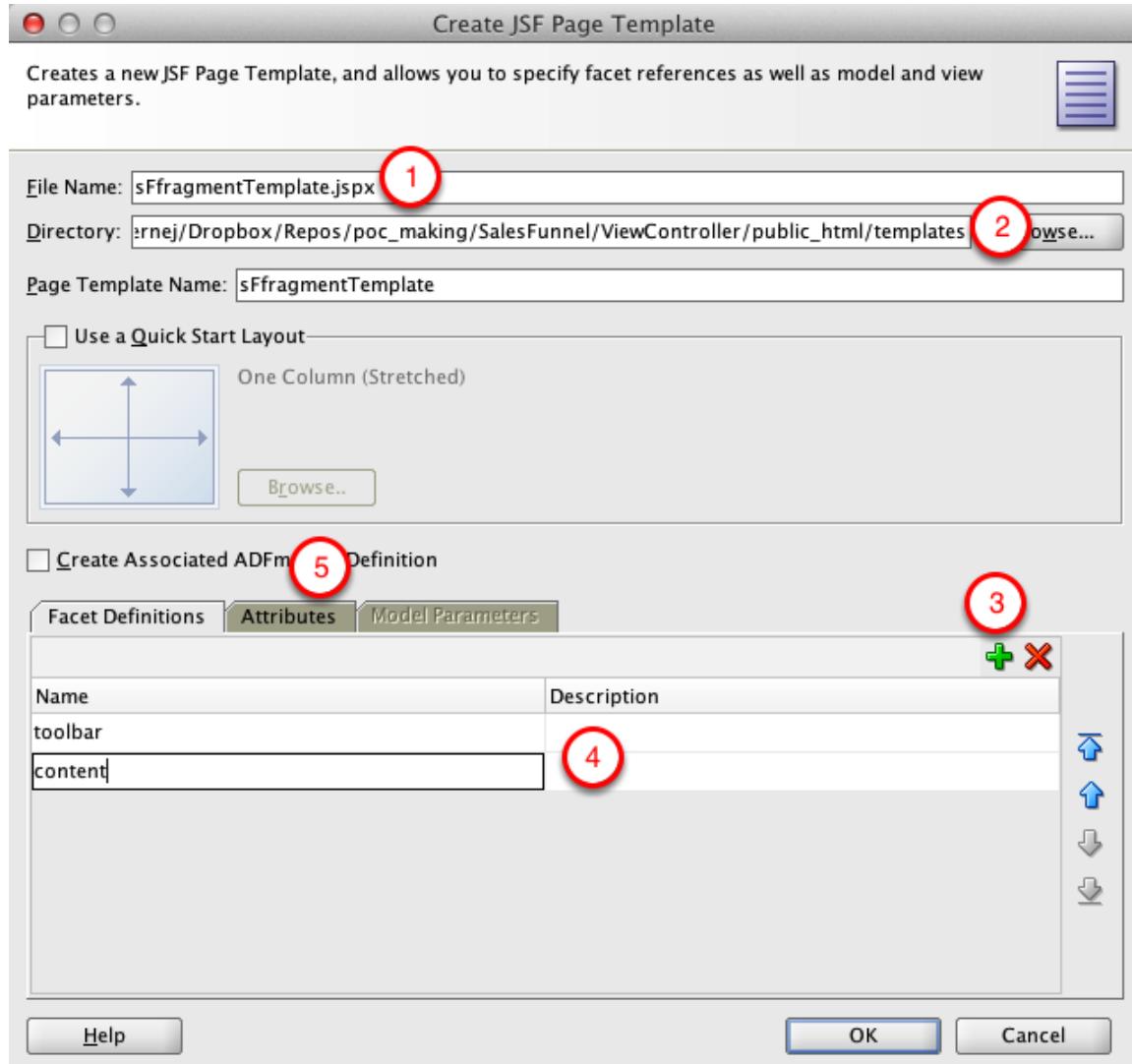
1. Right-click Web Content folder, select New in the popup menu

New Gallery



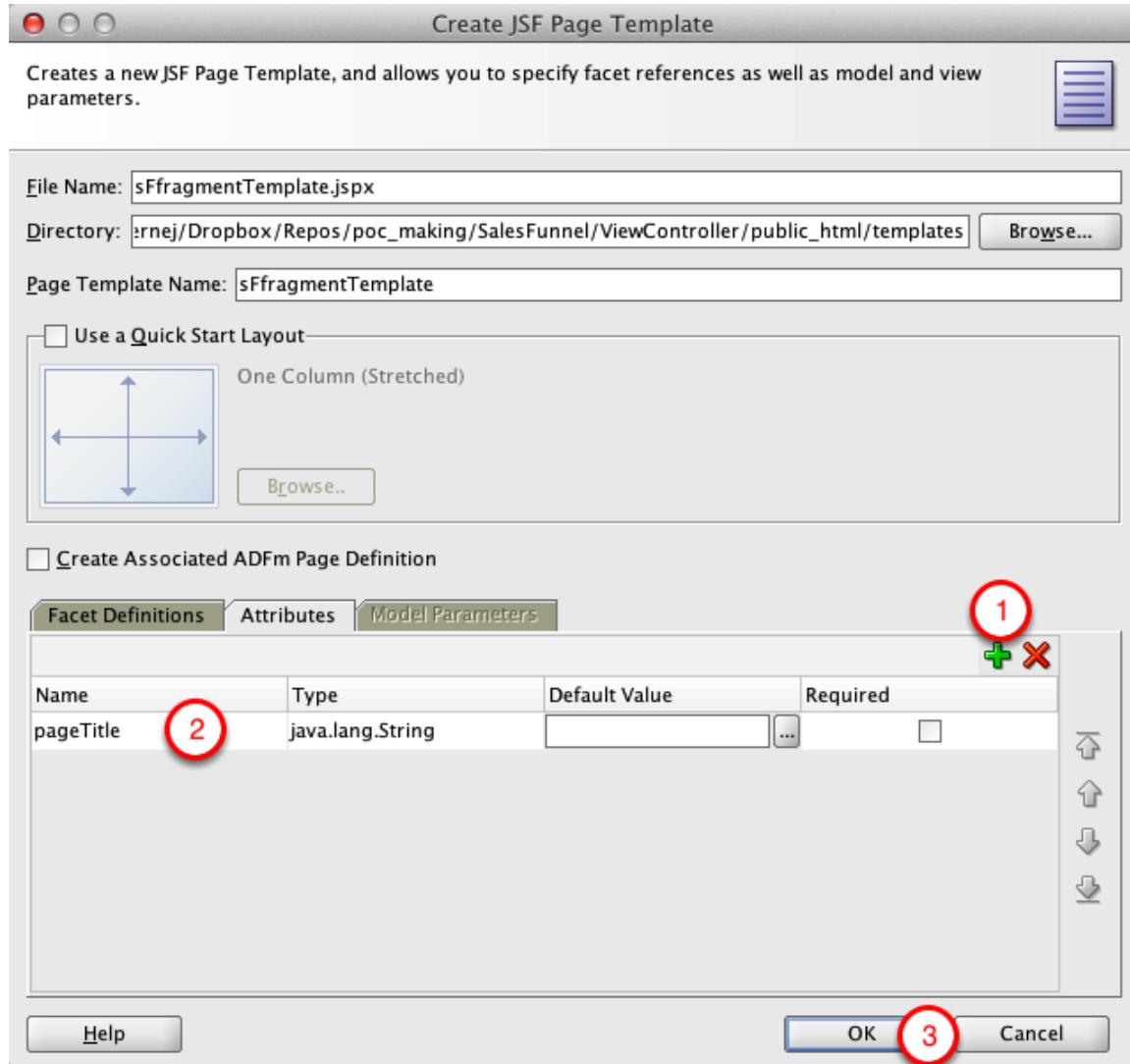
1. In Categories, select Web Tier->JSF
2. Select JSF Page Template
3. Click OK

Create JSF Page Template



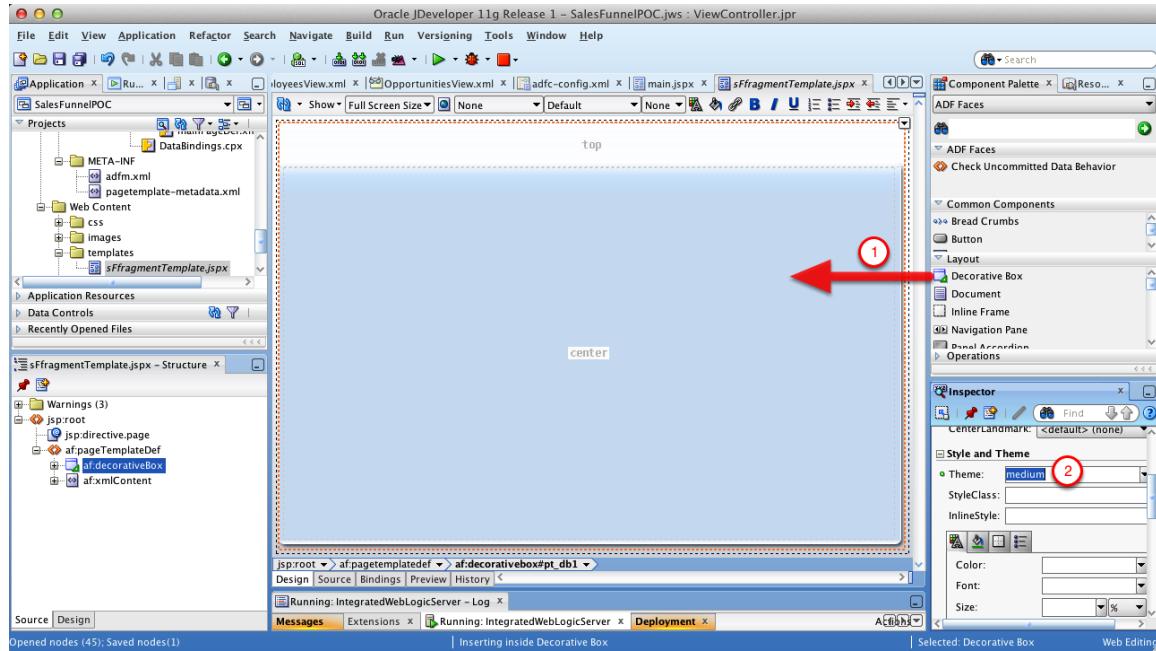
1. Set File name to sFFragmentTemplate.jspx
2. Append "/templates" to Directory
3. Click + in Facet Definitions tab
4. Add toolbar and content facets
5. Open Attributes tab

Create JSF Page Template



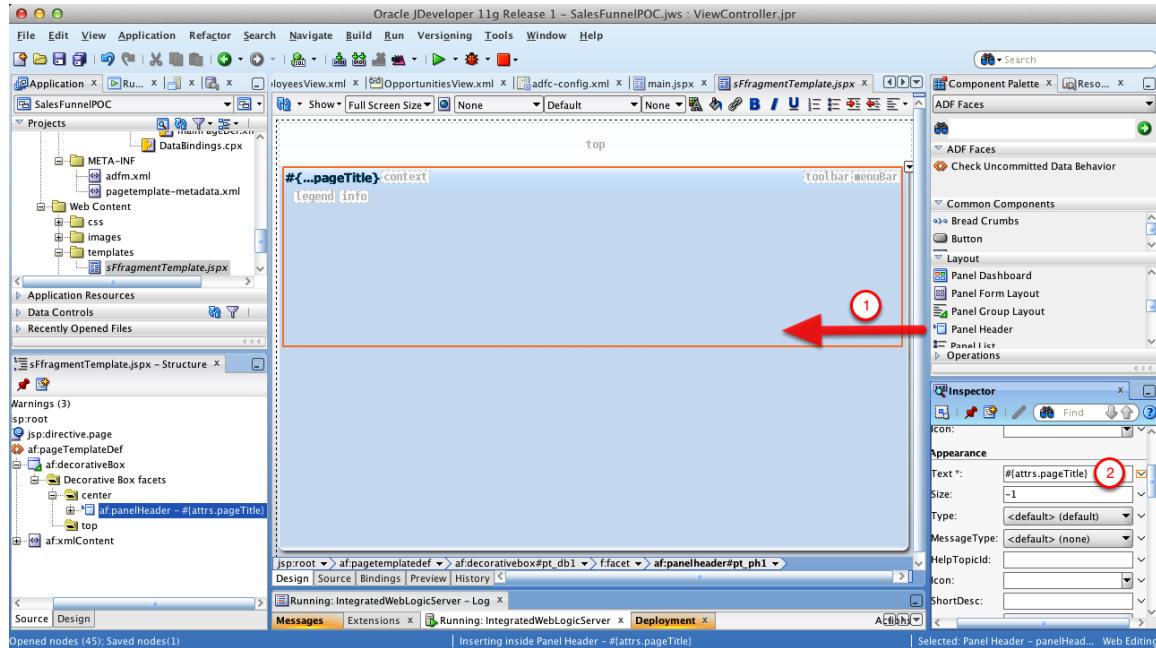
1. Click +
2. Add pageTitle attribute
3. Click OK

Add Decorative Box



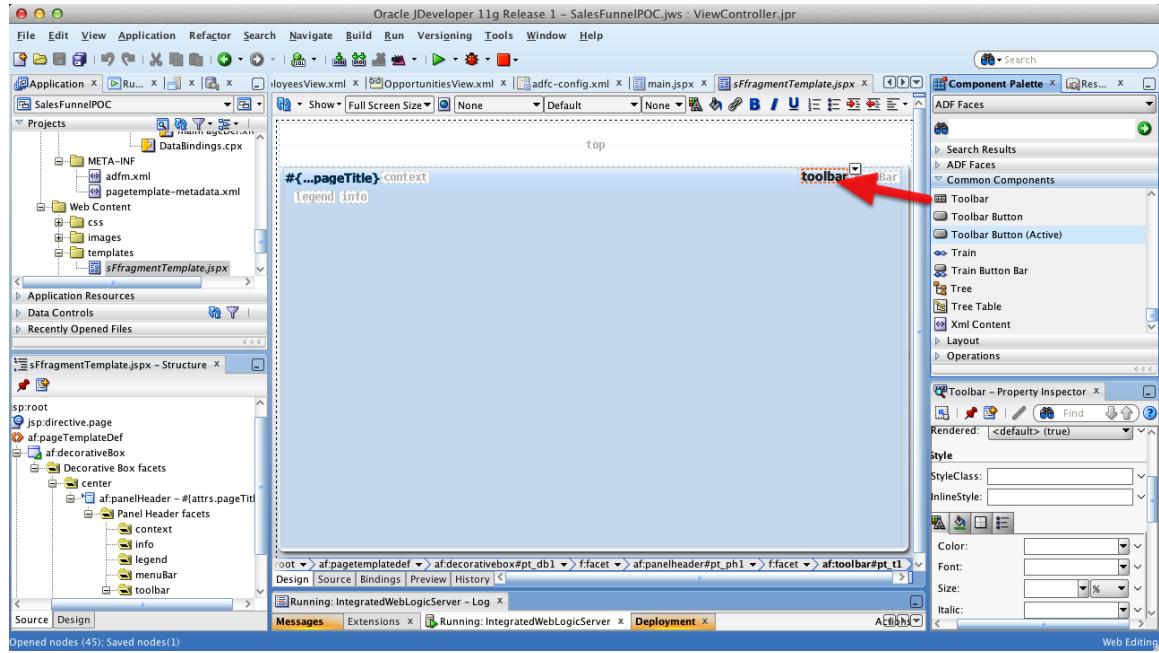
1. Drag and drop Decorative Box to the Design view
2. Set Theme property to medium

Add Panel Header



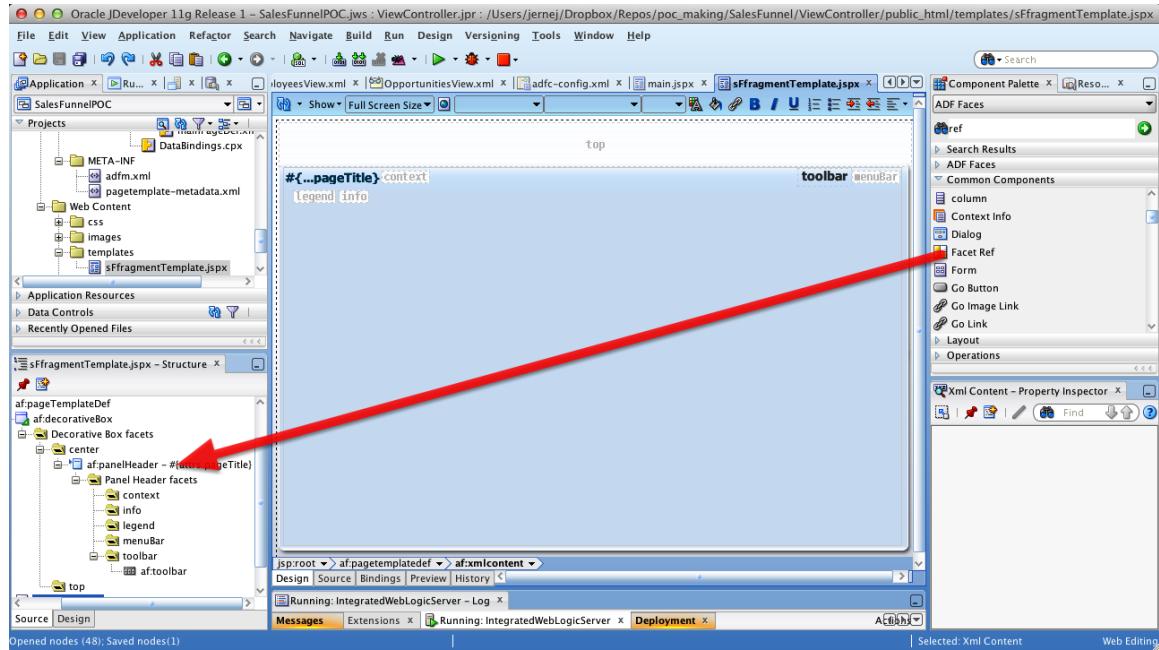
1. Drag and drop Panel Header to the center facet of the Decorative Box
2. Set Text property to "#{attrs.pageTitle}" (without quotes)

Add Toolbar



1. Drag and drop Toolbar inside toolbar facet of the Panel Header

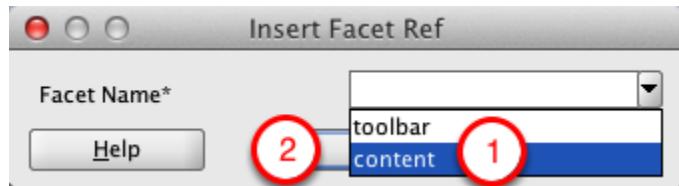
Add Content Facet Ref



1. Drag and drop Facet Ref component on top of panelHeader

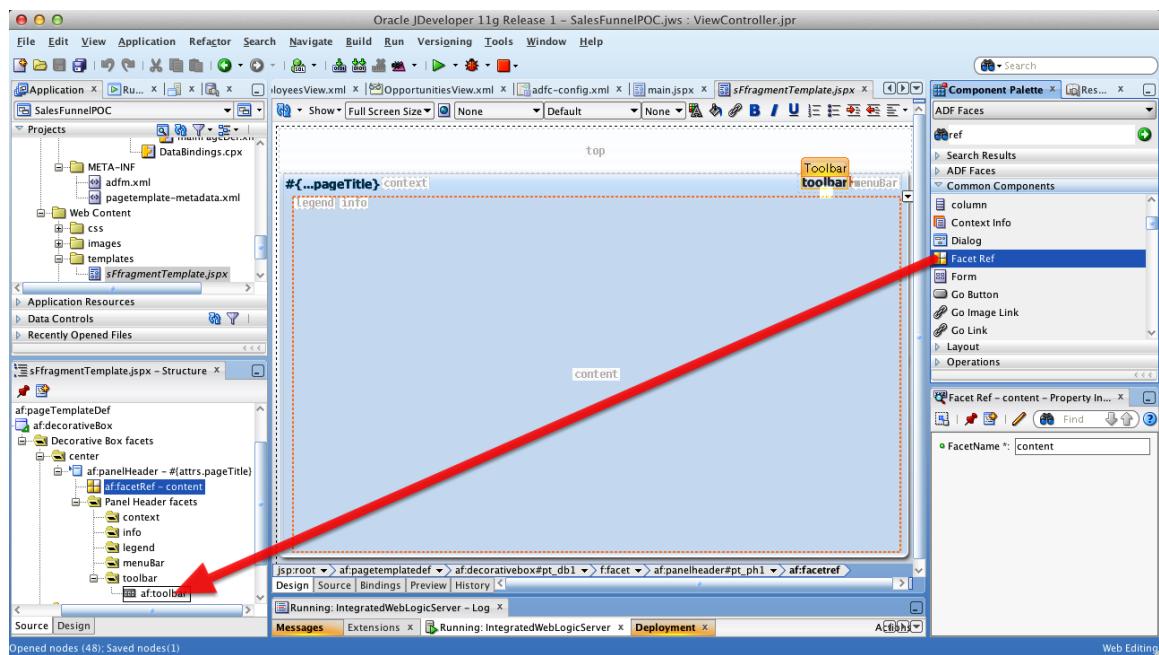
Insert Facet Ref

facet Definition in JD R2



1. Set Facet Name to content
2. Click OK

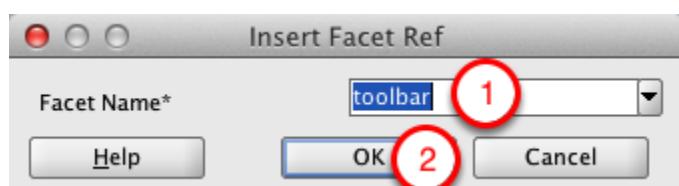
Add Toolbar Facet Ref



1. Drag and drop Facet Ref on top of af:toolbar

Insert Facet Ref

facet Definition in JD R2

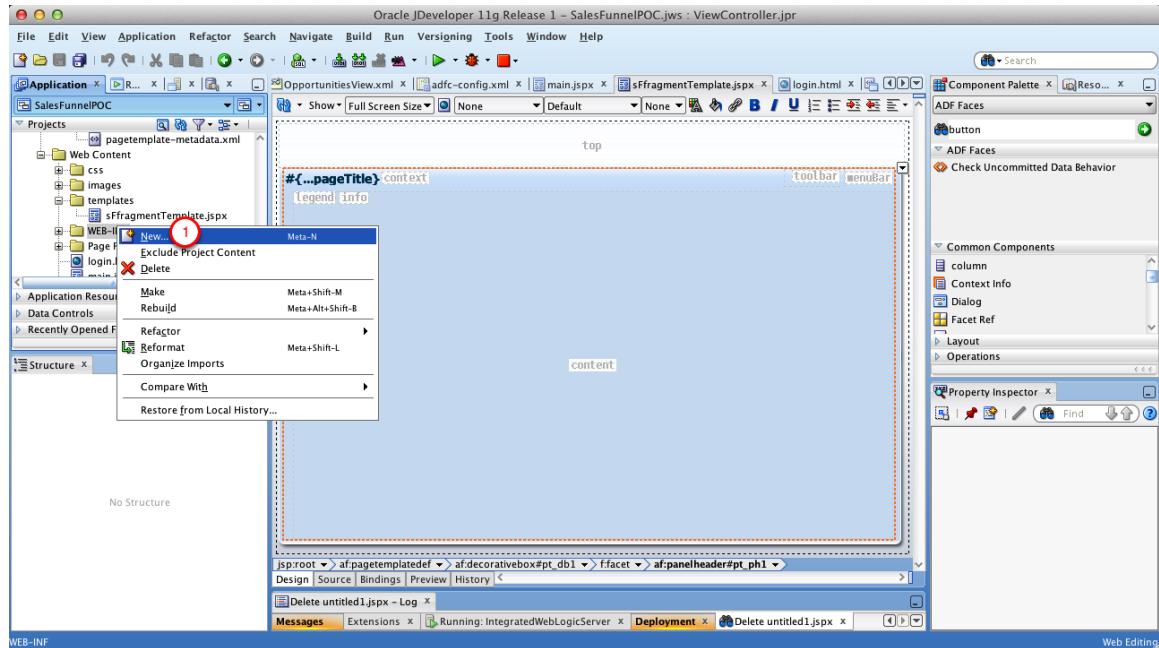


1. Set Facet Name to toolbar
2. Click OK

20. BTF Template

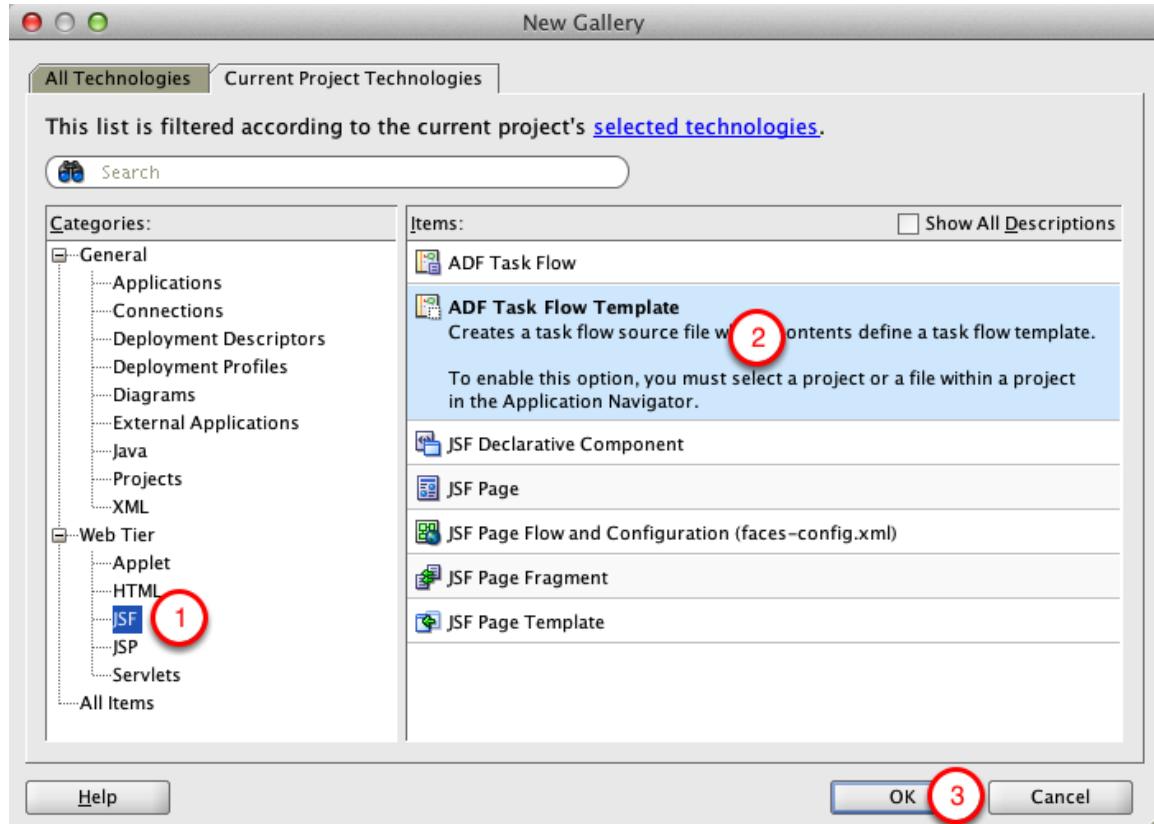
It is a good practice to use Bounded Task Flow templates to enable reuse, because any ADF bounded task flow based on a template has the same set of activities, control flows, input parameters, and managed bean definitions that the template contains. In our project we'll use an empty BTF template which we could extend later on if we needed to.
The most common use for BTF template would be exception handling.

Create New Bounded Task Flow Template



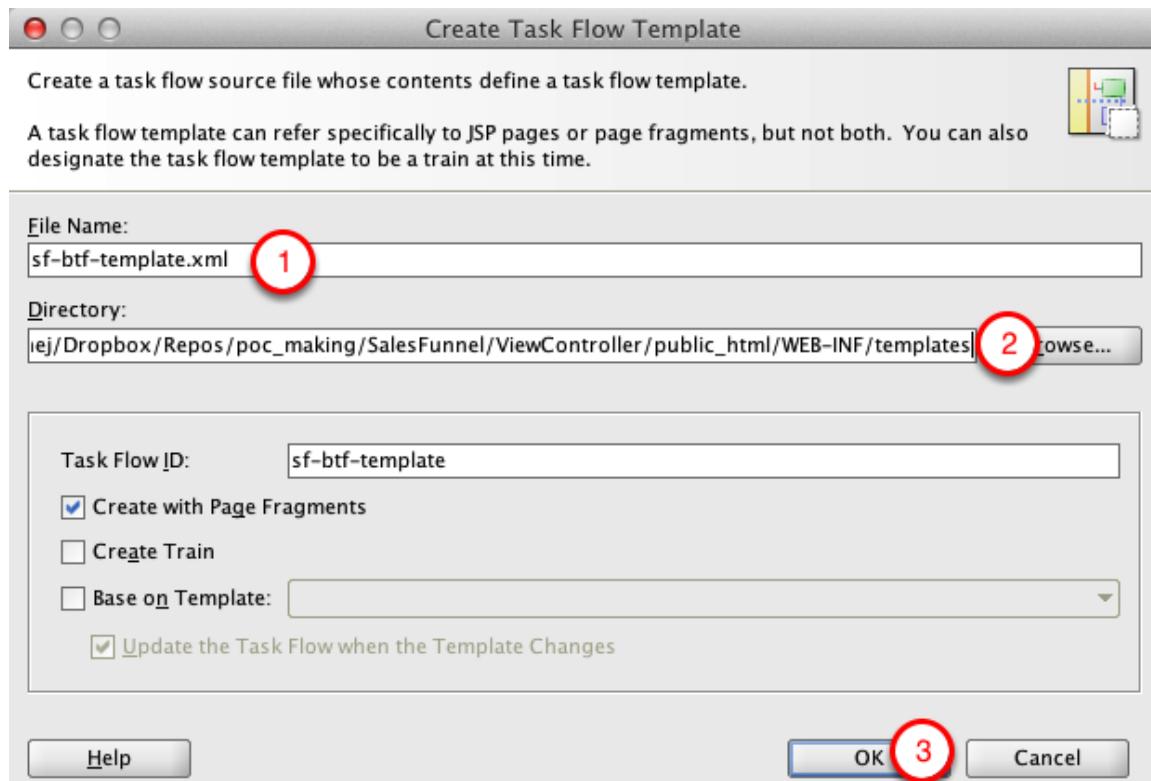
1. Right-click WEB-INF inside Web Content folder and click New in the popup menu

New Gallery



1. Select Web Tier->JSF category
2. Select ADF Task Flow Template
3. Click OK

Create Task Flow Template

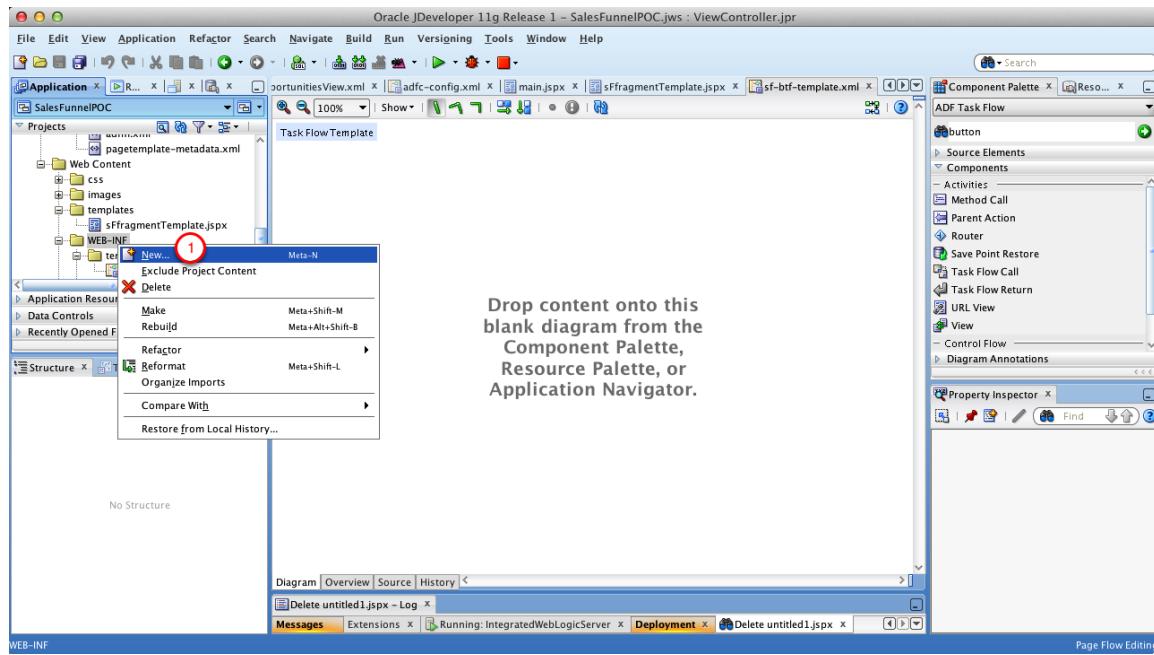


1. Set name to sf-btf-template.xml
2. Append Directory with "/templates"
3. Click OK

III. Implementing Core Use Cases

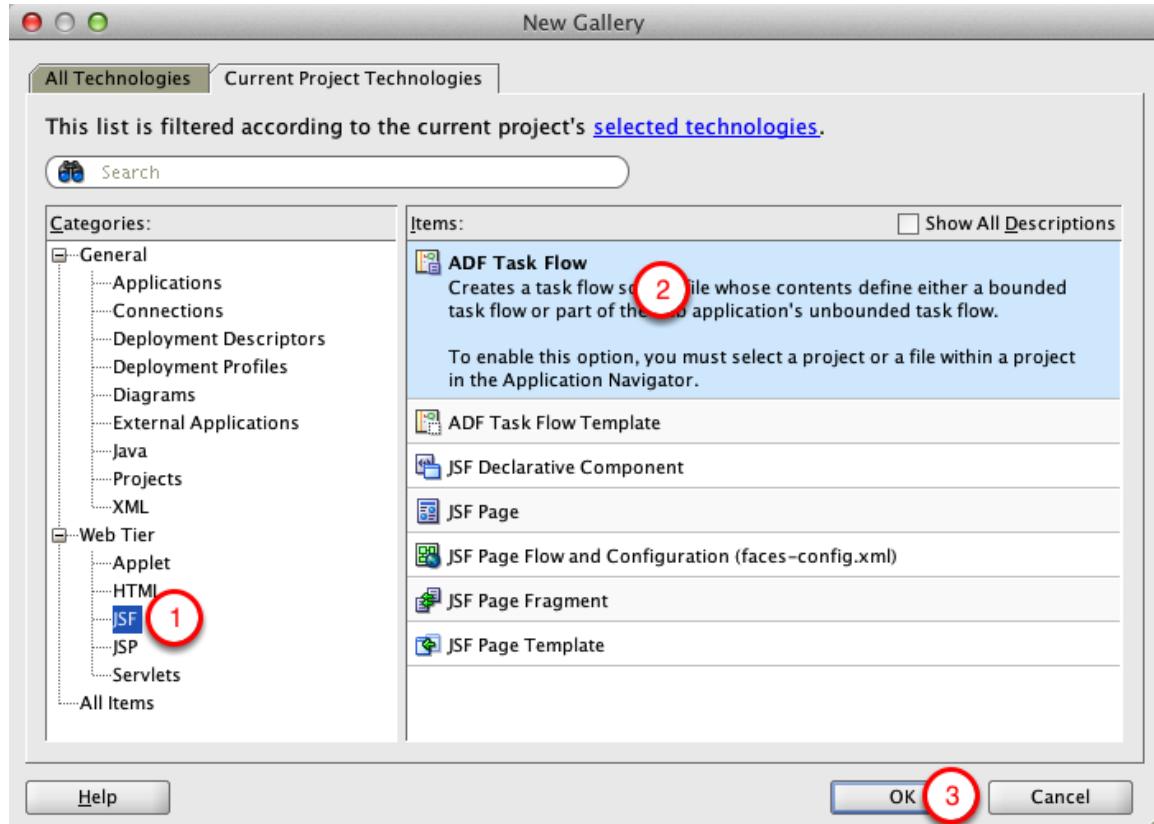
21. Lead Edit BTF

Create New Task Flow



1. Right-click WEB-INF and click New in the popup menu

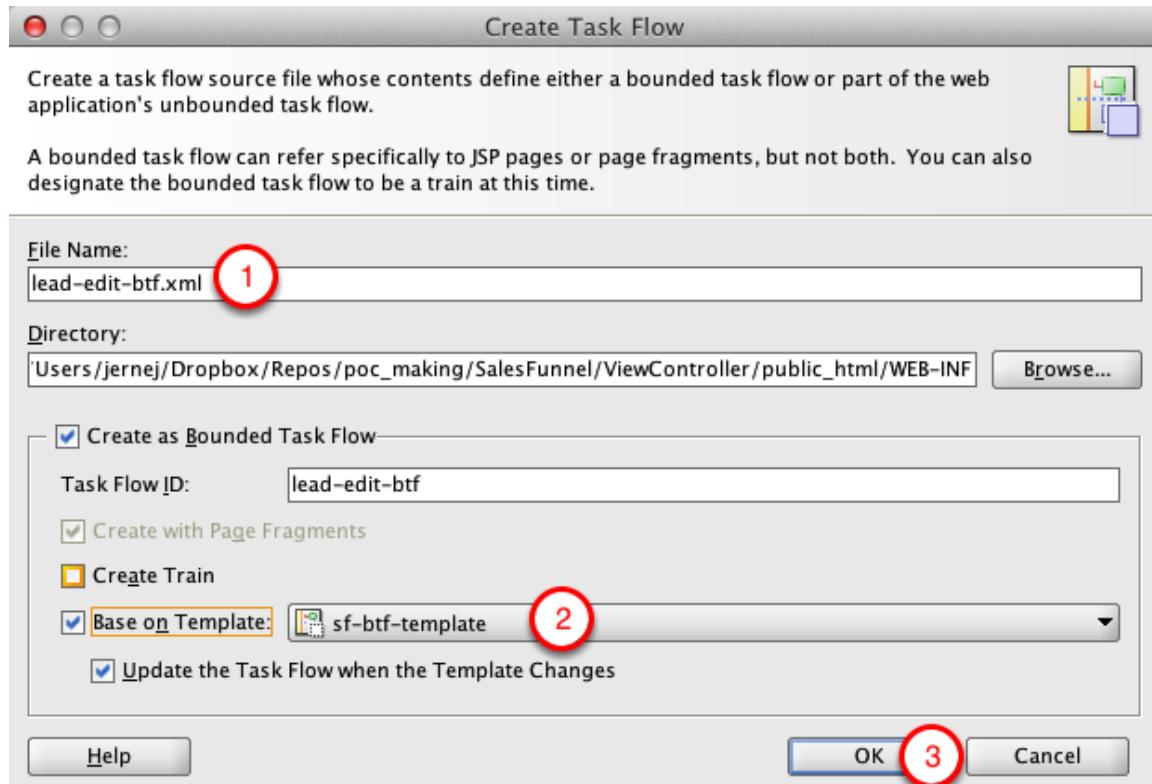
New Gallery



1. Select JSF category
2. Select ADF Task Flow*
3. Click OK

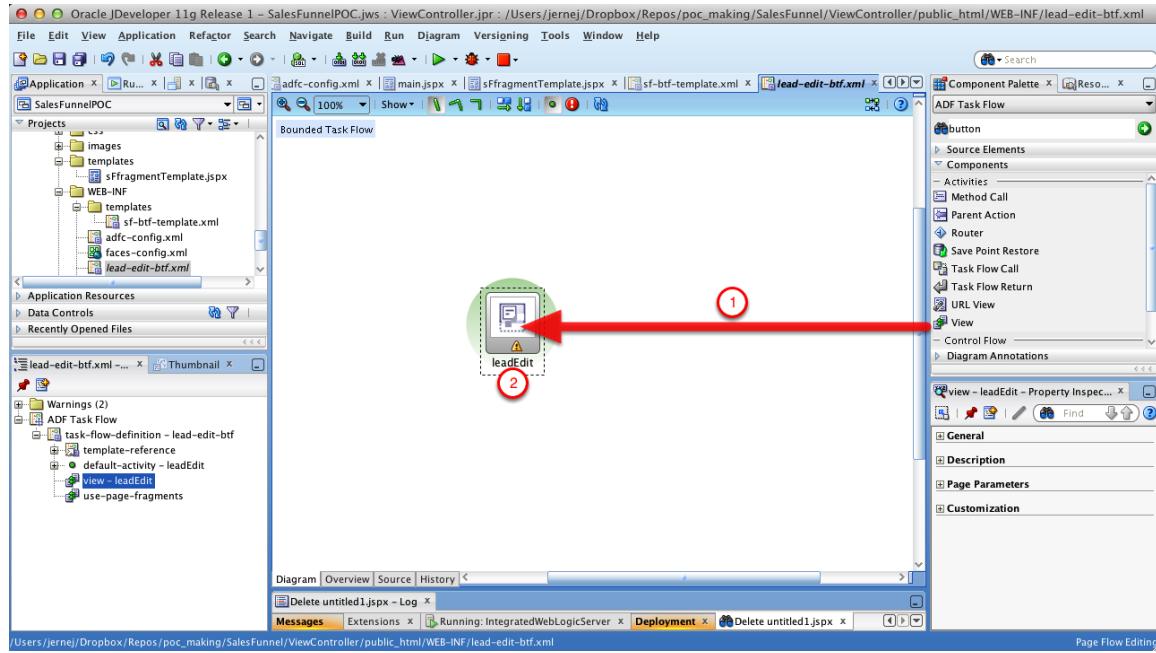
*Make sure you don't select ADF Task Flow Template by mistake

Create Task Flow



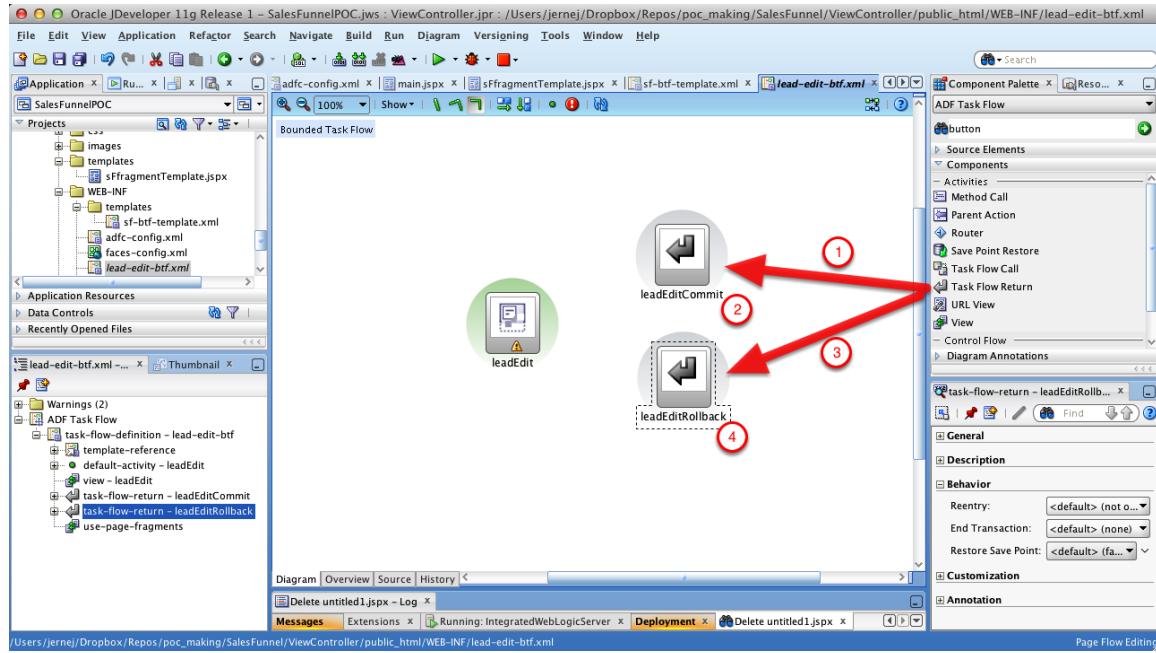
1. Set file name to lead-edit-btf.xml
2. Check Base on template and select sf-btf-template
3. Click OK

Add leadEdit View activity



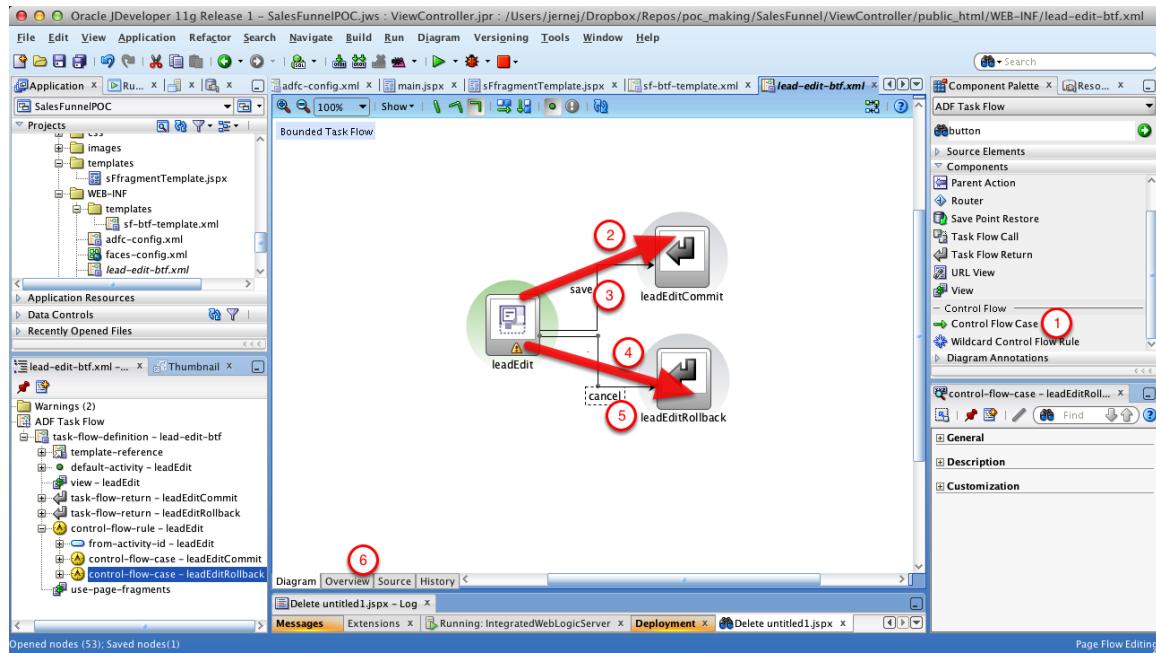
1. Drag and drop View activity to the diagram
2. Rename it to leadEdit

Add Task Flow Returns



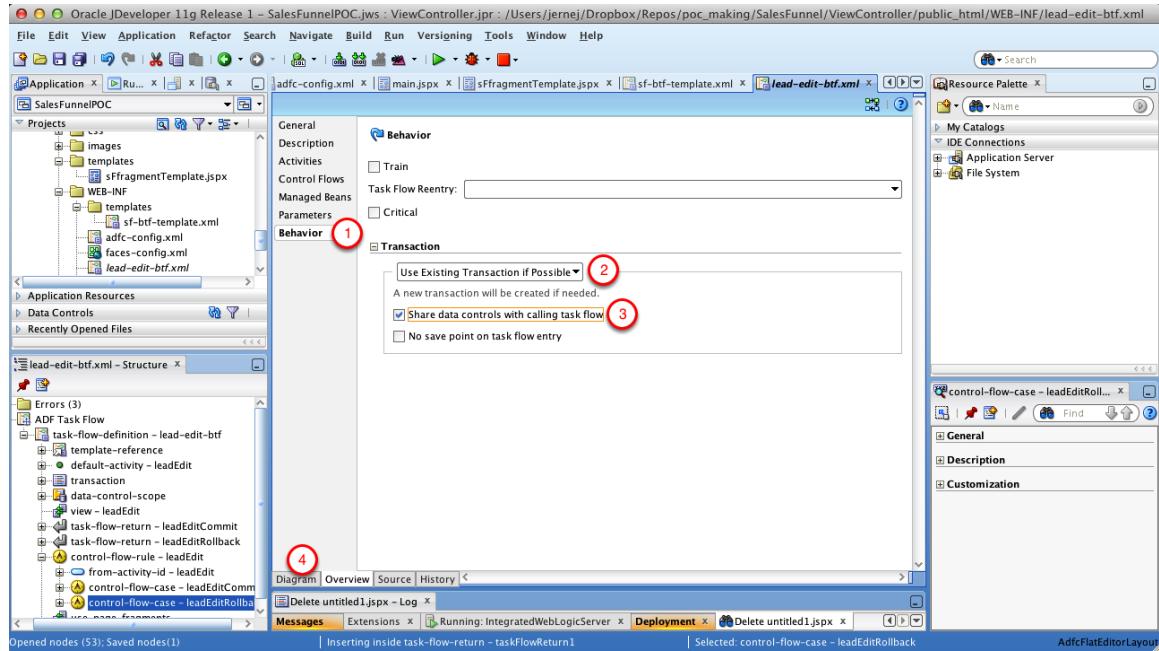
1. Drag and drop Task Flow Return activity to the diagram
2. Rename it to leadEditCommit
3. Drag and drop Task Flow Return activity to the diagram
4. Rename it to leadEditRollback

Connect leadEdit to Returns



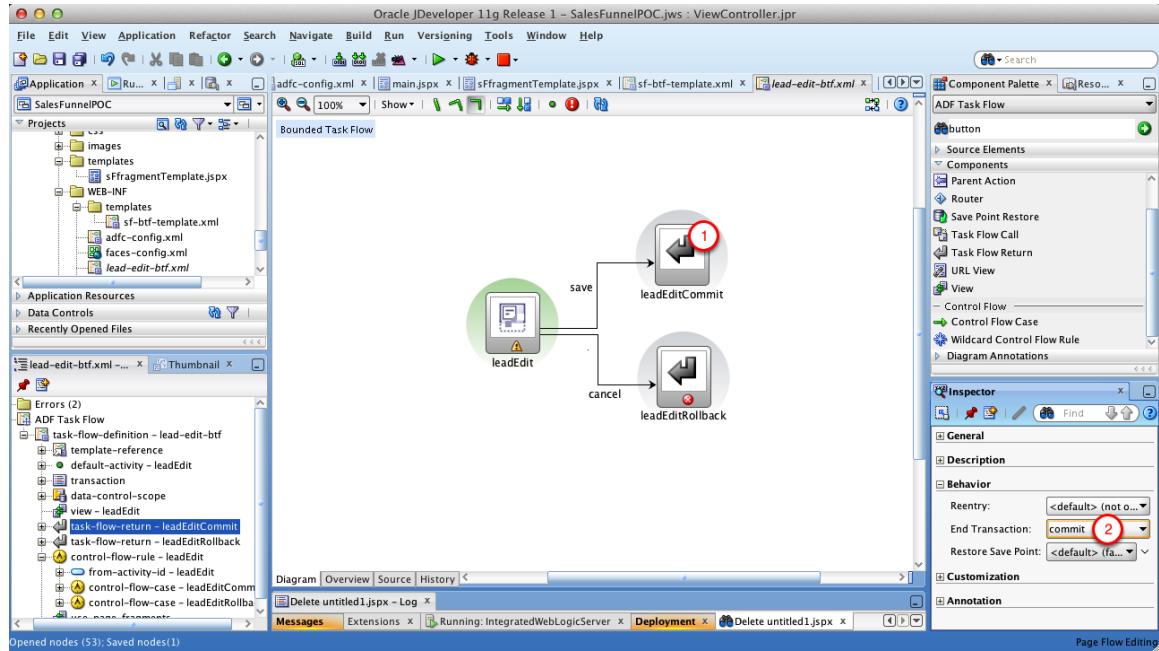
1. Click Control Flow Case in the Component Palette
2. Connect leadEdit to leadEditCommit
3. Rename connection to save
4. Connect leadEdit to leadEditRollback
5. Rename connection to cancel
6. Open Overview tab

Configure Task Flow Behavior



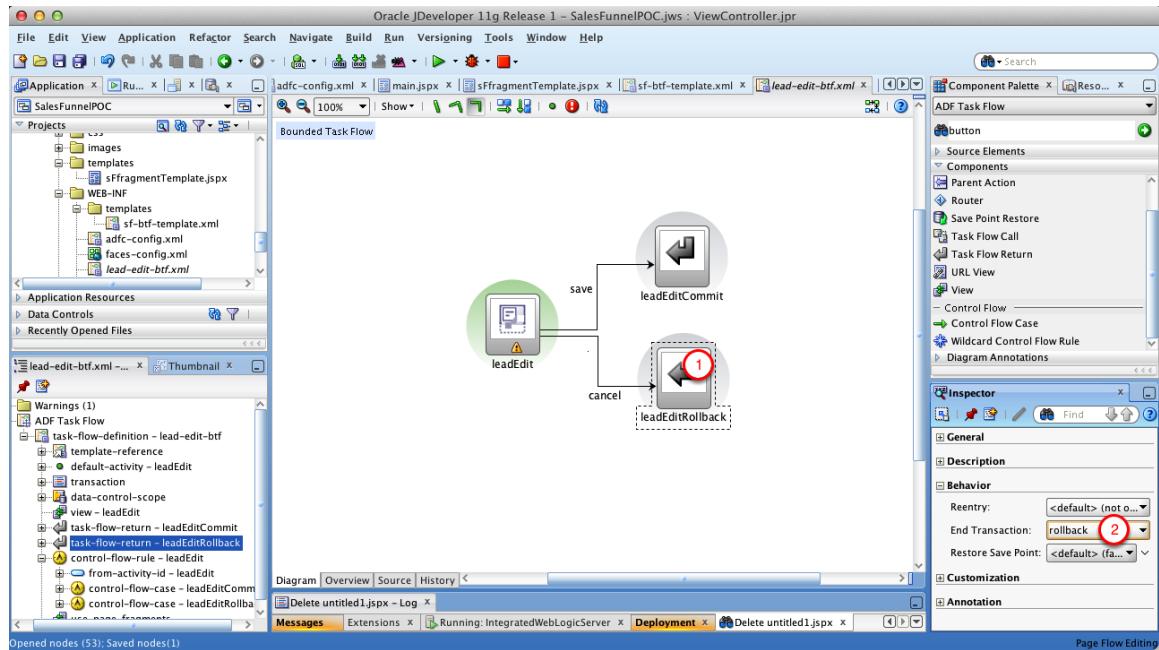
1. Open Behavior tab
2. Set Transaction option to "Use Existing Transaction if Possible"
3. Check Share data controls with calling task flow
4. Open Diagram tab

Configure leadEditCommit



1. Select leadEditCommit activity
2. Set End Transaction property to commit

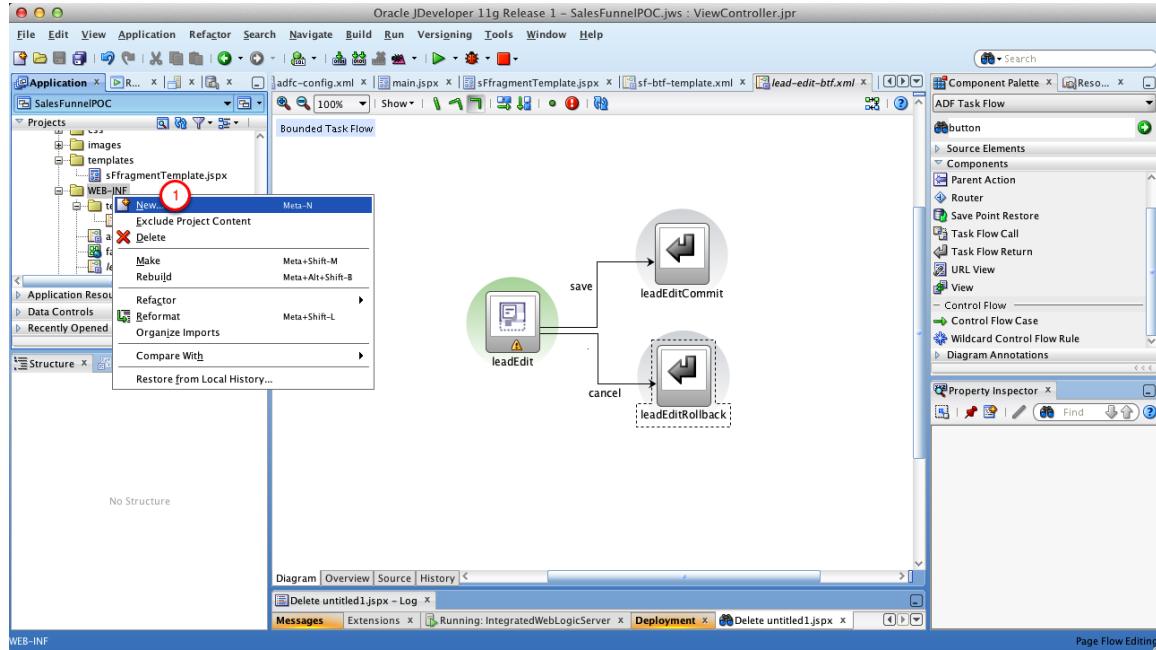
Configure leadEditRollback



1. Select leadEditRollback activity
2. Set End Transaction property to rollback

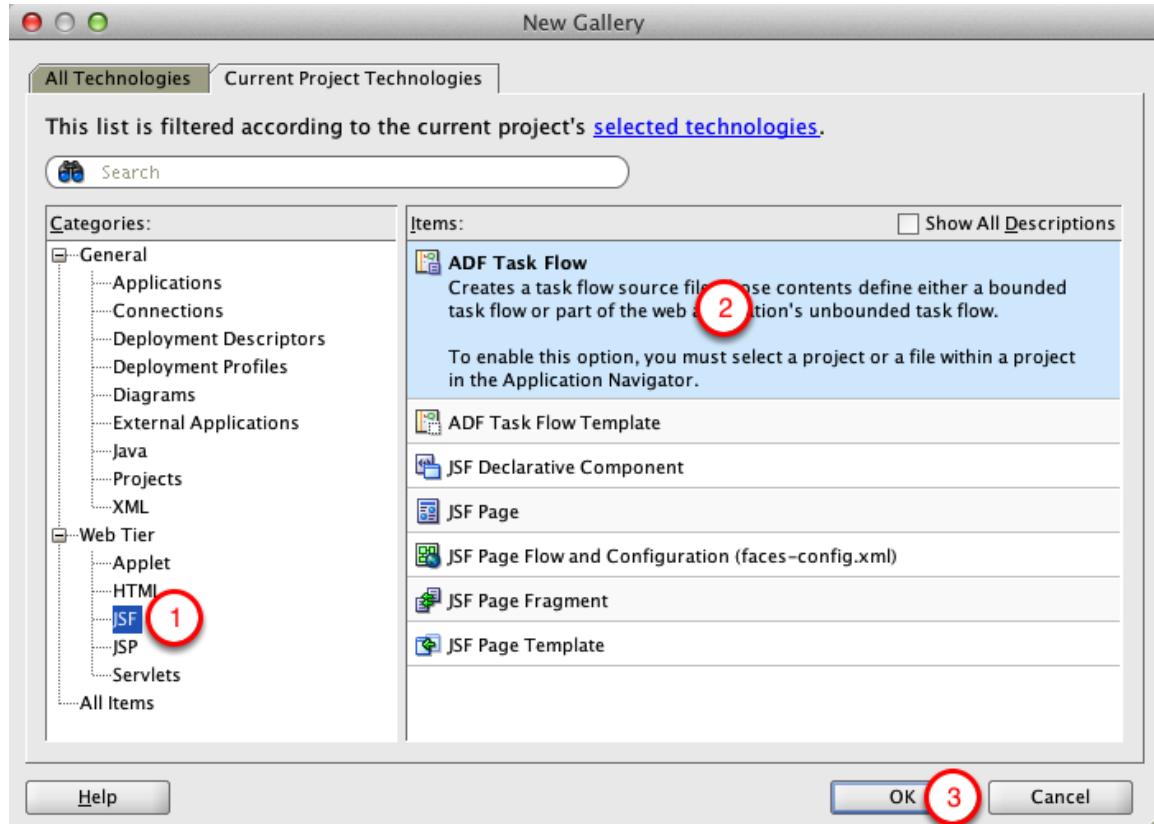
22. Lead Create BTF

Create New Task Flow



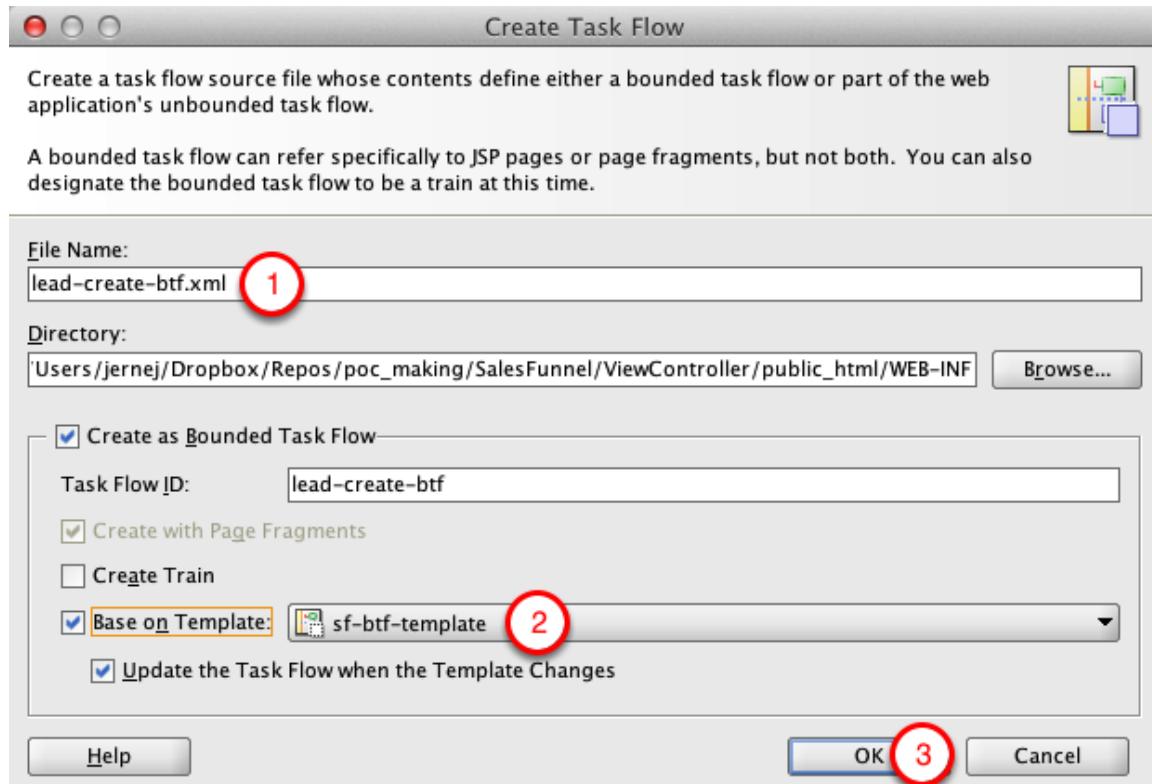
1. Right-click WEB-INF and select New in the popup menu

New Gallery



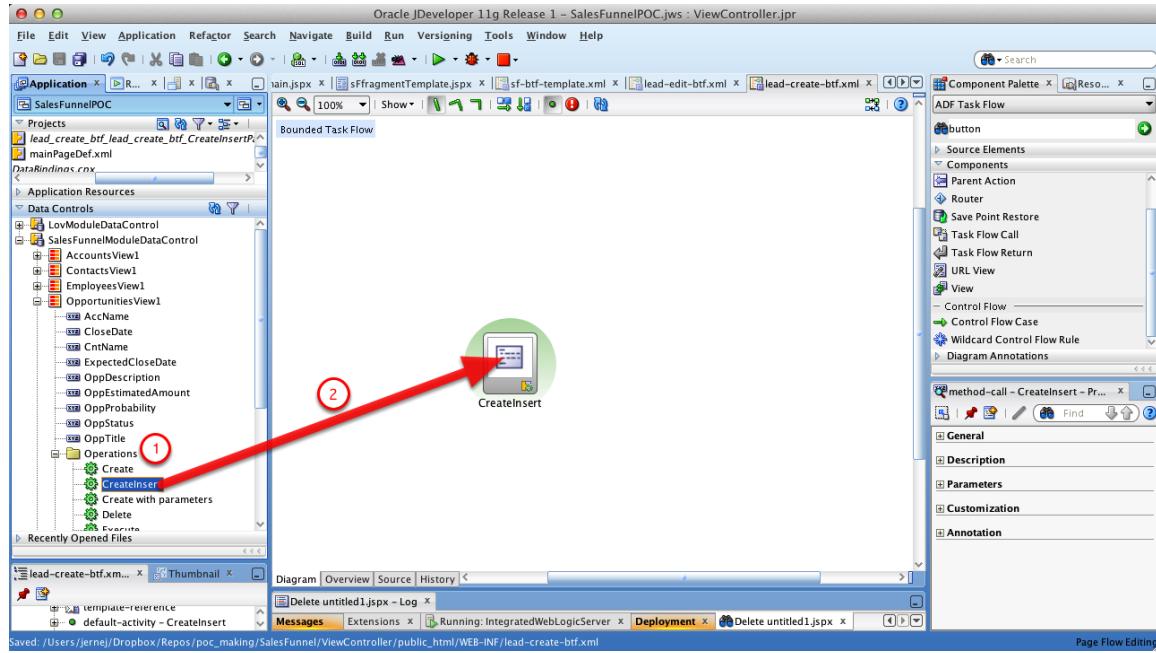
1. Select JSF category
2. Select ADF Task Flow
3. Click OK

Create Task Flow



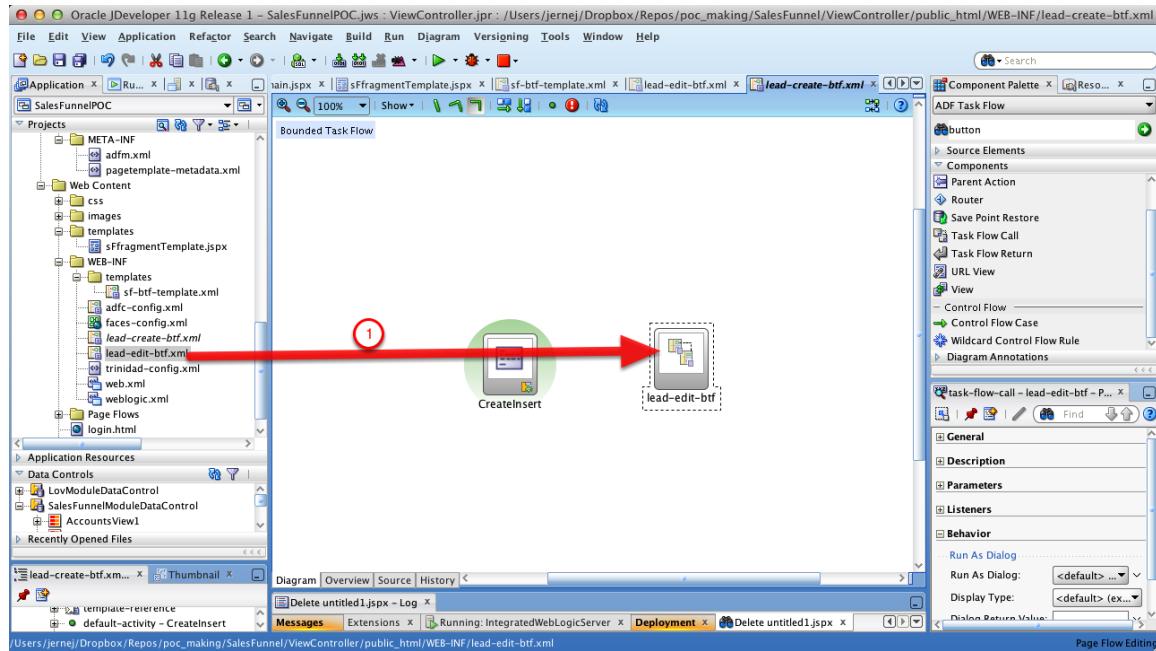
1. Set file name to lead-create-btf.xml
2. Check Base on template and select sf-btf-template
3. Click OK

Add Opportunities CreateInsert operation



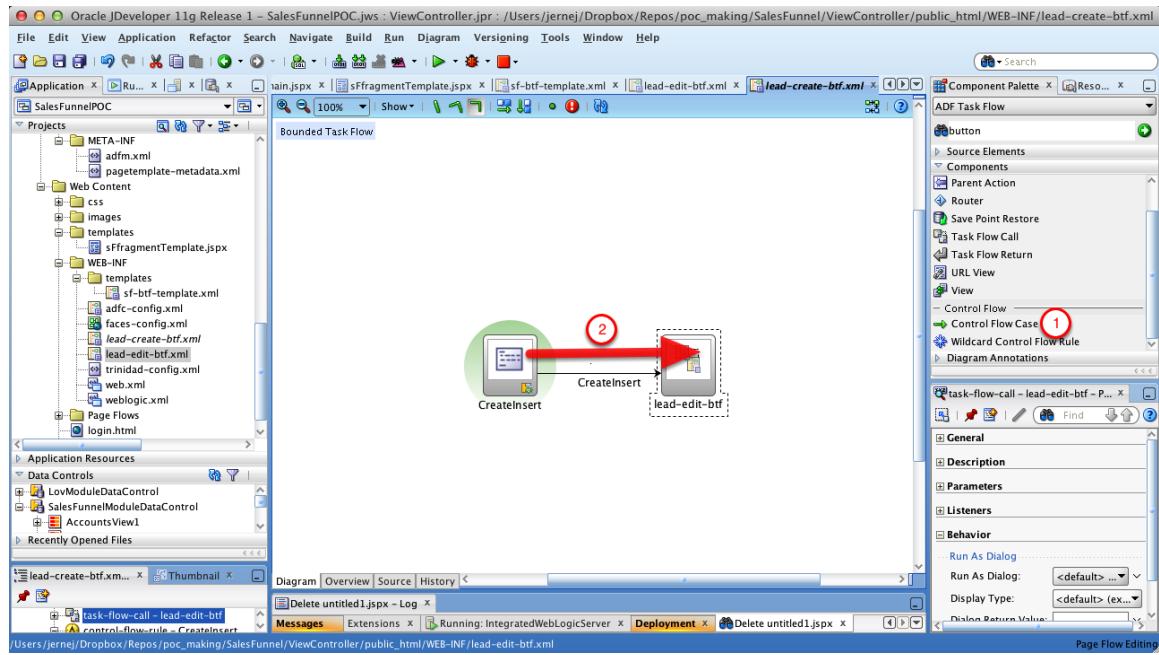
1. Expand Data Controls, locate OpportunitiesView1, expand Operations
2. Drag and drop CreateInsert operation to the diagram

Add lead-edit-btf



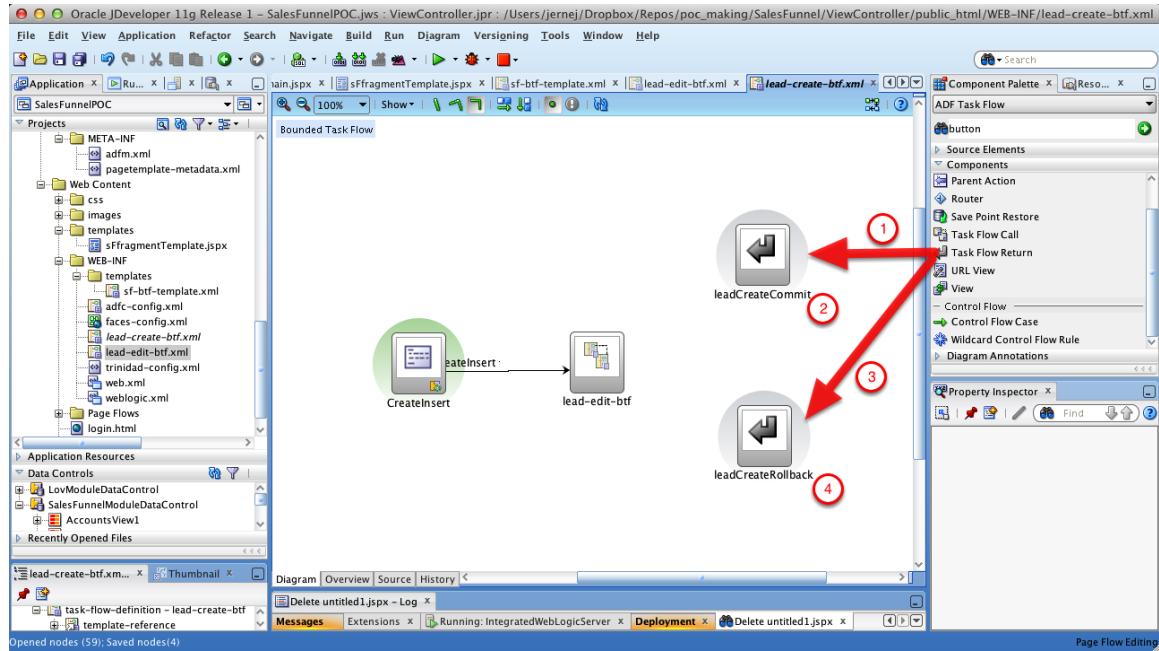
1. Drag and drop lead-edit-btf.xml to the diagram

Connect CreateInsert to lead-edit-btf



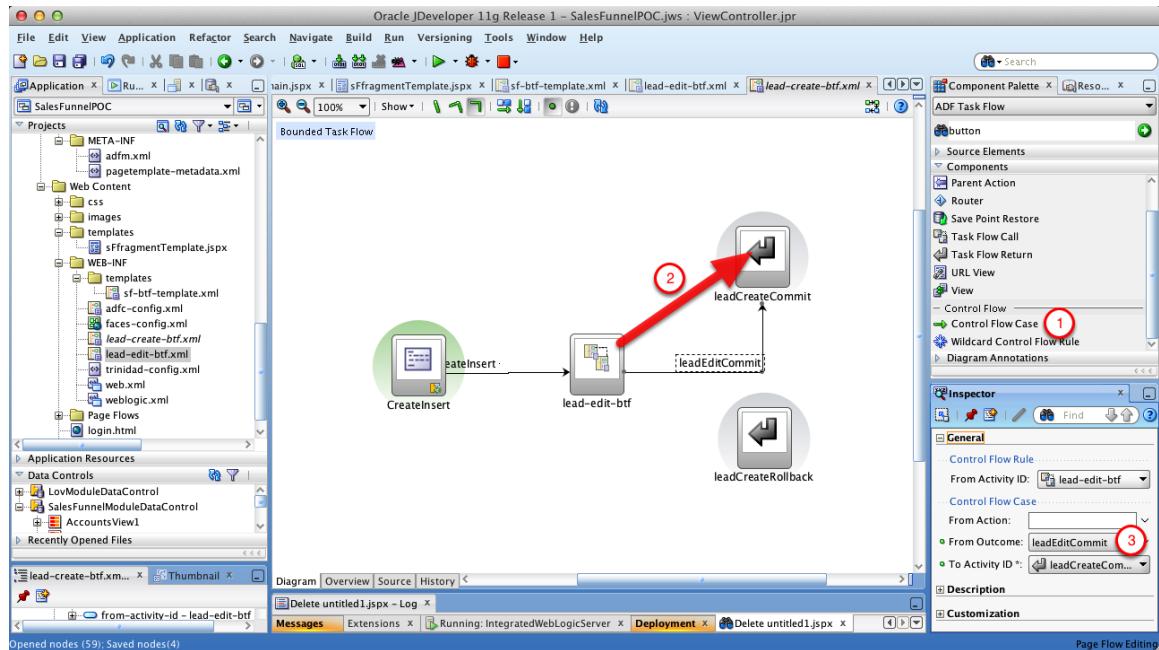
1. Select Control Flow Case in Component Palette
2. Connect CreateInsert to lead-edit-btf

Add Task Flow Returns



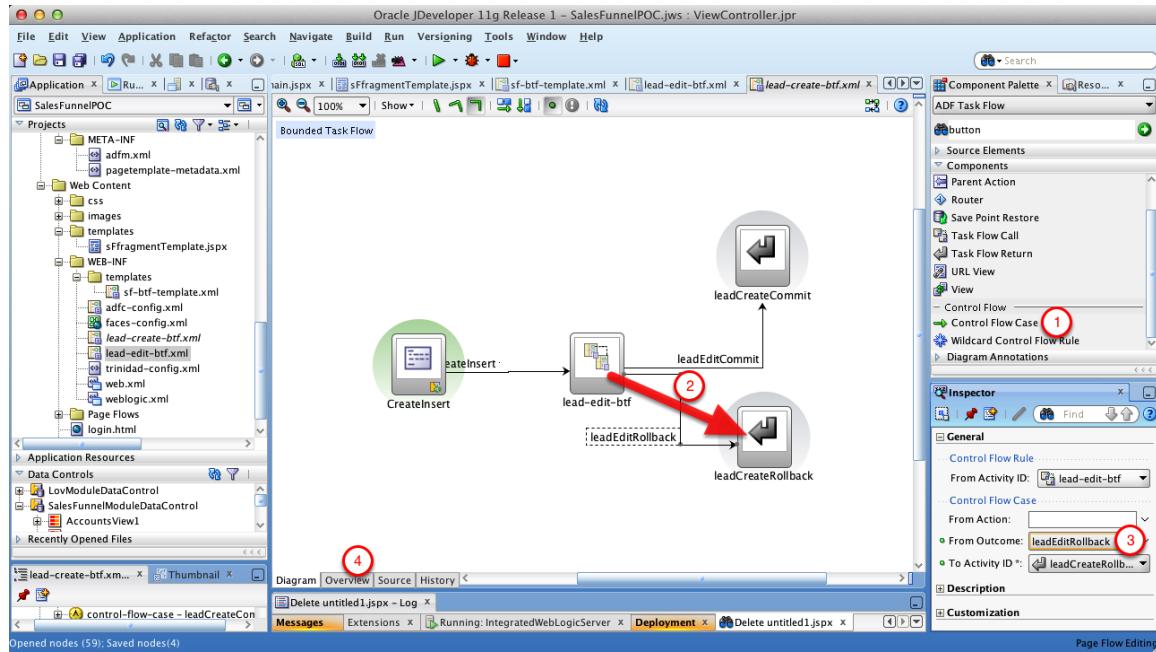
1. Drag and drop Task Flow Return activity to the diagram
2. Rename it to leadCreateCommit
3. Drag and drop Task Flow Return activity to the diagram
4. Rename it to leadCreateRollback

Connect lead-edit-btf to leadCreateCommit



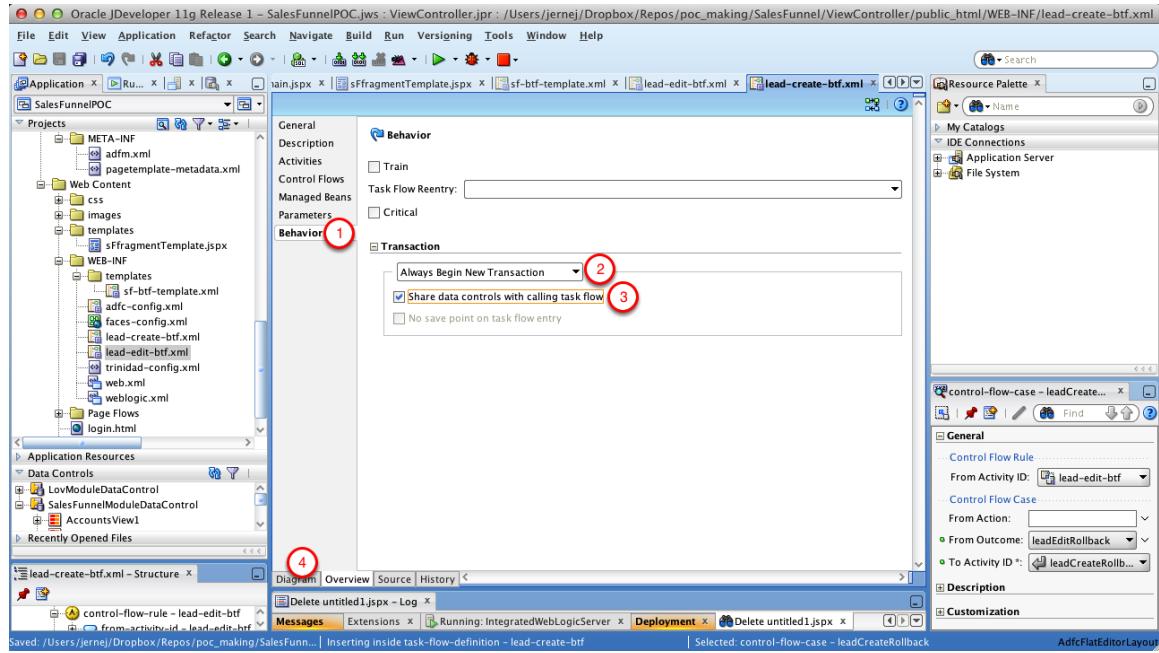
1. Select Control Flow Case
2. Connect lead-edit-btf to leadCreateCommit
3. Set From Outcome to leadEditCommit

Connect lead-edit-btf to leadCreateRollback



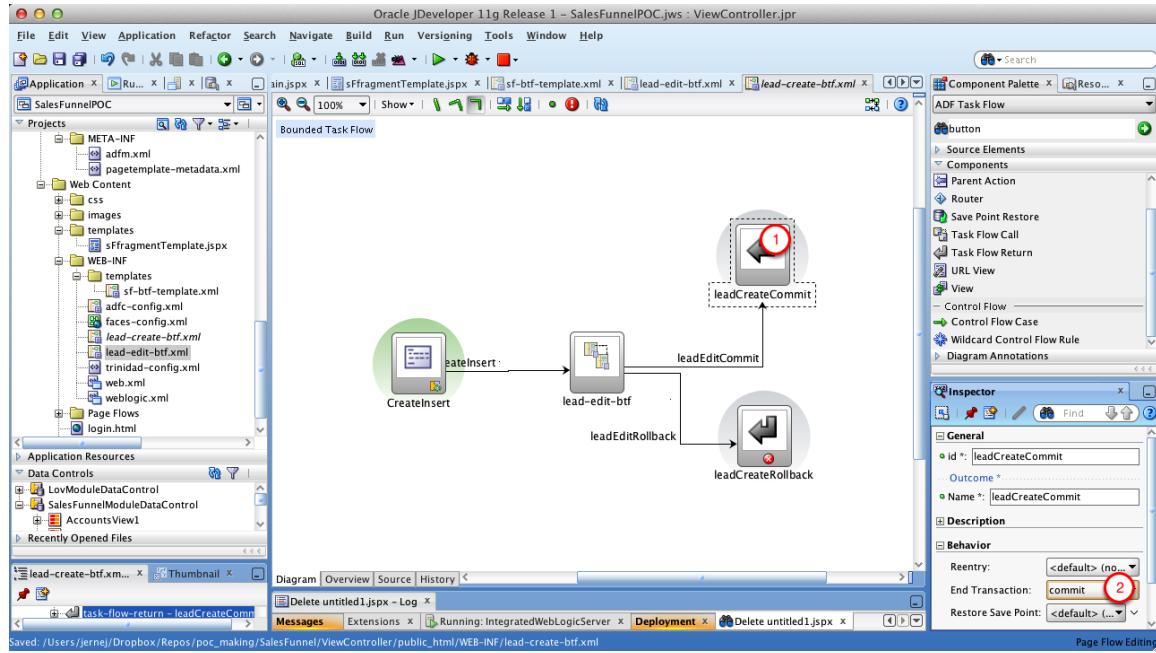
1. Select Control Flow Case
2. Connect lead-edit-btf to leadCreateRollback
3. Set From Outcome to leadEditRollback
4. Open Overview Tab

Configure lead-create-btf behavior



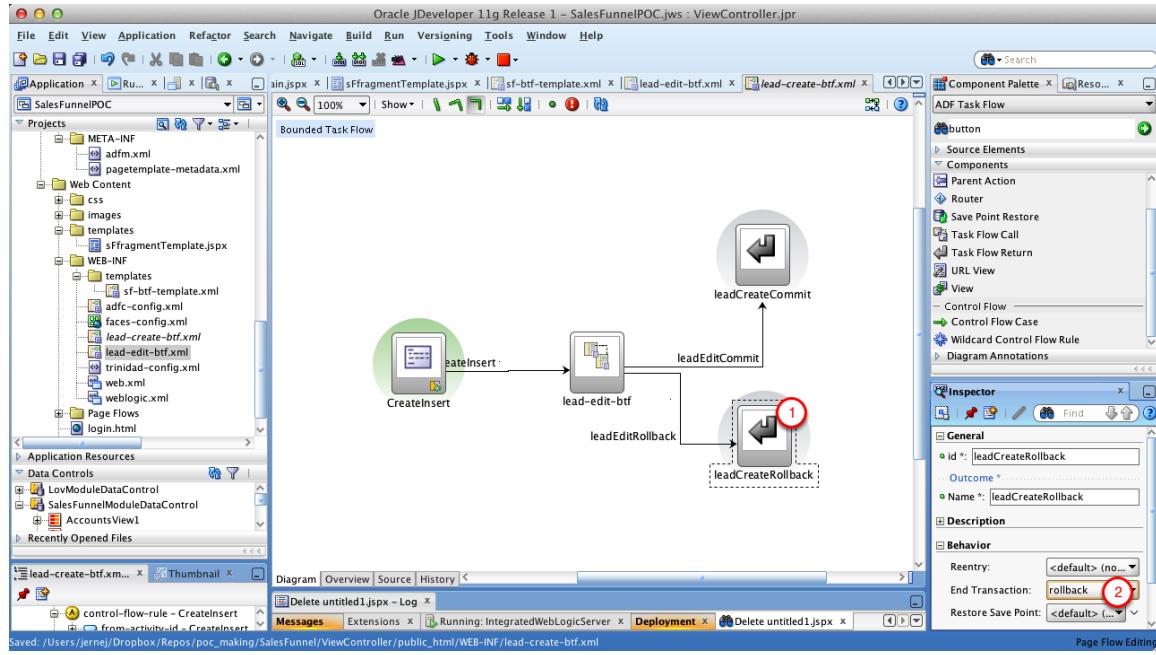
1. Open Behavior tab
2. Set transaction option to "Always Begin New Transaction"
3. Check "Share Data controls with calling task flow"
4. Open Diagram tab

Configure leadCreateCommit



1. Select leadCreateCommit
2. Set End Transaction to commit

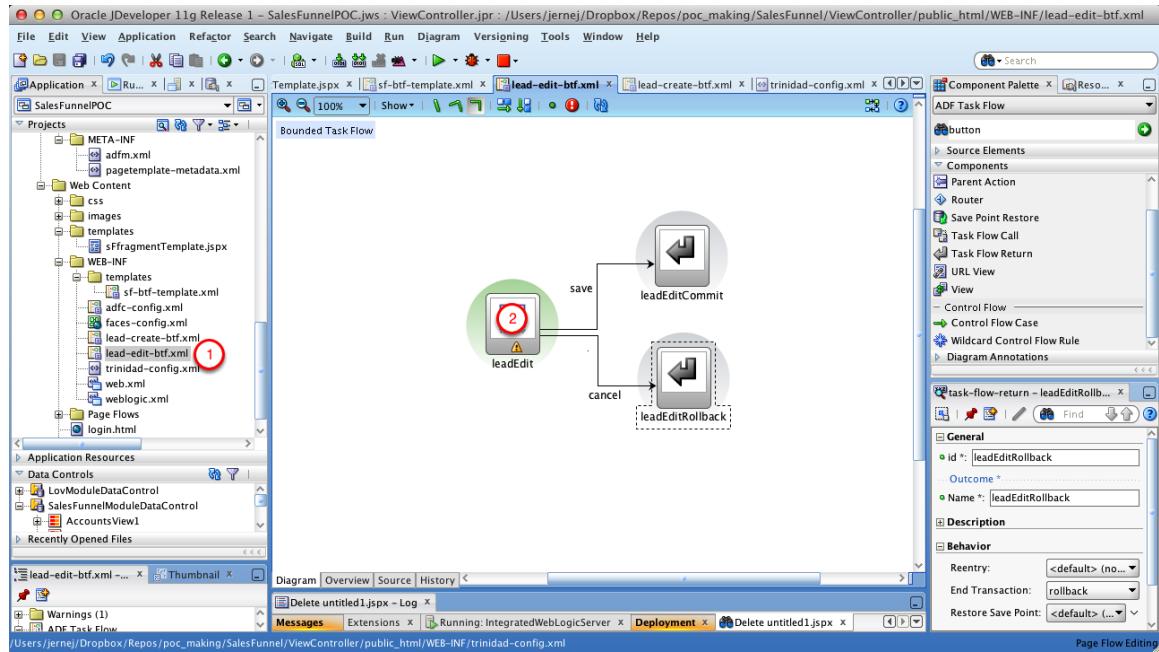
Configure leadCreateRollback



1. Select leadCreateRollback
2. Set End Transaction to rollback

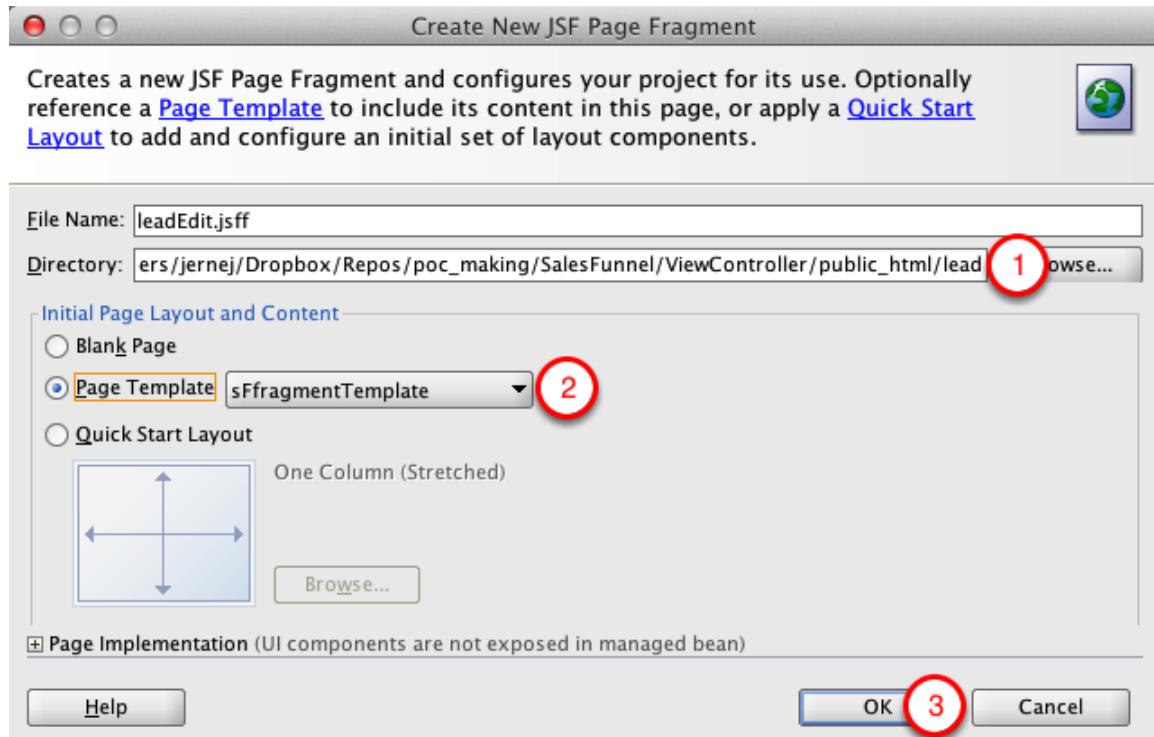
23. Lead Edit Form

Start Page Fragment Wizard



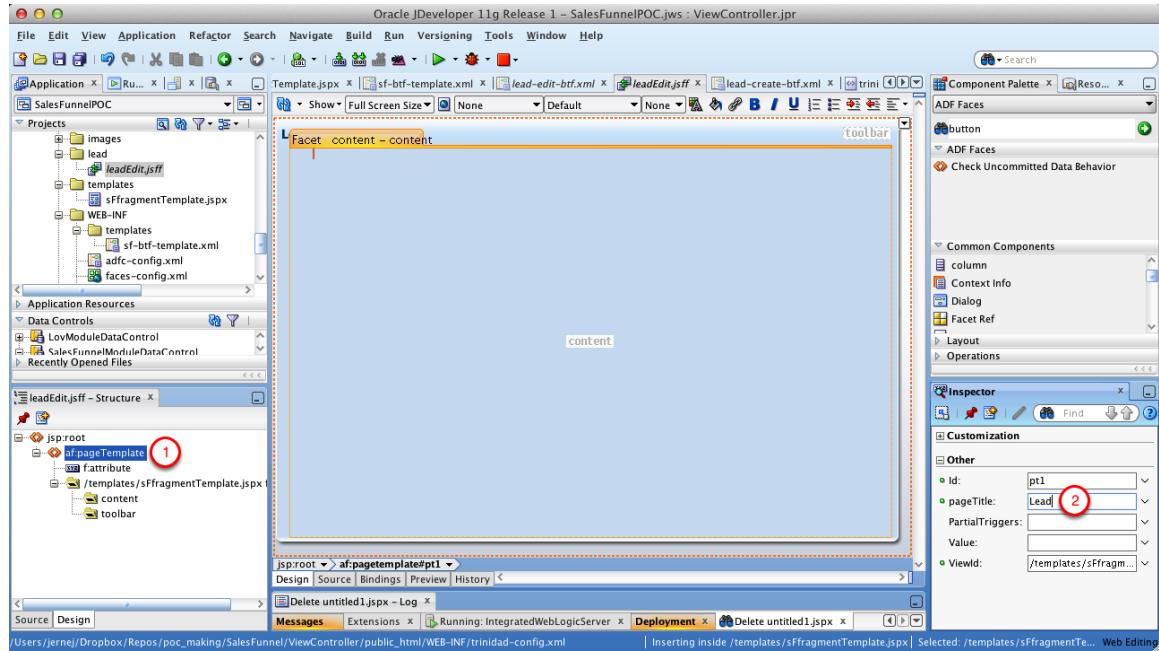
1. Open lead-edit-btf.xml
2. Double click leadEdit view

Create New JSF Page Fragment



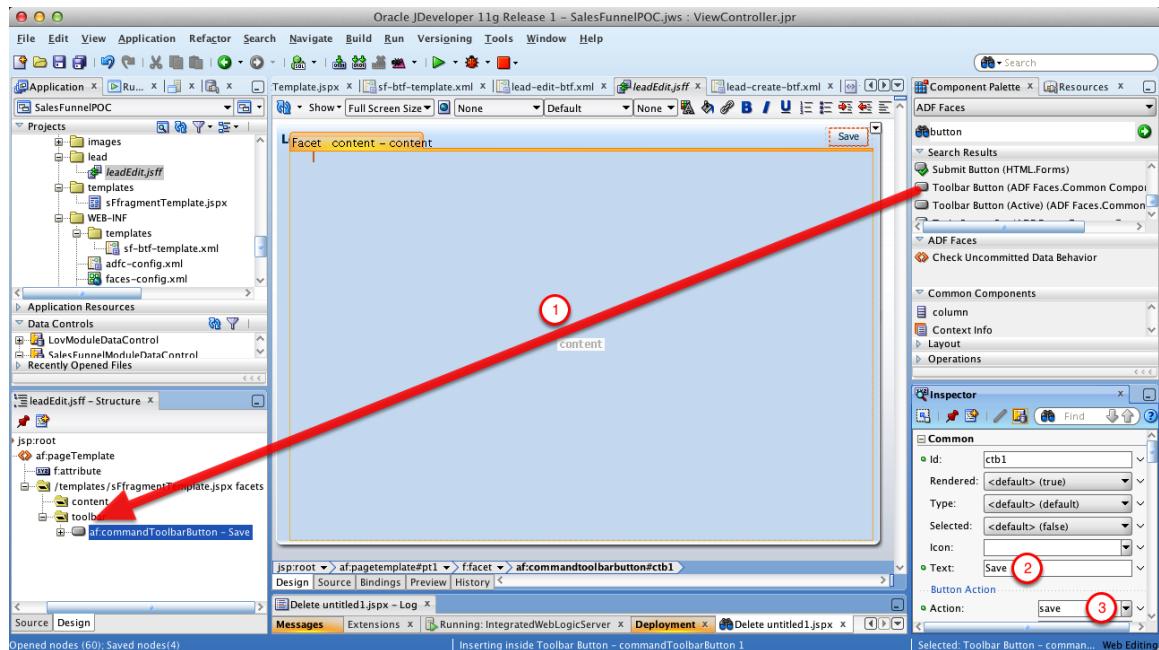
1. Append "/lead" to Directory
2. Set Page Template to sFfragmentTemplate
3. Click OK

Configure template attributes



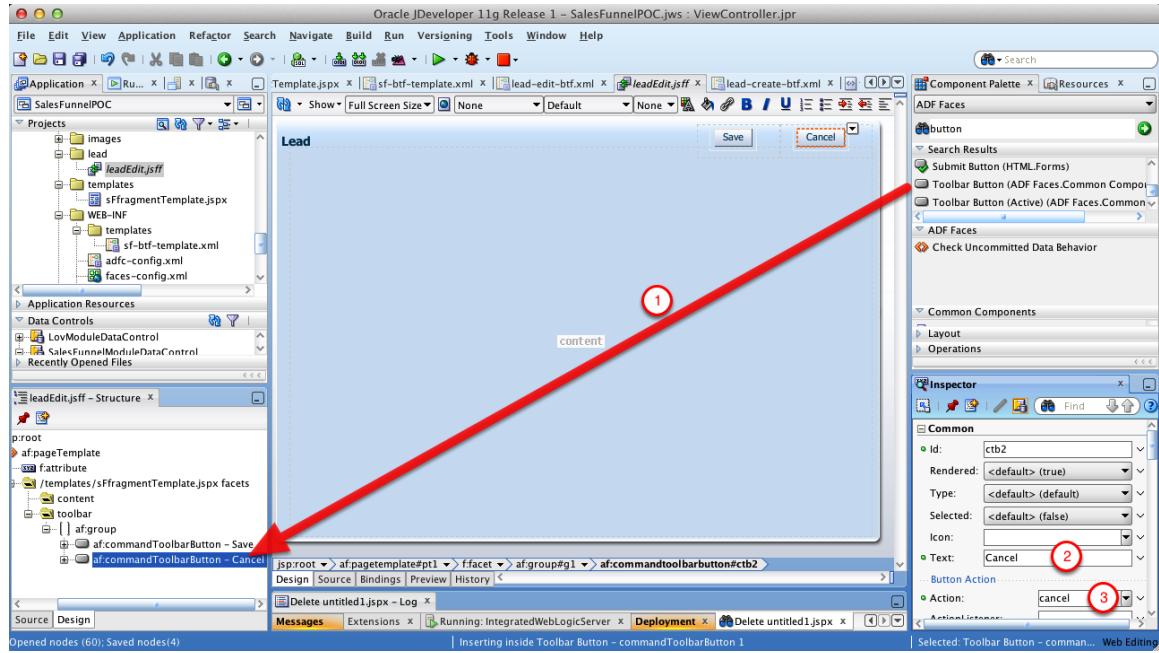
1. Select af:pageTemplate
2. Set pageTitle to Lead

Add Save button



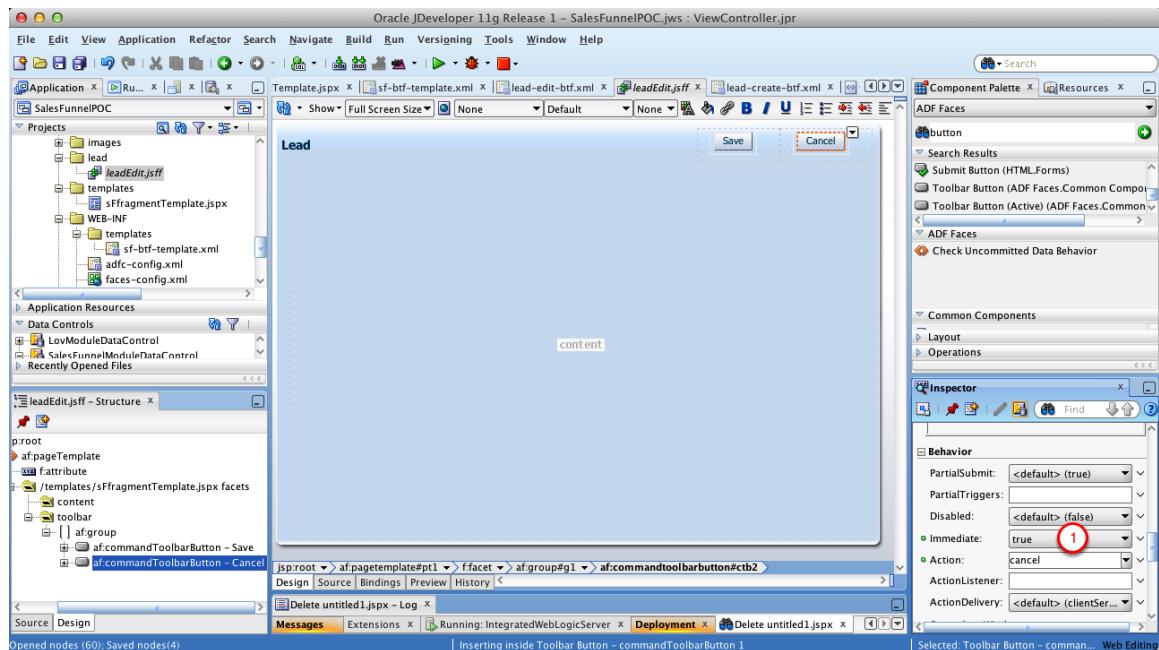
1. Drag and drop Toolbar Button to toolbar facet
2. Set Text to Save
3. Set action to save

Add Cancel button



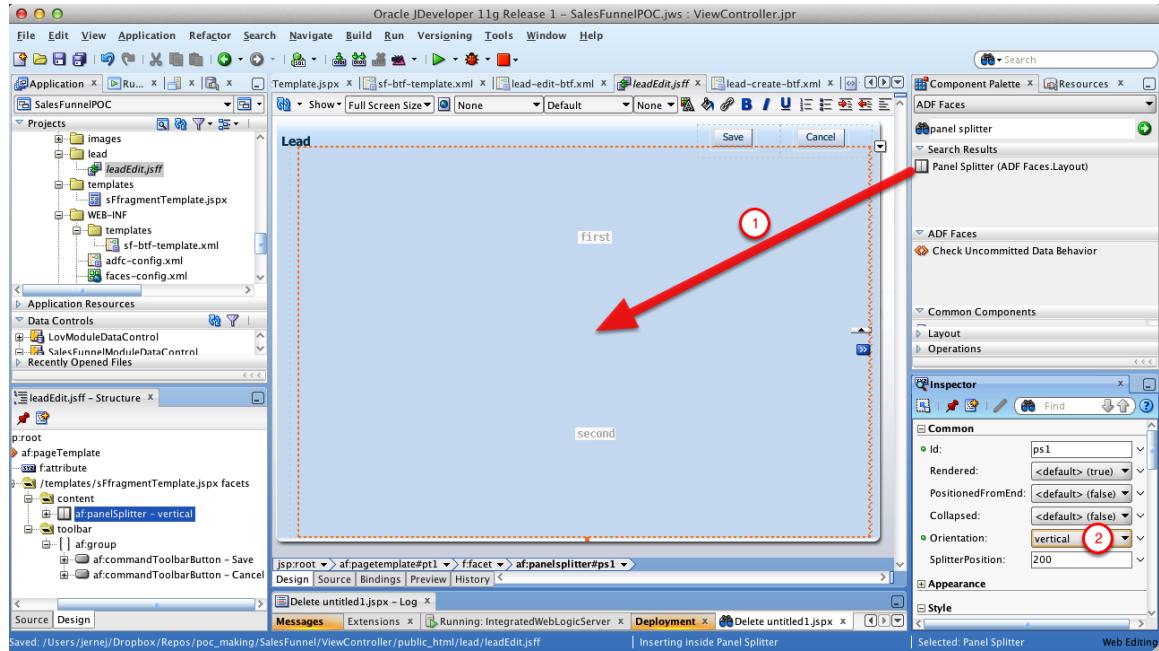
1. Drag and drop Toolbar Button to toolbar facet*
 2. Set Text to Cancel
 3. Set action to cancel
- *af:group will be created automatically

Skip validation on Cancel button



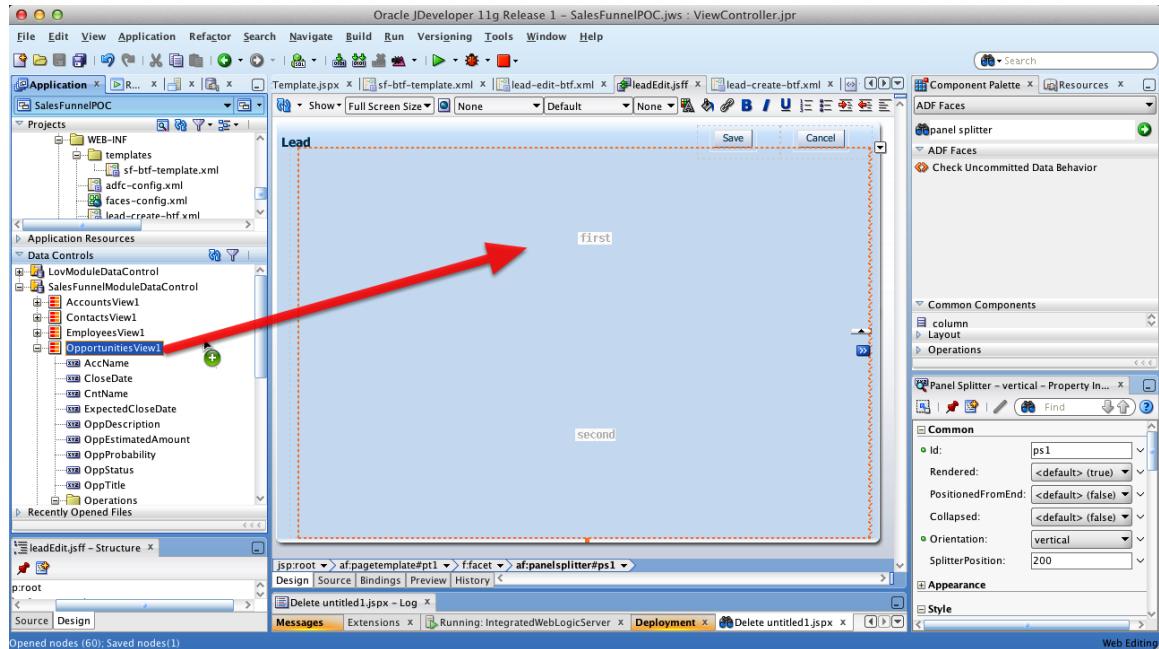
1. Set Immediate property of the Cancel button to true

Add Panel Splitter



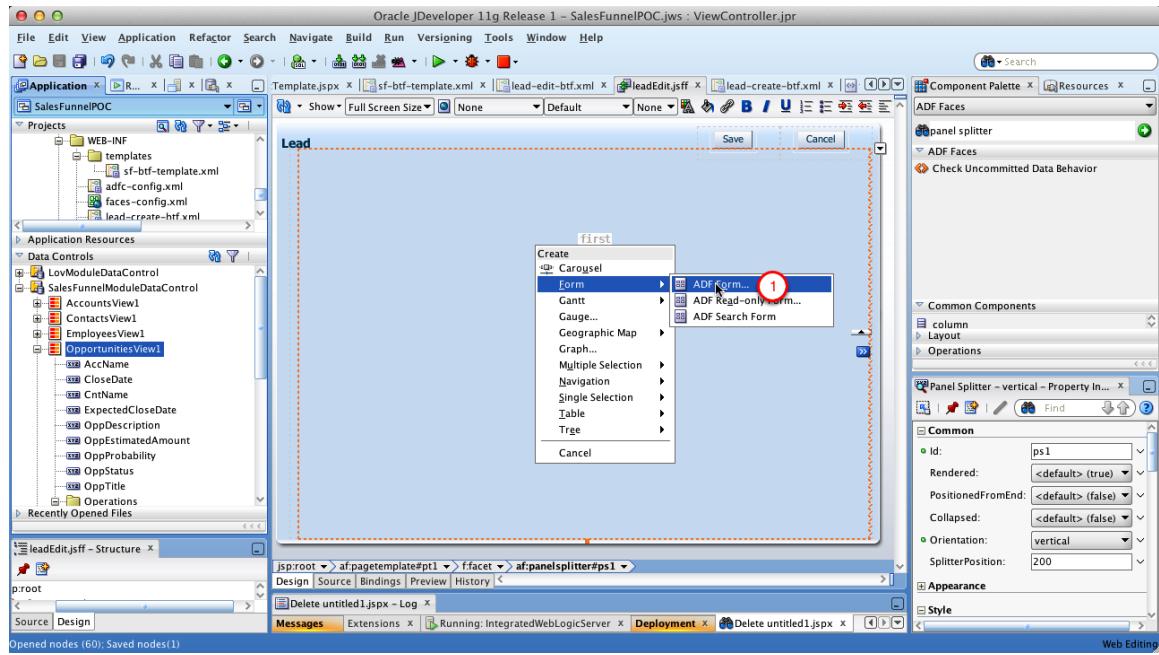
1. Drag and drop Panel Splitter to the content facet
2. Set Orientation to vertical

Drag And Drop OpportunitiesView1 on the form



1. Drag and drop OpportunitiesView1 to the first facet of the panel splitter

Create ADF Form



1. Select Form->Adf Form from the context menu*

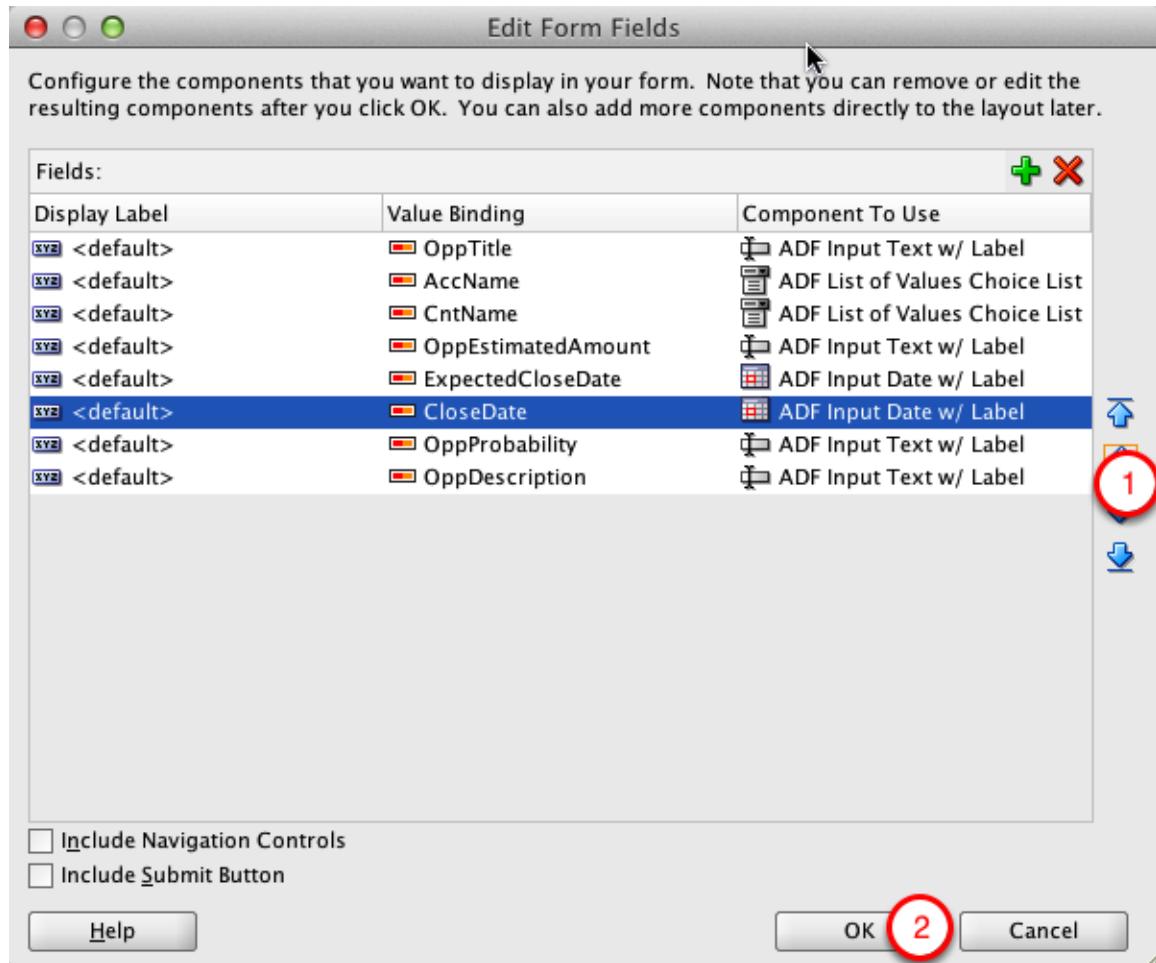
*the menu opens after Drag and drop operation

Edit Form Fields

Configure the components that you want to display in your form. Note that you can remove or edit the resulting components after you click OK. You can also add more components directly to the layout later.

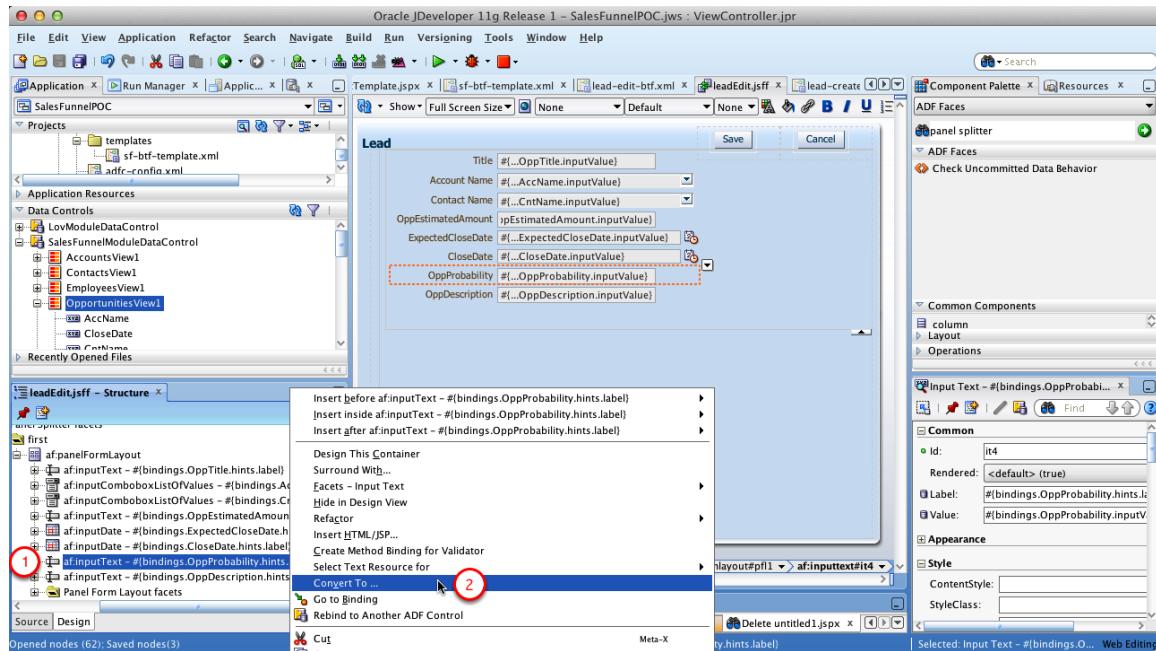
Fields:		
Display Label	Value Binding	Component To Use
xyz <default>	OppTitle	ADF Input Text w/ Label
xyz <default>	AccName	ADF List of Values Choice List
xyz <default>	CntName	ADF List of Values Choice List
xyz <default>	OppEstimatedAmount	ADF Input Text w/ Label
xyz <default>	ExpectedCloseDate	ADF Input Date w/ Label
xyz <default>	CloseDate	ADF Input Date w/ Label
xyz <default>	OppProbability	ADF Input Text w/ Label
xyz <default>	OppDescription	ADF Input Text w/ Label

Include Navigation Controls
 Include Submit Button



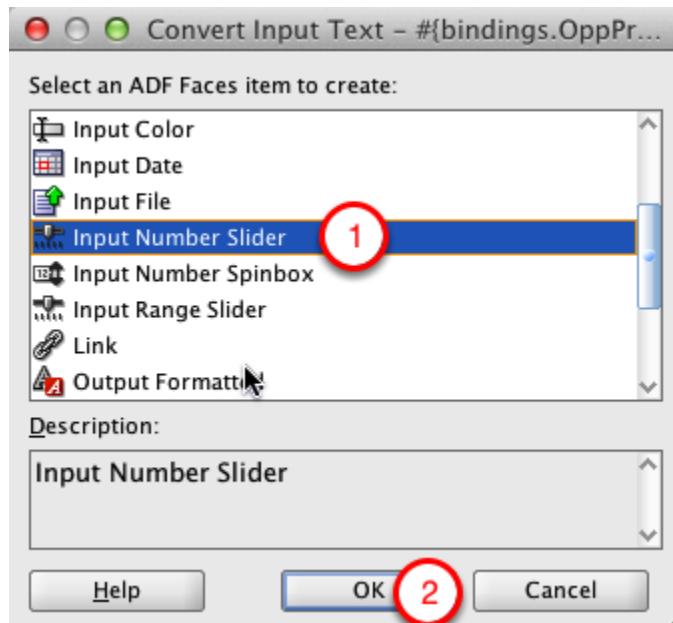
1. Use up and down arrows to organize the attributes to match the screenshot
2. Click OK

Convert OppProbability input text



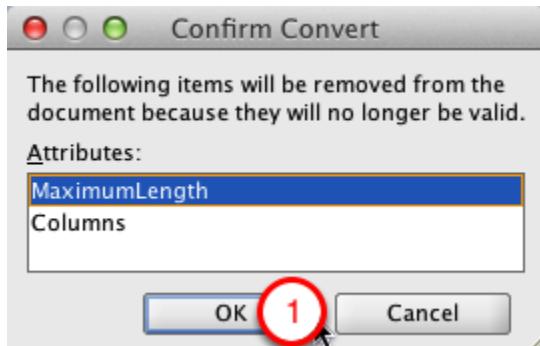
1. Expand panelFormLayout in the Structure Window and right-click OppProbability inputText
2. Select Convert To in the popup menu

Convert Input Text - #{bindings.OppProbability.hints.label}



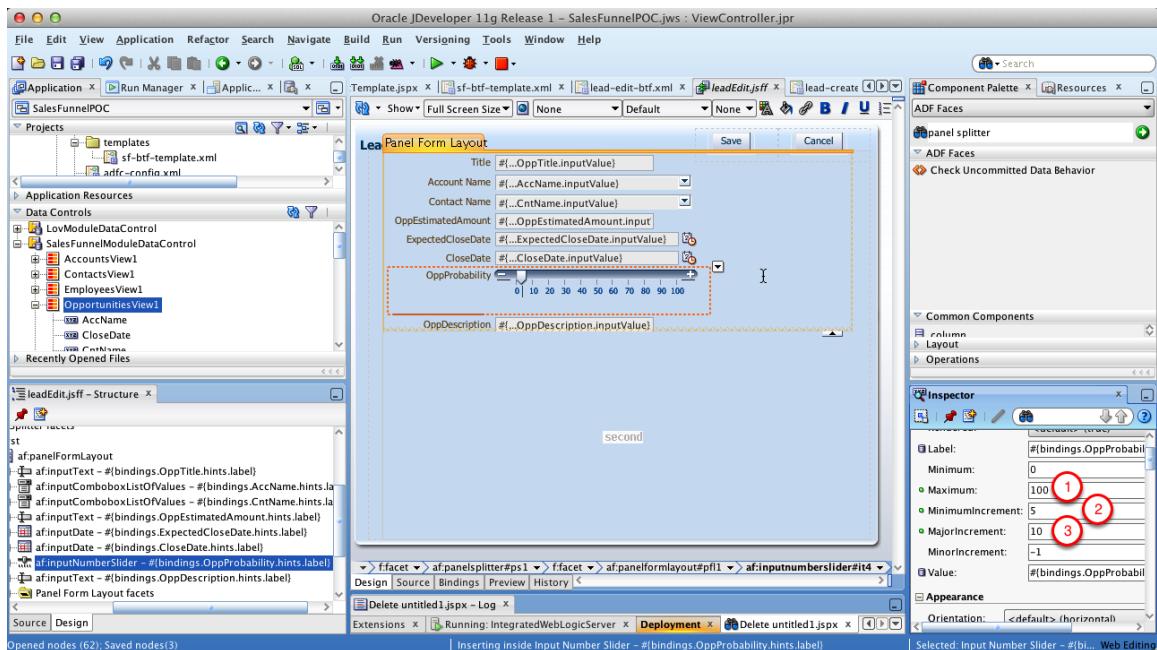
1. Select Input Number Slider
2. Click OK

Confirm Convert



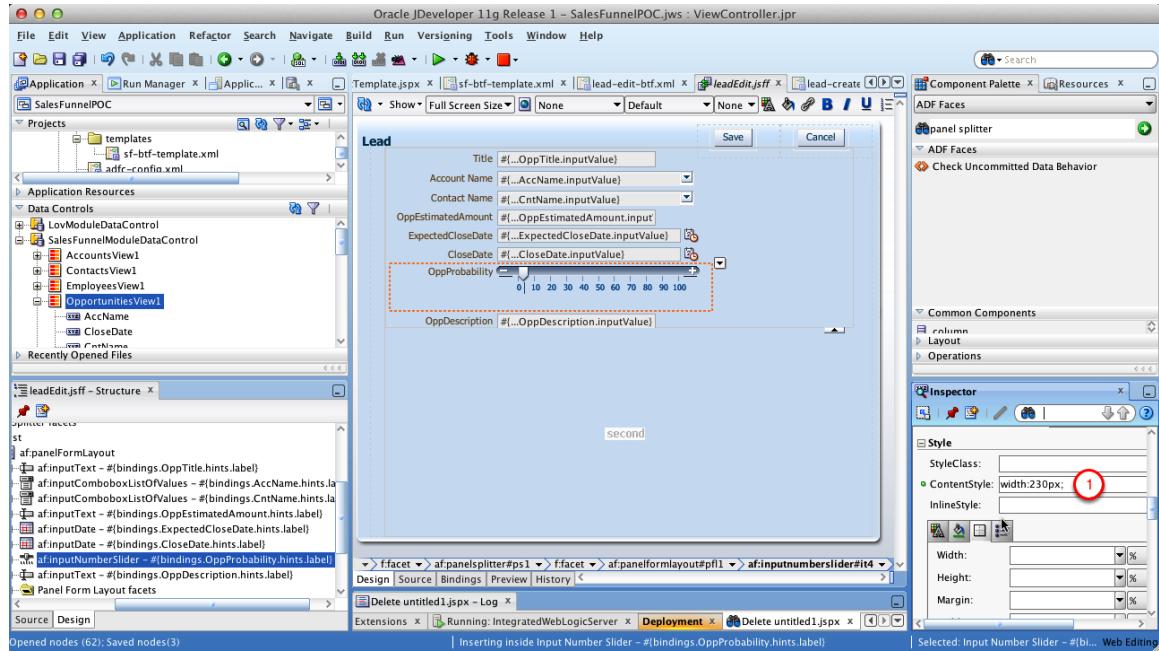
1. Click OK

Set inputNumberSlider properties



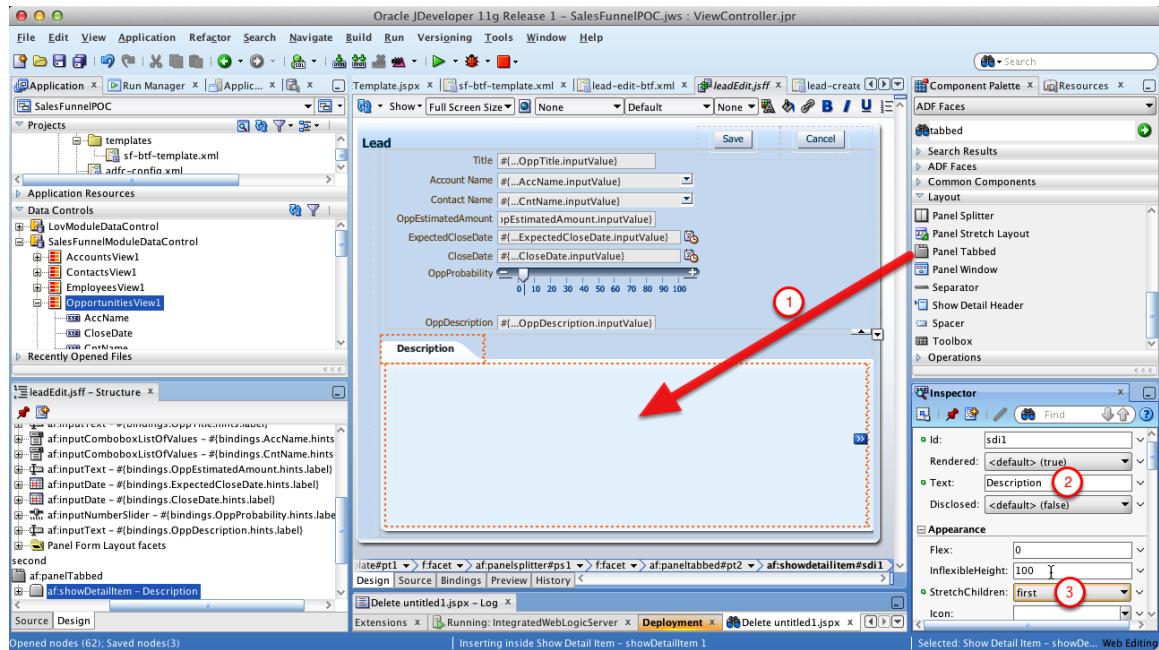
1. Set Maximum: 100
2. MinimumIncrement: 5
3. MajorIncrement: 10

Set OppProbability ContentStyle / width



1. Set ContentStyle to "width:230px;" without quotes

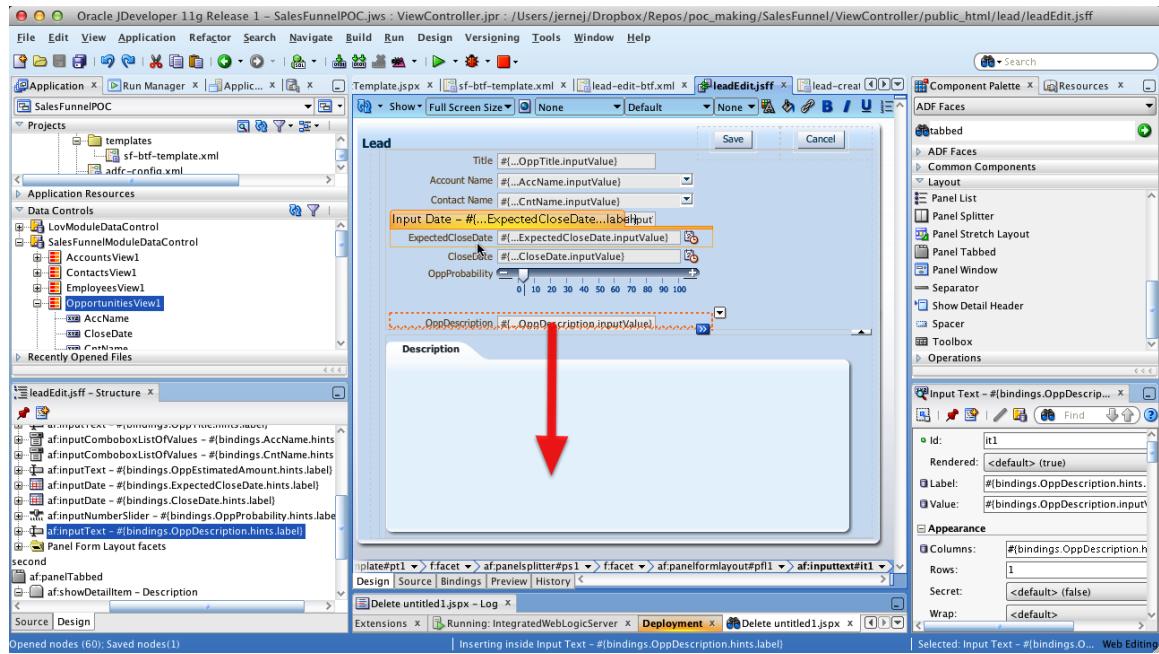
Add Panel Tabbed



1. Drag and drop Panel Tabbed to the second facet of the panel splitter
2. Set Text to Description

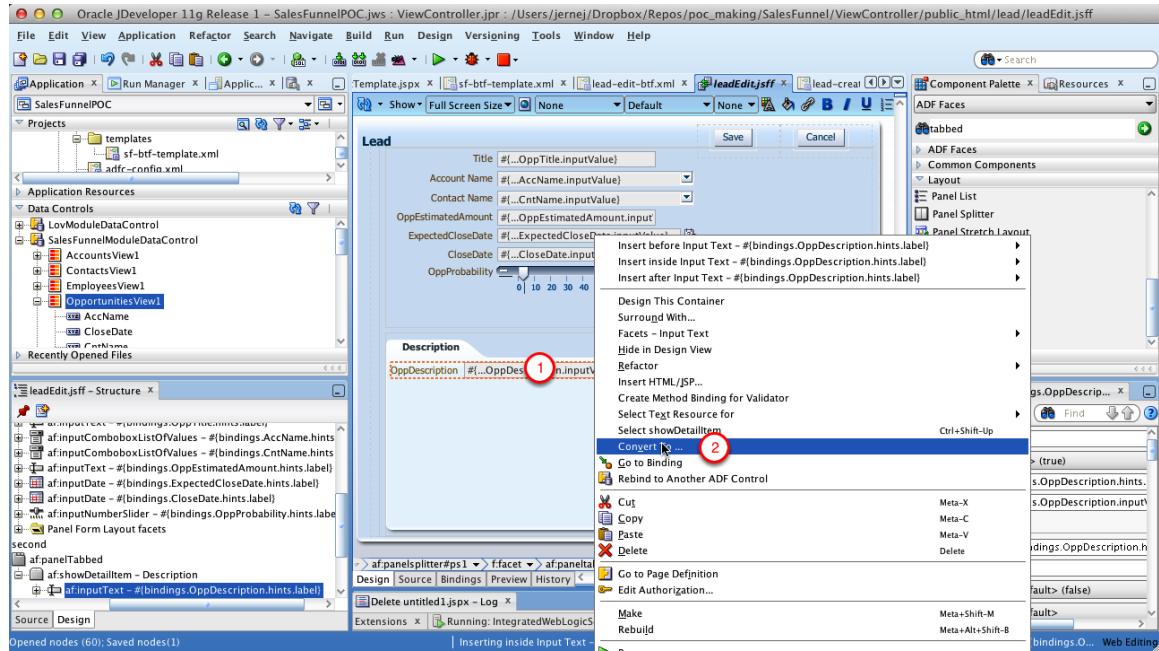
3. Set StretchChildren to first

Move OppDescription to the tab



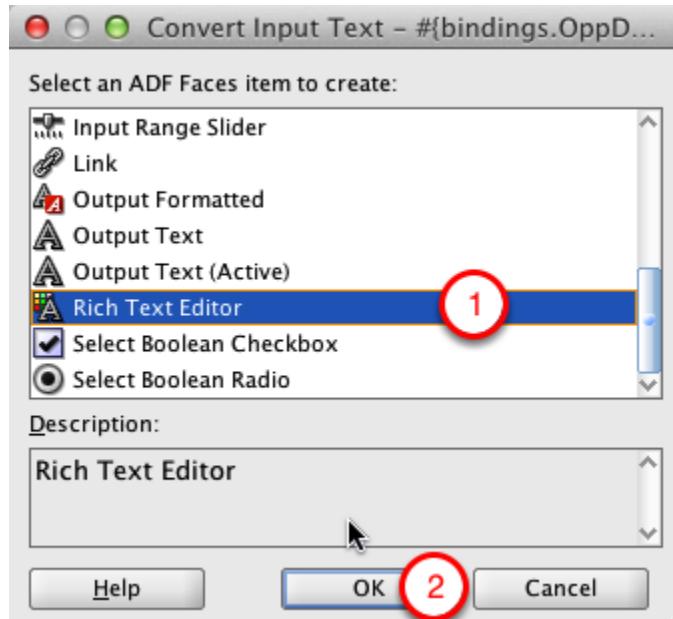
1. Drag and drop OppDescription inputText on description tab

Convert OppDescription



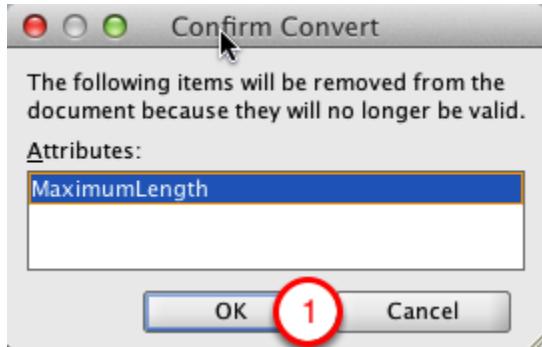
1. Right-click oppDescription
2. Click Convert to in the popup menu

Convert Input Text - #{bindings.OppDescription.hints.label}



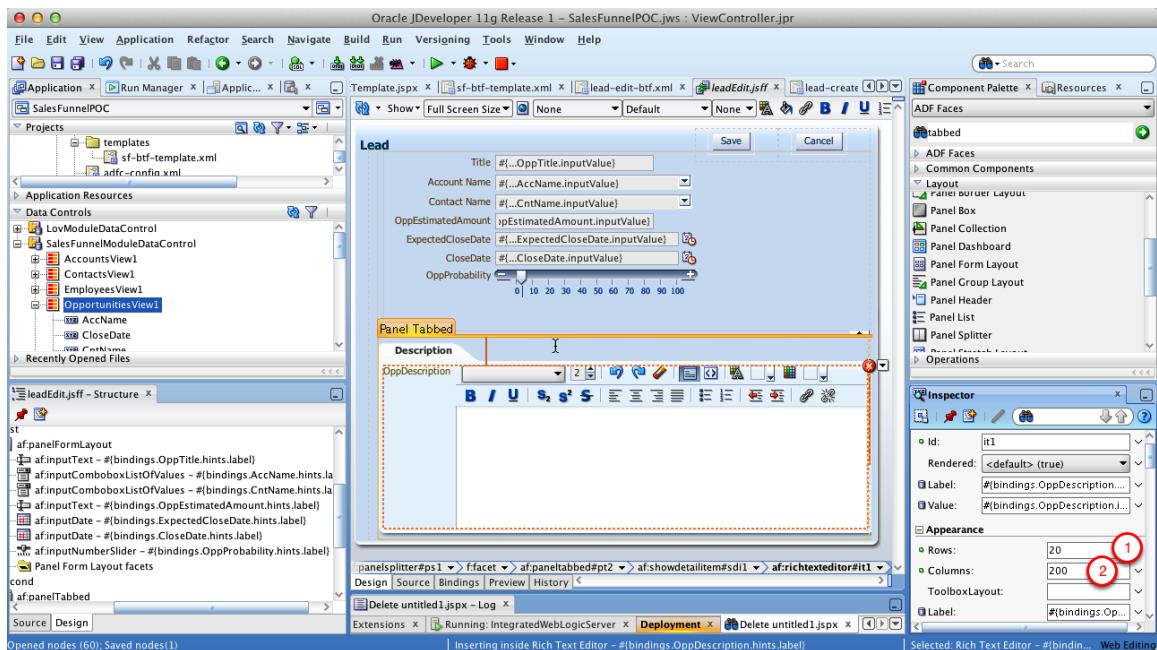
1. Select Rich Text Editor
2. Click OK

Confirm Convert



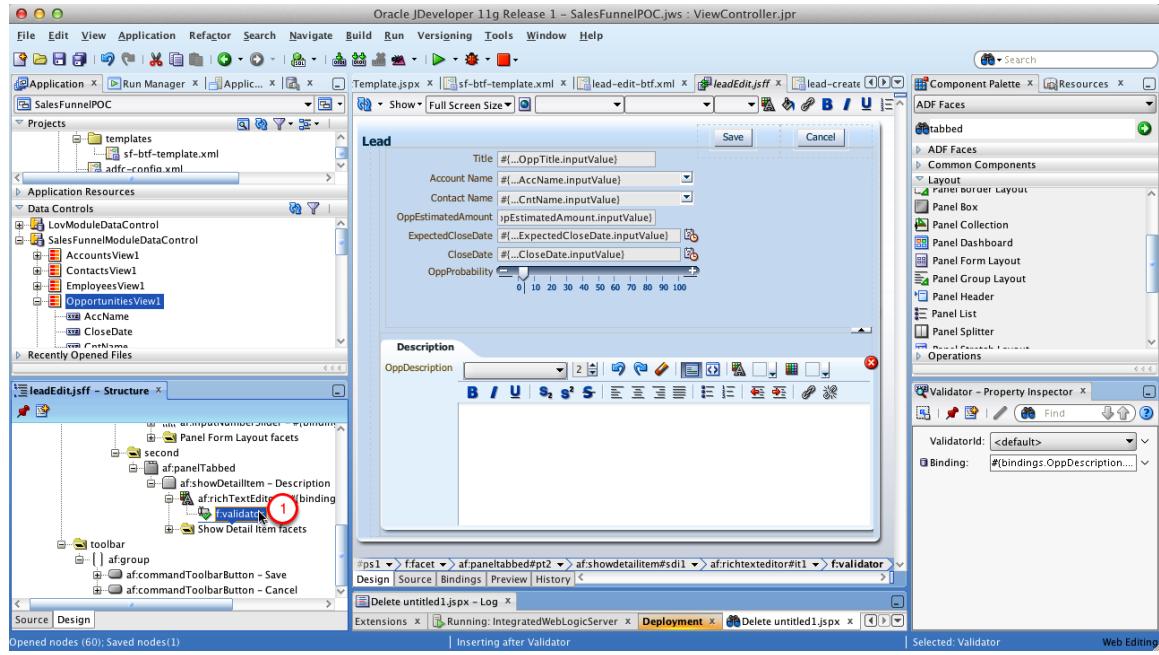
1. Click OK

Set Rich Text Editor properties



1. Set Rows to 20
2. Set Columns to 200

Remove Rich Text Editor validator

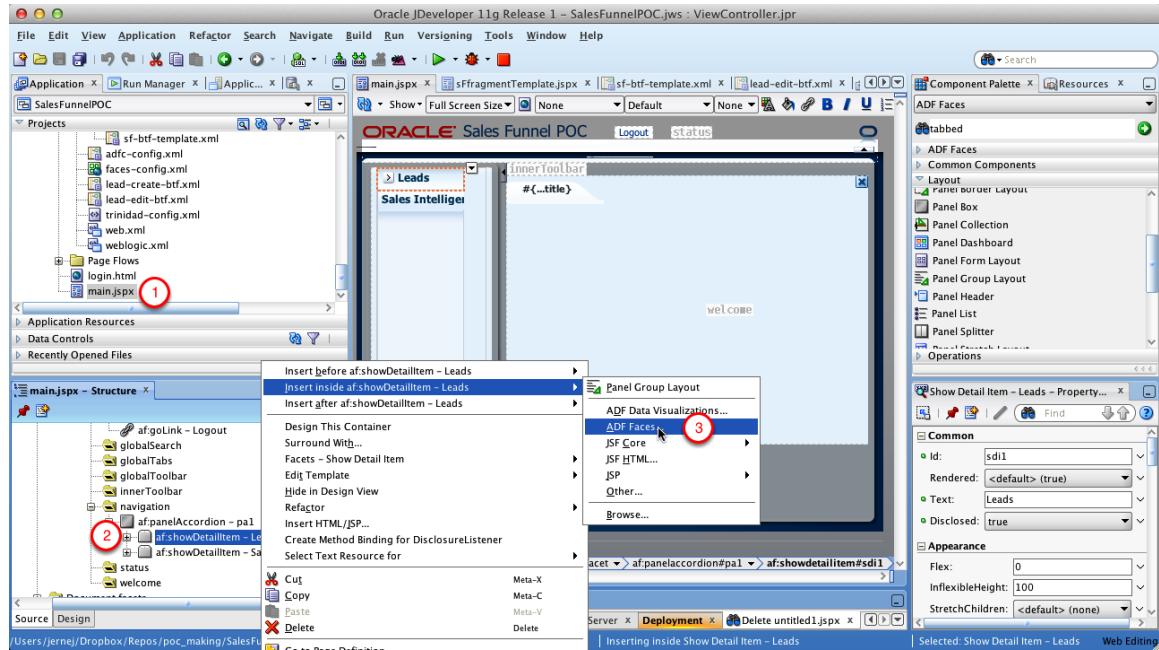


1. Expand richTextEditor, select f:validator and delete it

24. Add Lead Create to Main Form

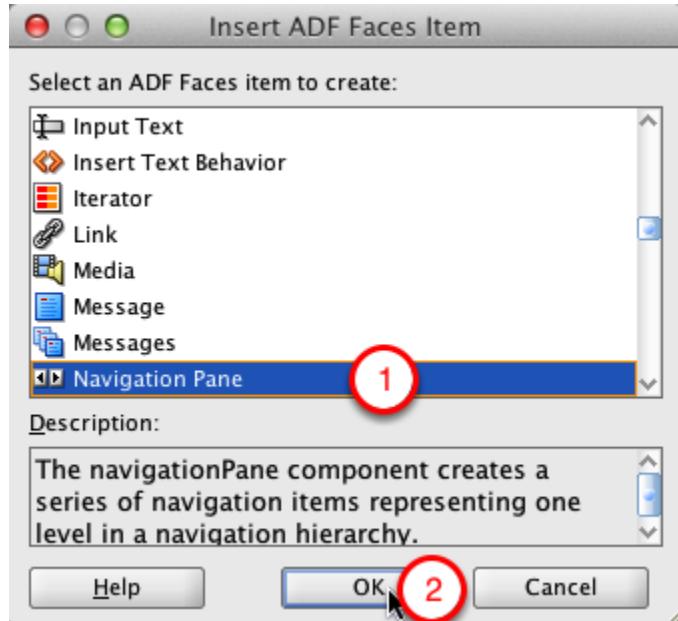
In this lesson we are going to complete the steps needed to add new lead functionality to the Main Form

Add Navigation Pane



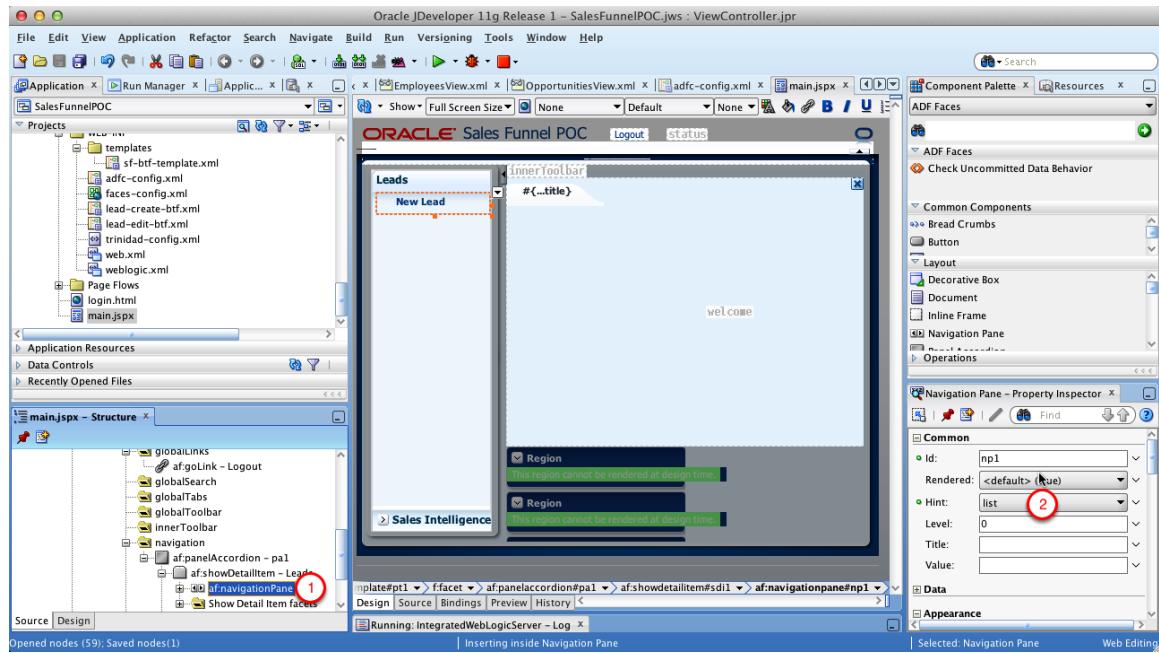
1. Open main.jspx
2. Expand structure, locate and right-click showDetailItem - Leads
3. Select Insert inside ... -> ADF Faces

Insert ADF Faces Item



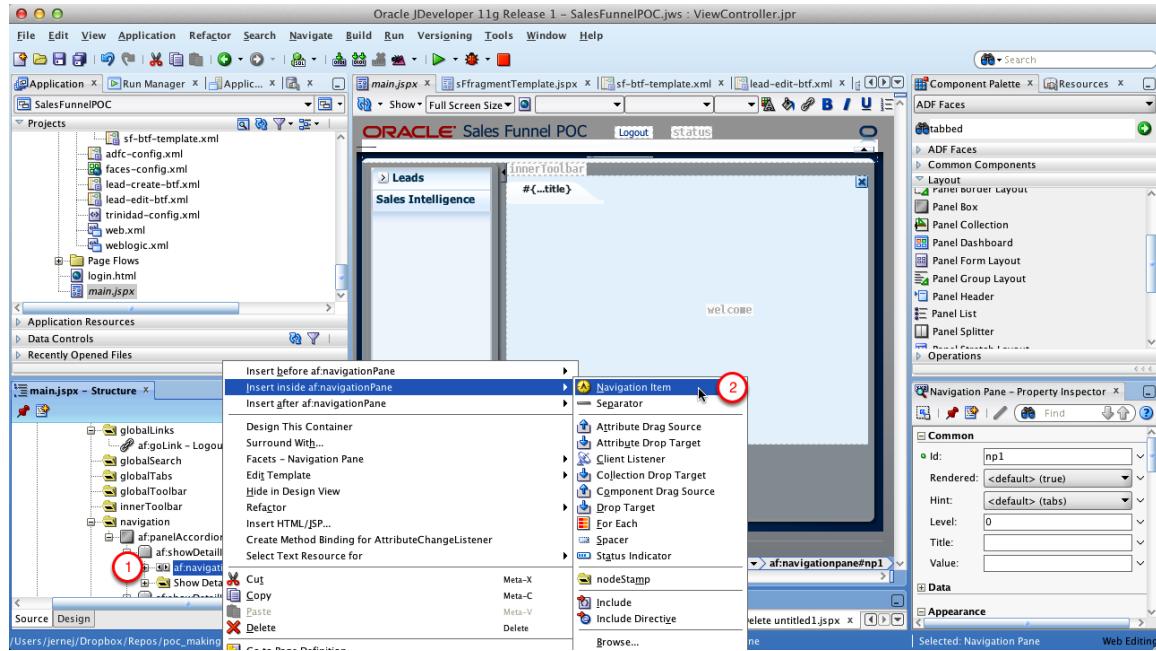
1. Select Navigation Pane
2. Click OK

Set Navigation Pane Hint to List



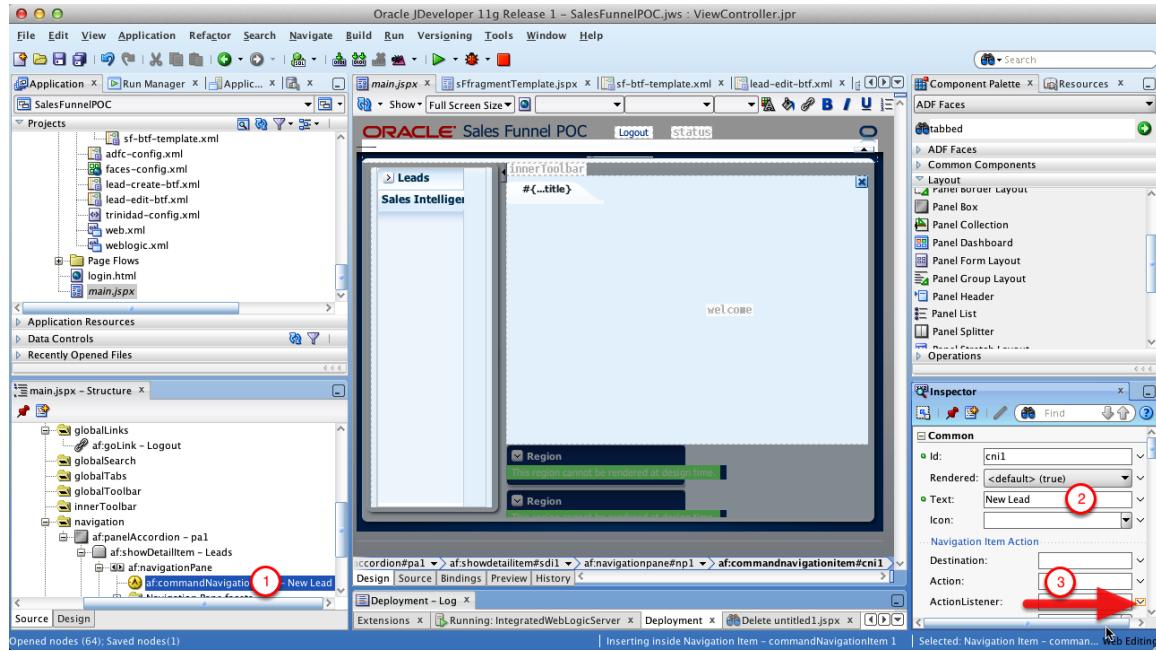
1. Select navigationPane
2. Set Hint to list

Add Navigation Item



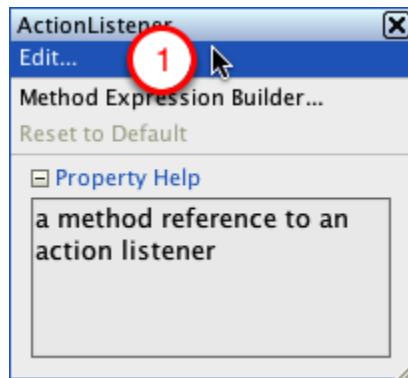
1. Right-click navigationPane
2. Select Insert inside ... -> Navigation Item

Set Navigation Item's properties



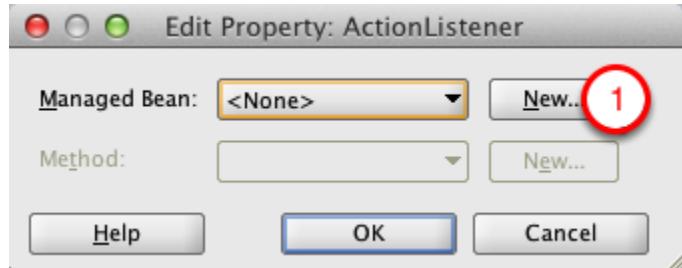
1. Select commandNavigationItem
2. Set Text to New Lead
3. Click the little down arrow next to Action Listener property text box

Edit ActionListener



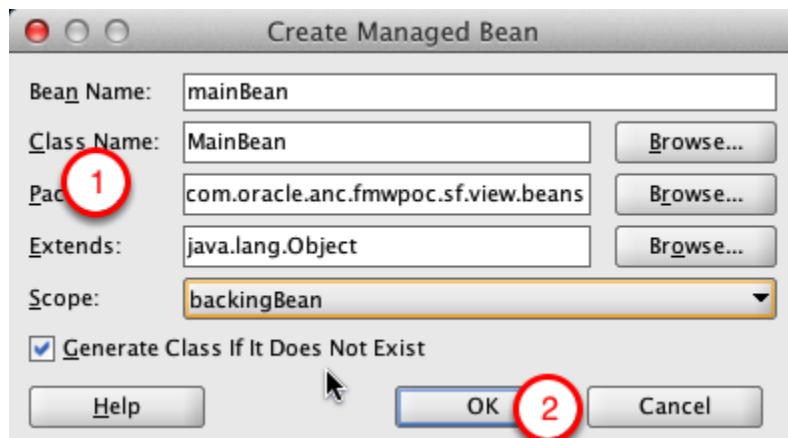
1. Click Edit

Create New Managed Bean



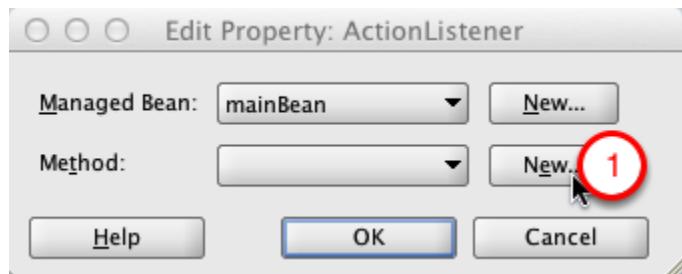
1. Click New next to Managed Bean

Create Managed Bean



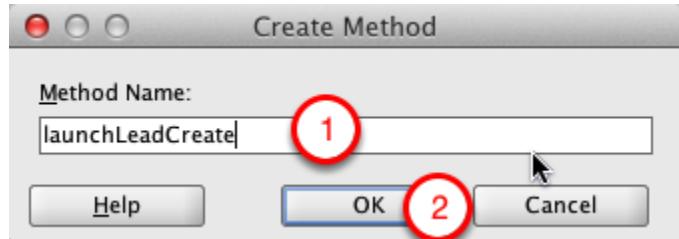
1. Make properties match the screenshot
2. Click OK

Edit Property: ActionListener



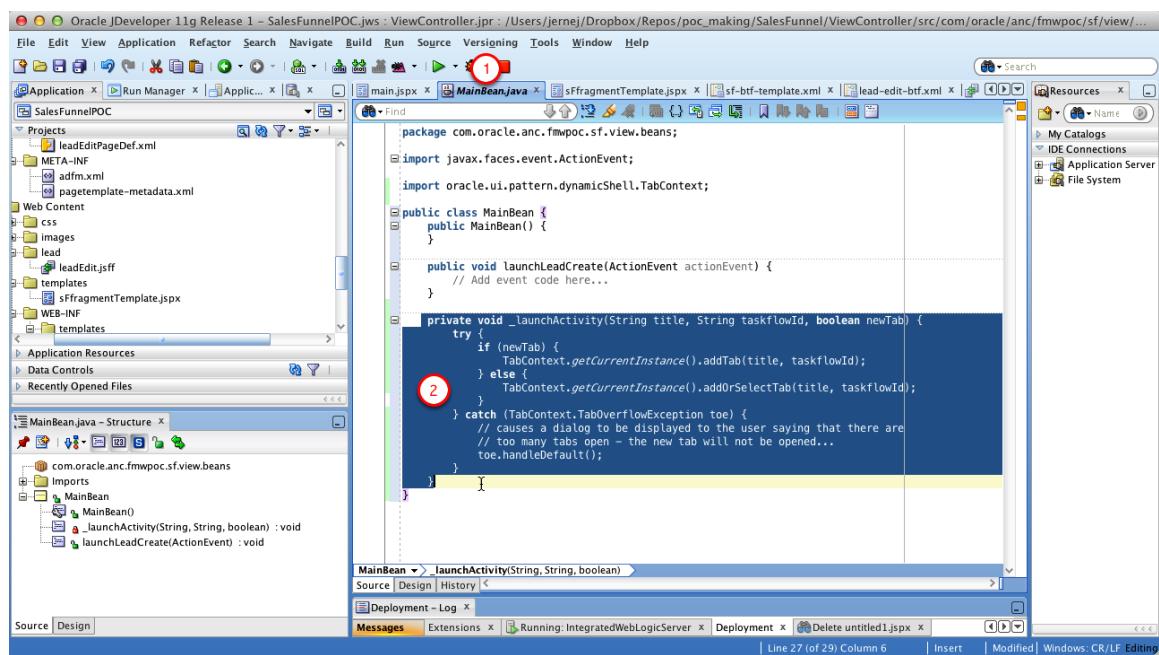
1. Click New next to Method

Create Method



1. Set name to launchLeadCreate
2. Click OK

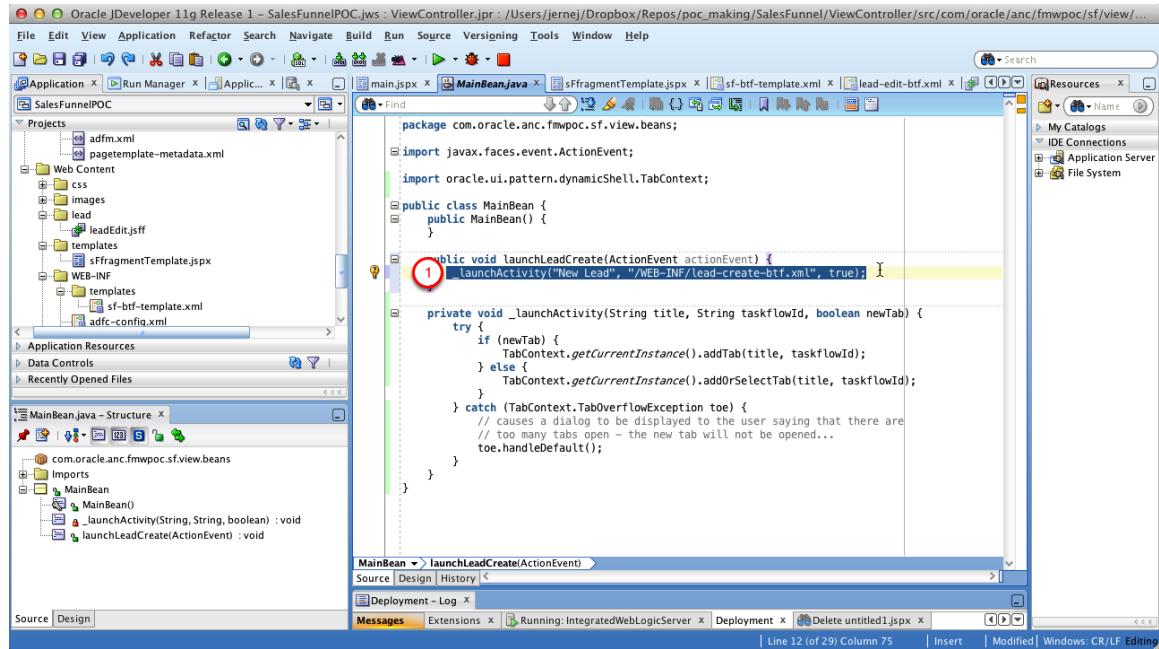
Add UI Shell helper code



1. Locate and open MainBean.java
2. Copy paste the below code inside the bean implementation

```
private void launchActivity(String title, String taskflowId, boolean newTab) {
    try {
        if (newTab) {
            TabContext.getCurrentInstance().addTab(title, taskflowId);
        } else {
            TabContext.getCurrentInstance().addOrSelectTab(title, taskflowId);
        }
    } catch (TabContext.TabOverflowException toe) {
        // causes a dialog to be displayed to the user saying that there are
        // too many tabs open - the new tab will not be opened...
        toe.handleDefault();
    }
}
```

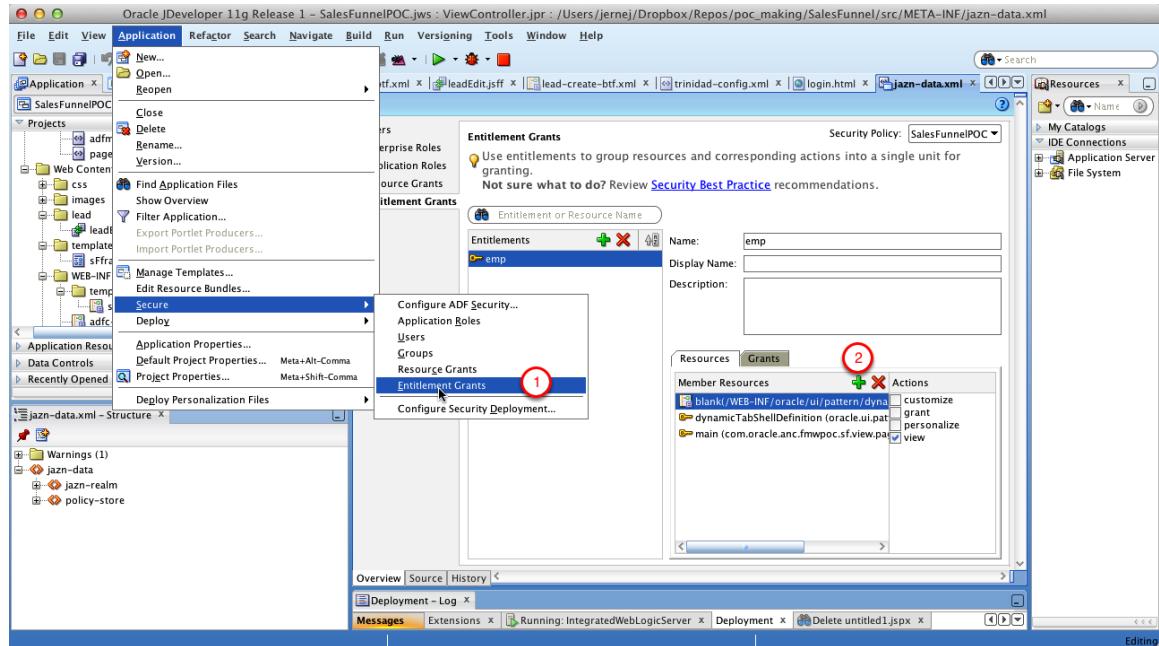
Implement launchLeadCreate



1. Copy paste the below code inside launchLeadCreate method

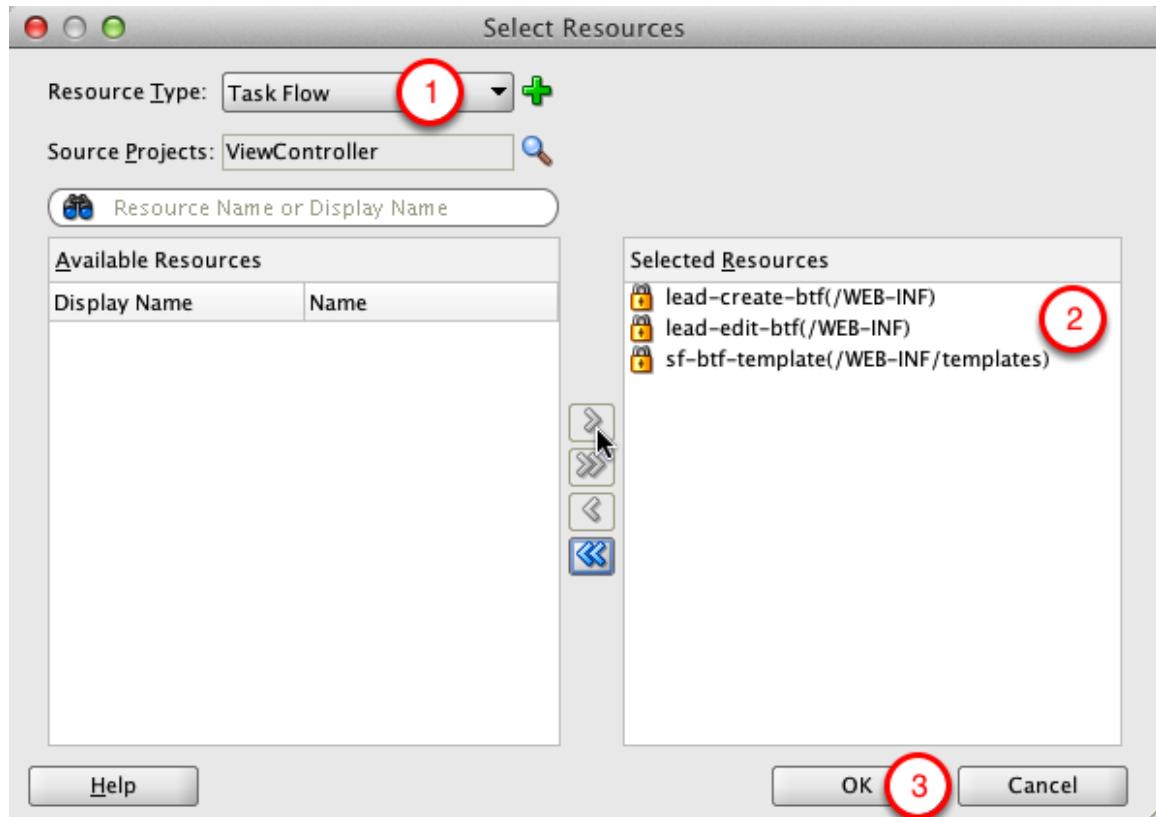
```
_launchActivity("New Lead", "/WEB-INF/lead-create-btf.xml", true);
```

Add security grants



1. In the menu, open Application->Secure->Entitlement Grants

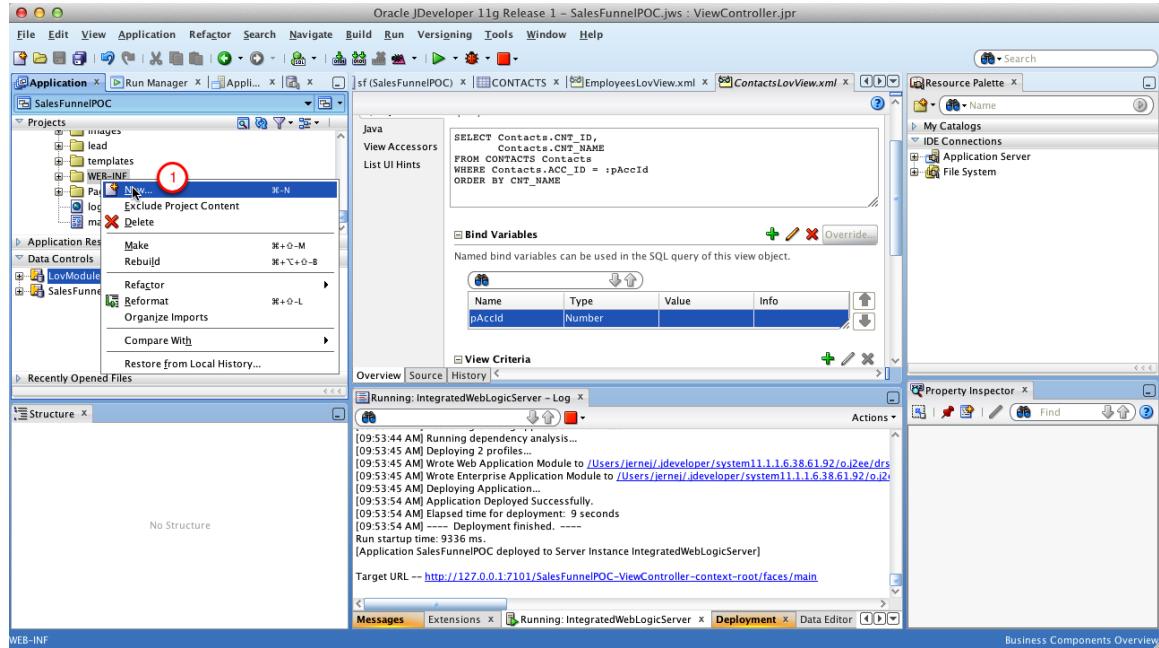
Select Resources



1. Set Resource Type to Task Flow
2. Select and slide lead-create-btf, lead-edit-btf and sf-btf-template to the right
3. Click OK

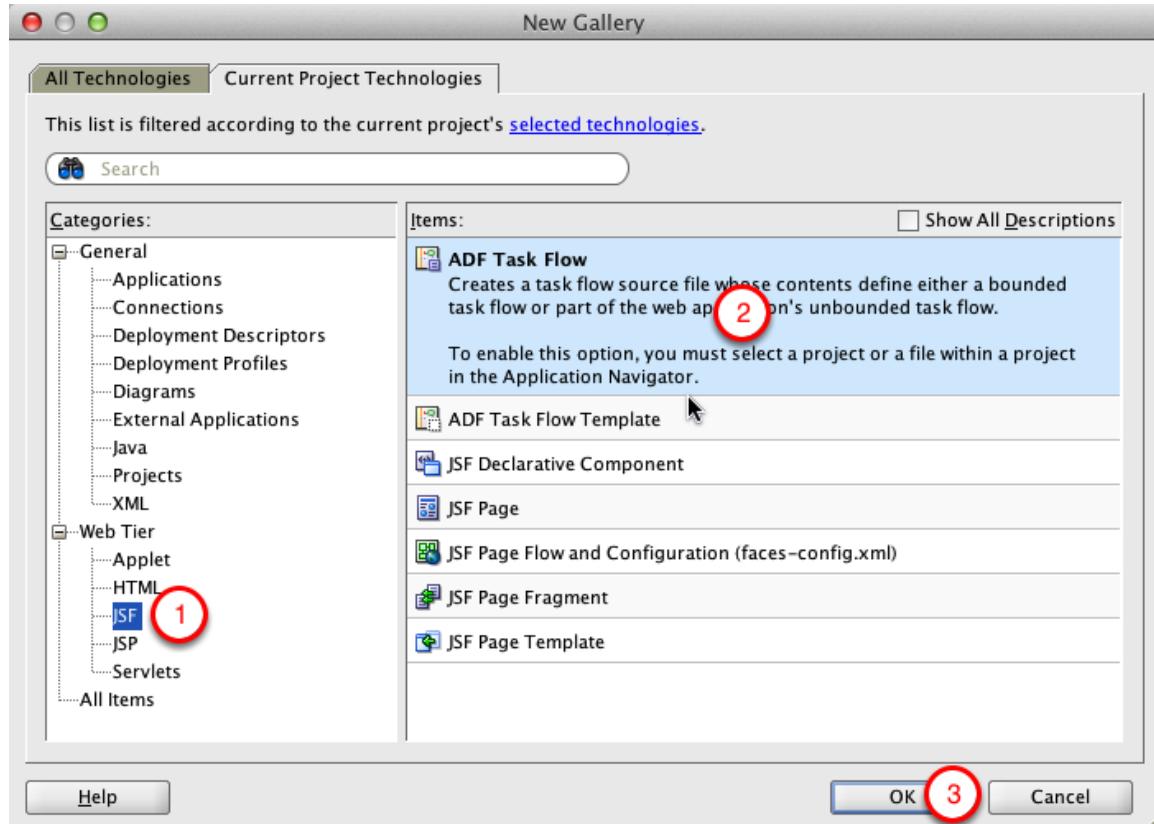
25. Account Edit BTF

Create New Task Flow



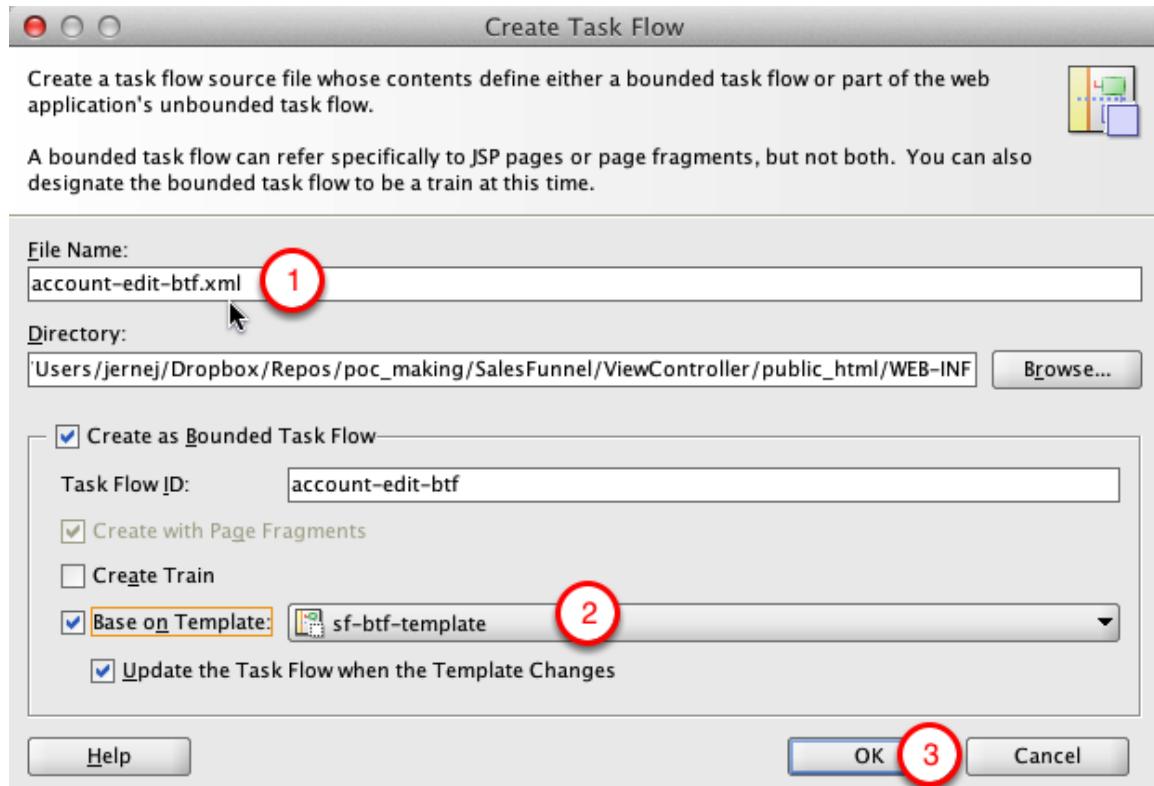
1. Right-click WEB-INF folder and click New in the popup menu

New Gallery



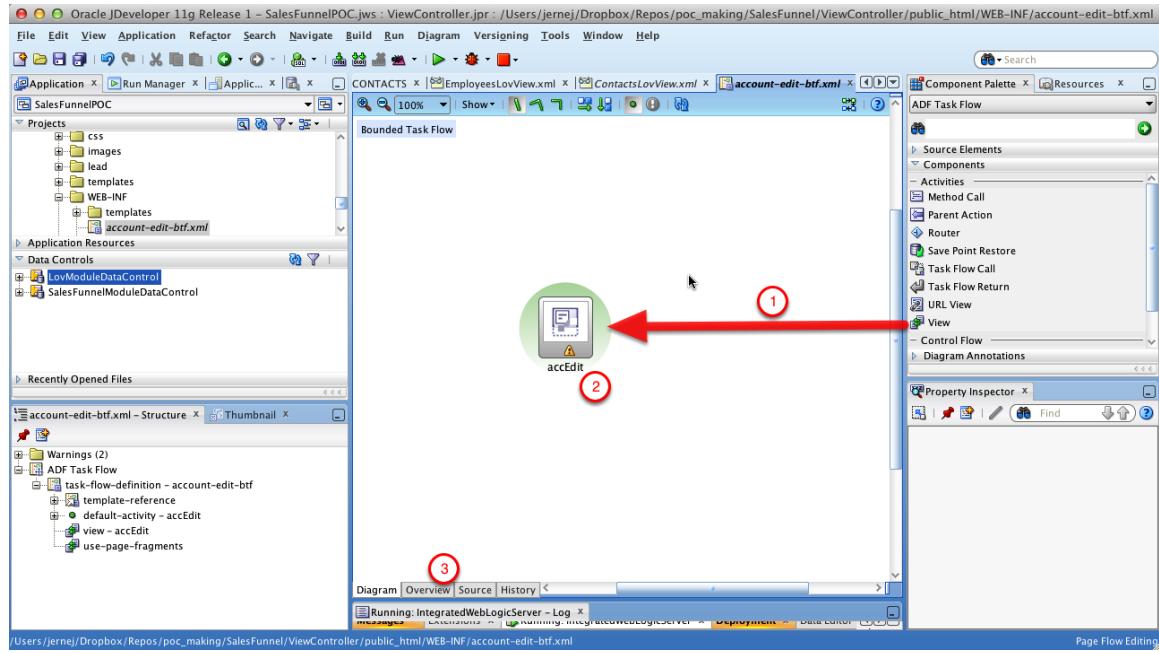
1. Select JSF category
2. Select ADF Task Flow
3. Click OK

Create Task Flow



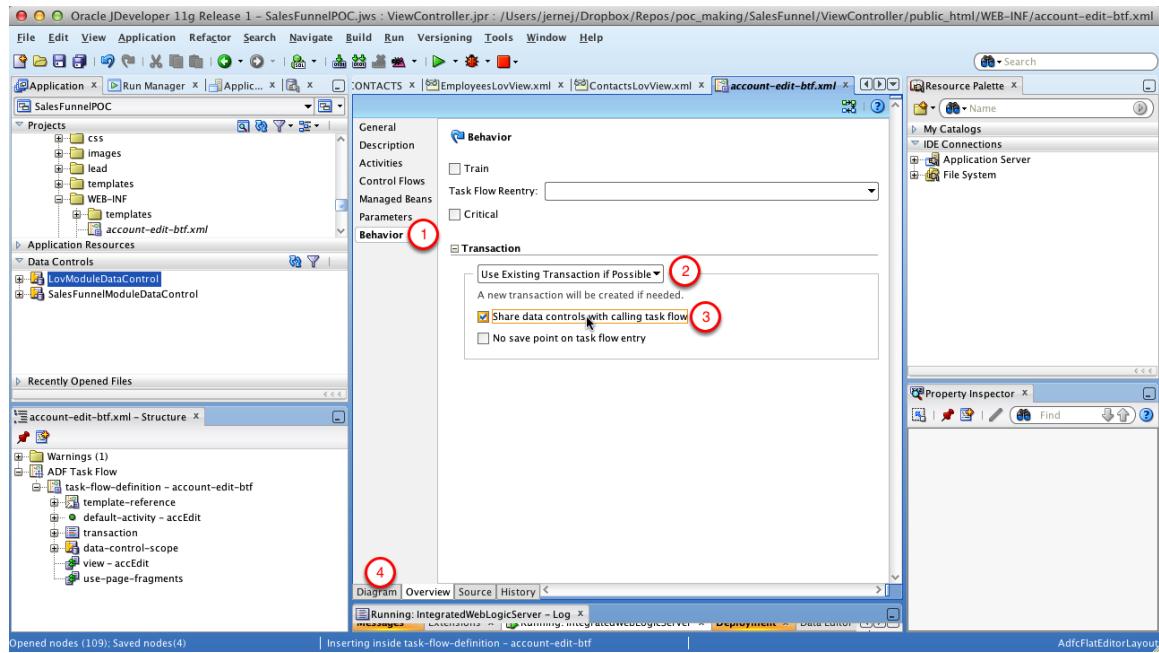
1. Set name to account-edit-btf.xml
2. Base it on sf-btf-template
3. Click OK

Add accEdit View



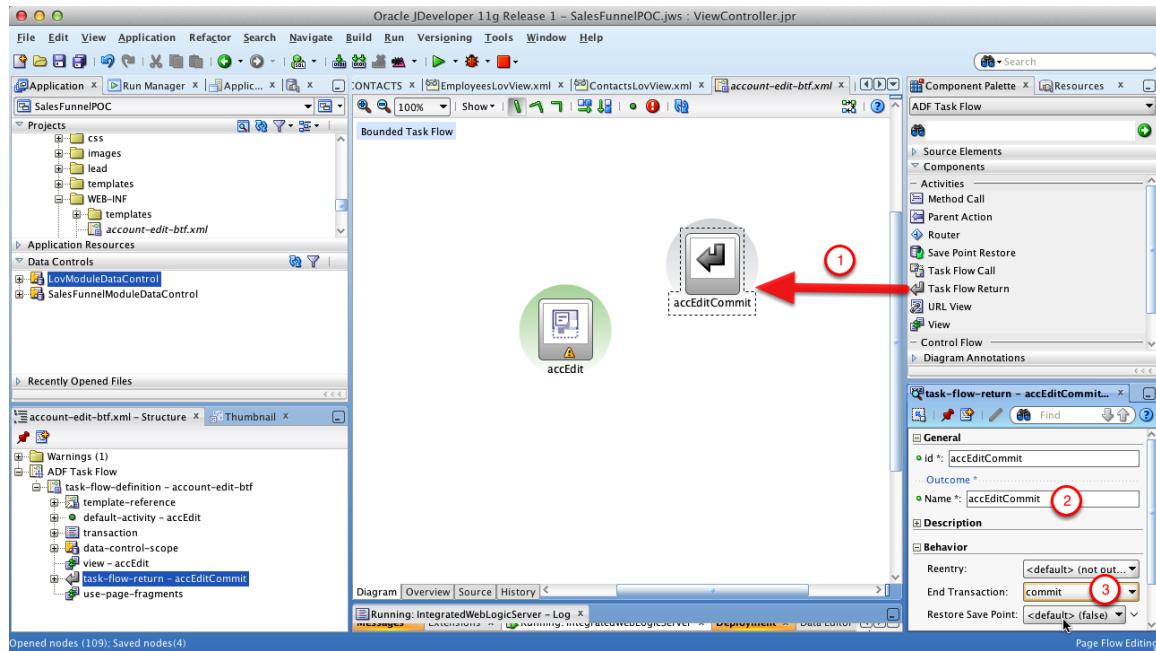
1. Drag and drop View activity on the diagram
2. Set name to accEdit
3. Click Overview tab

Configure acc-edit-btf behavior



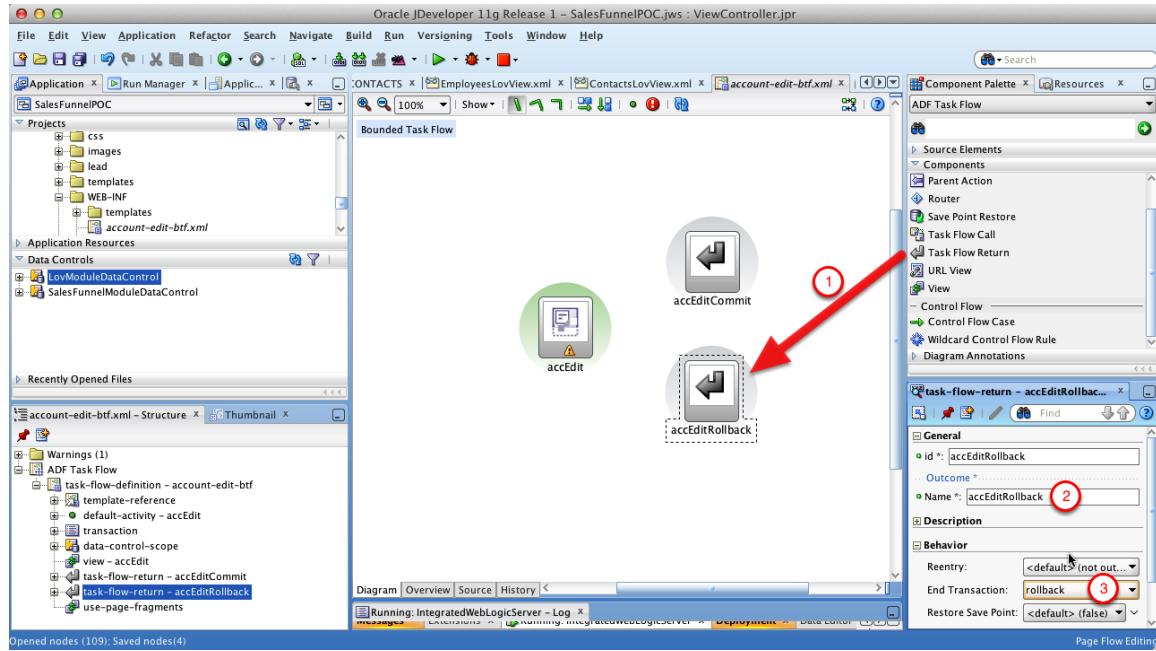
1. Open Behavior tab
2. Set transaction option to Use Existing Transaction If Possible
3. Check Share Data Controls with calling task flow
4. Open Diagram

Add accEditCommit Task Flow Return



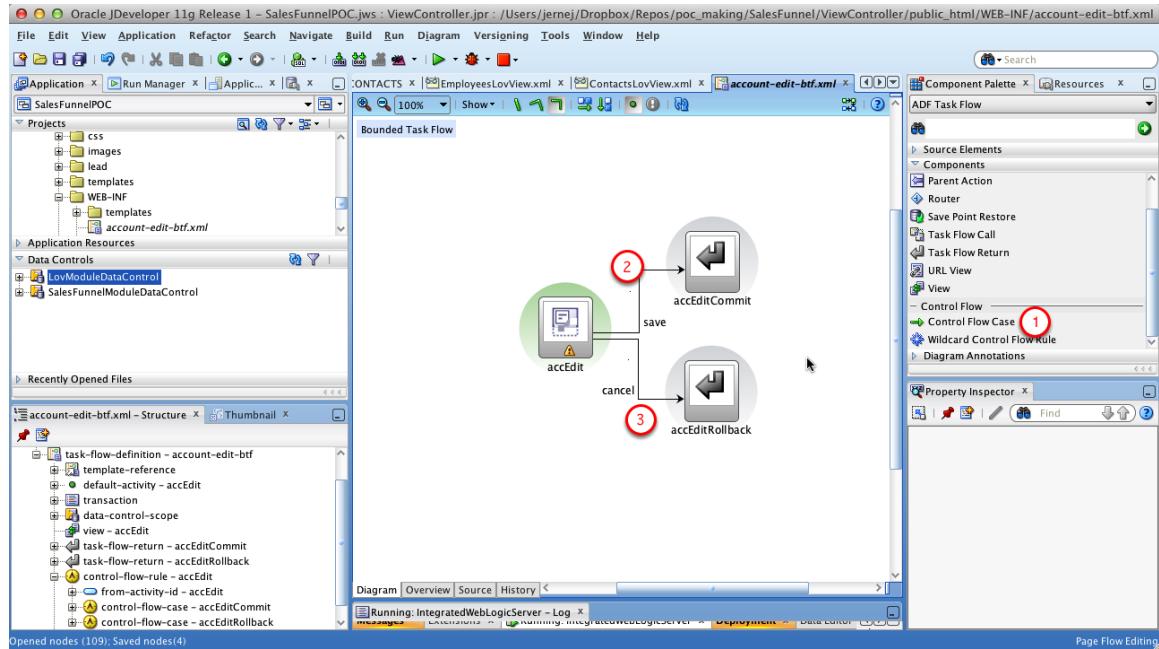
1. Add task flow return activity on the diagram
2. Name it accEditCommit
3. Set End Transaction to commit

Add accEditRollback Task Flow Return



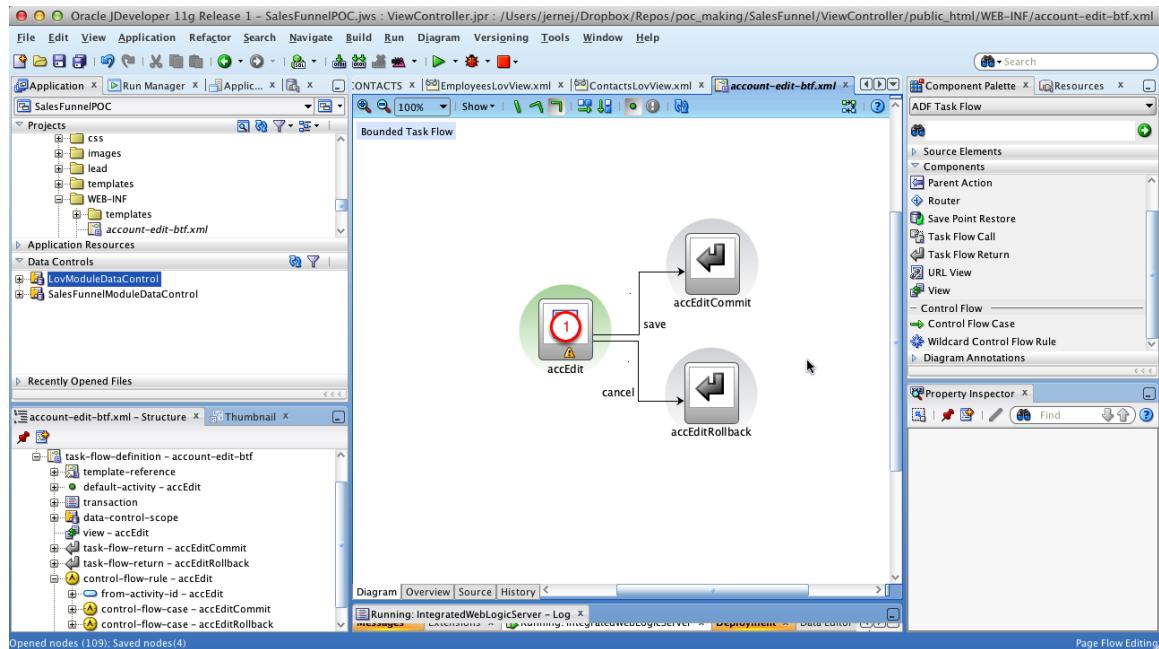
1. Add task flow return activity on the diagram
2. Name it accEditRollback
3. Set End Transaction to rollback

Connect accEdit to task flow returns



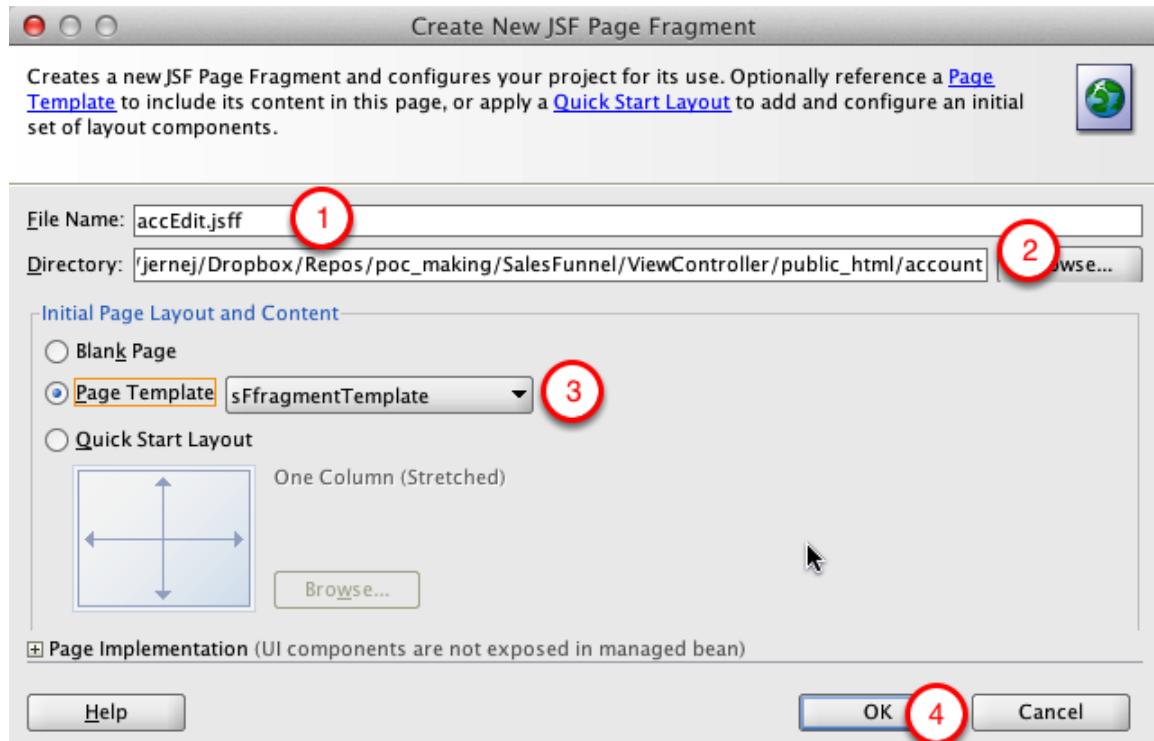
1. Select Control Flow Case
2. Connect accEdit to accEditCommit and name it "save"
3. Connect accEdit to accEditRollback and name it "cancel"

Implementing accEdit view



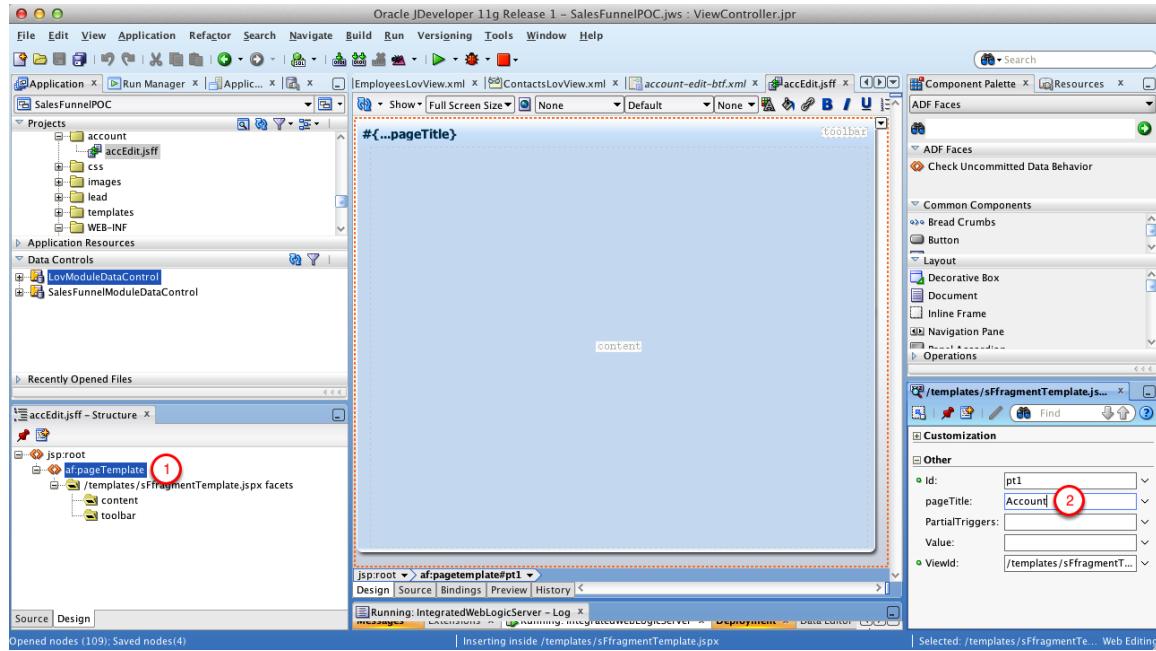
1. Double-click on accEdit view

Create New JSF Page Fragment



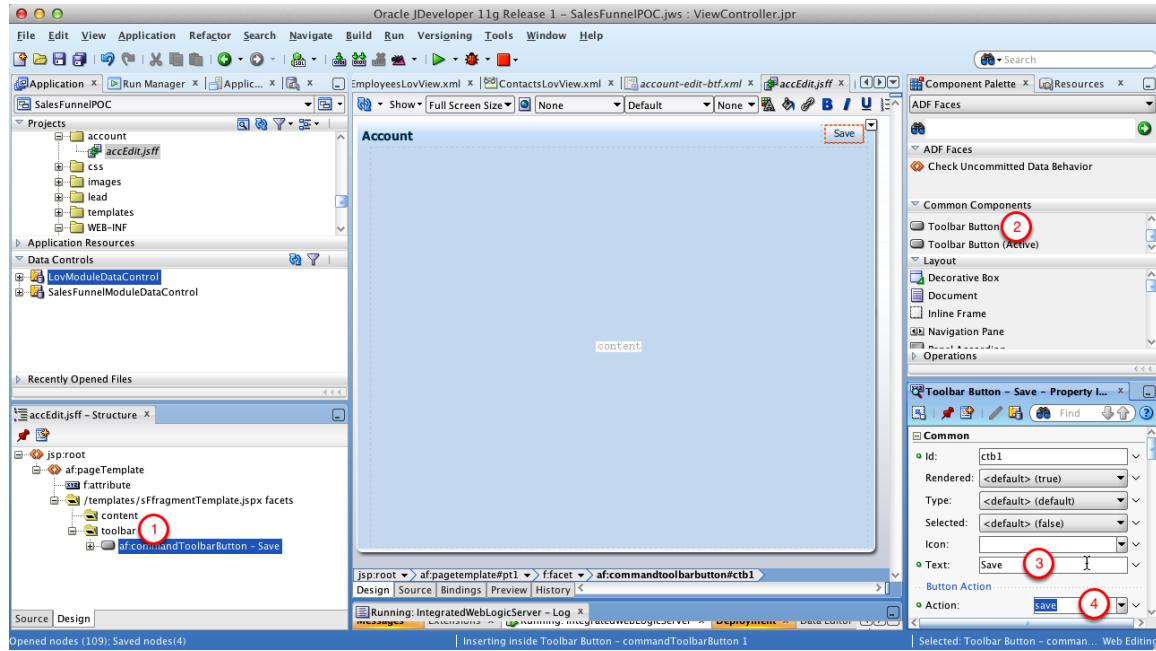
1. Name the file "accEdit.jsff"
2. Append "/account" to directory name
3. Select "sFFragmentTemplate"
4. Click OK

Set template attributes



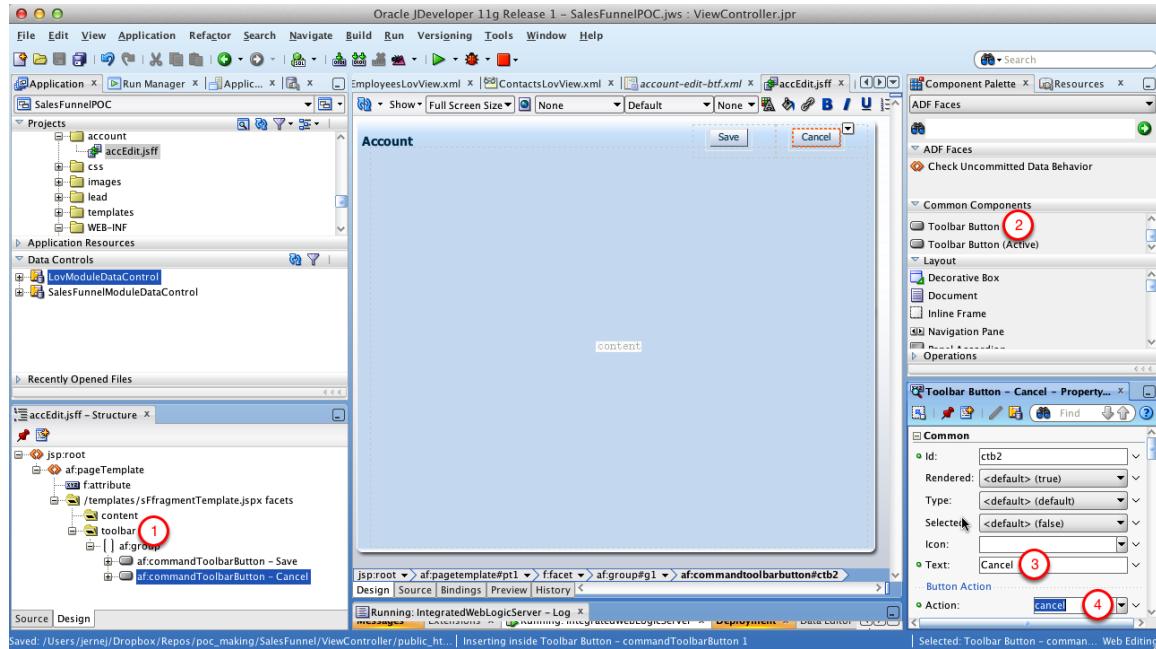
1. Select `af:pageTemplate` in the Structure window
2. Set `pageTitle` property to "Account"

Add Save button



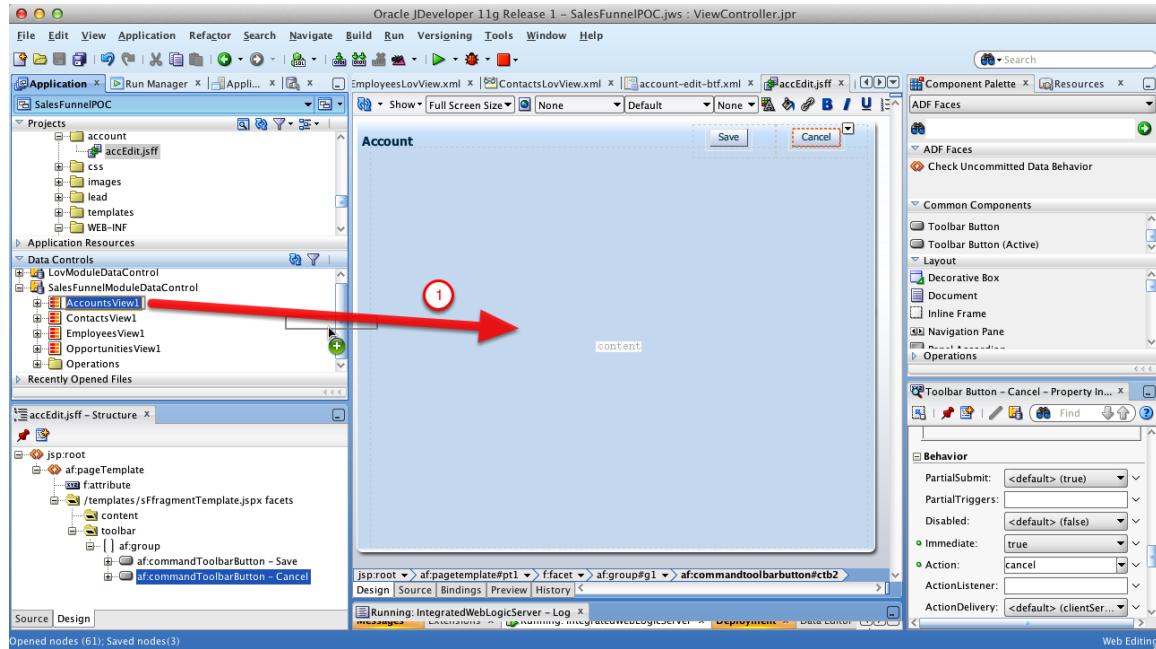
1. In the Structure window, select toolbar facet
2. Find Toolbar Button in the Component Palette and click it, then select the button in the Structure window
3. Set Text property to "Save"
4. Set Action property to "save"

Add Cancel button



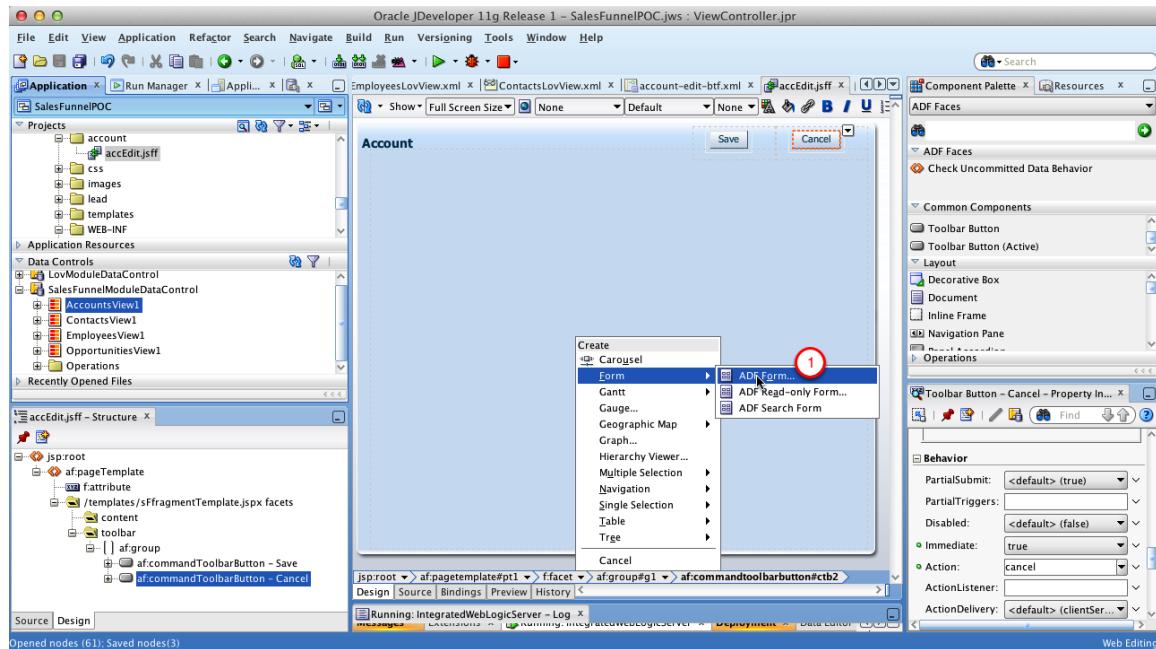
1. In the Structure window, select toolbar facet
2. Find Toolbar Button in the Component Palette and click it, then select the button in the Structure window
3. Set Text property to "Cancel"
4. Set Action property to "cancel"
5. Set Immediate property to true (not visible on the screenshot)

Drag and drop AccountsView on the form



1. In Data Controls, expand SalesFunnelModuleDataControl, then drag and drop AccountsView1 to the content facet

Create ADF Form



1. From the context menu, select Form > ADF Form

Edit Form Fields

Configure the components that you want to display in your form. Note that you can remove or edit the resulting components after you click OK. You can also add more components directly to the layout later.

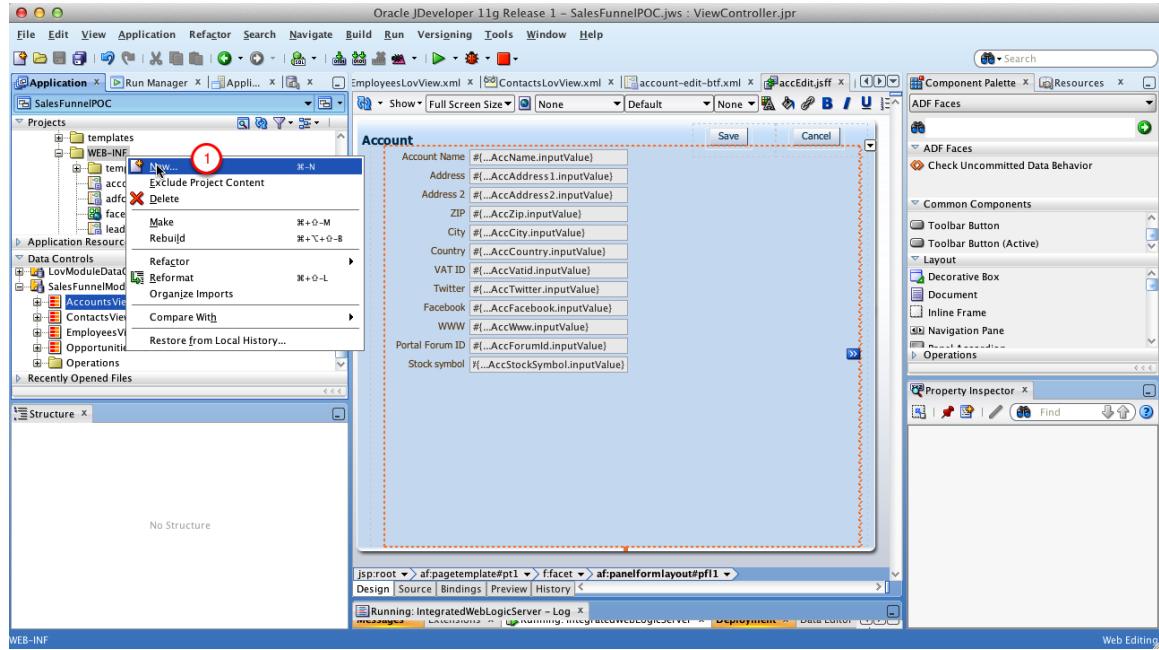
Fields:		
Display Label	Value Binding	Component To Use
xyz <default>	AccName	ADF Input Text w/ Label
xyz <default>	AccAddress1	ADF Input Text w/ Label
xyz <default>	AccAddress2	ADF Input Text w/ Label
xyz <default>	AccZip	ADF Input Text w/ Label
xyz <default>	AccCity	ADF Input Text w/ Label
xyz <default>	AccCountry	ADF Input Text w/ Label
xyz <default>	AccVatid	ADF Input Text w/ Label
xyz <default>	AccTwitter	ADF Input Text w/ Label
xyz <default>	AccFacebook	ADF Input Text w/ Label
xyz <default>	AccWww	ADF Input Text w/ Label
xyz <default>	AccForumId	ADF Input Text w/ Label
xyz <default>	AccStockSymbol	ADF Input Text w/ Label

Include Navigation Controls
 Include Submit Button

1. Use the defaults and just click OK button

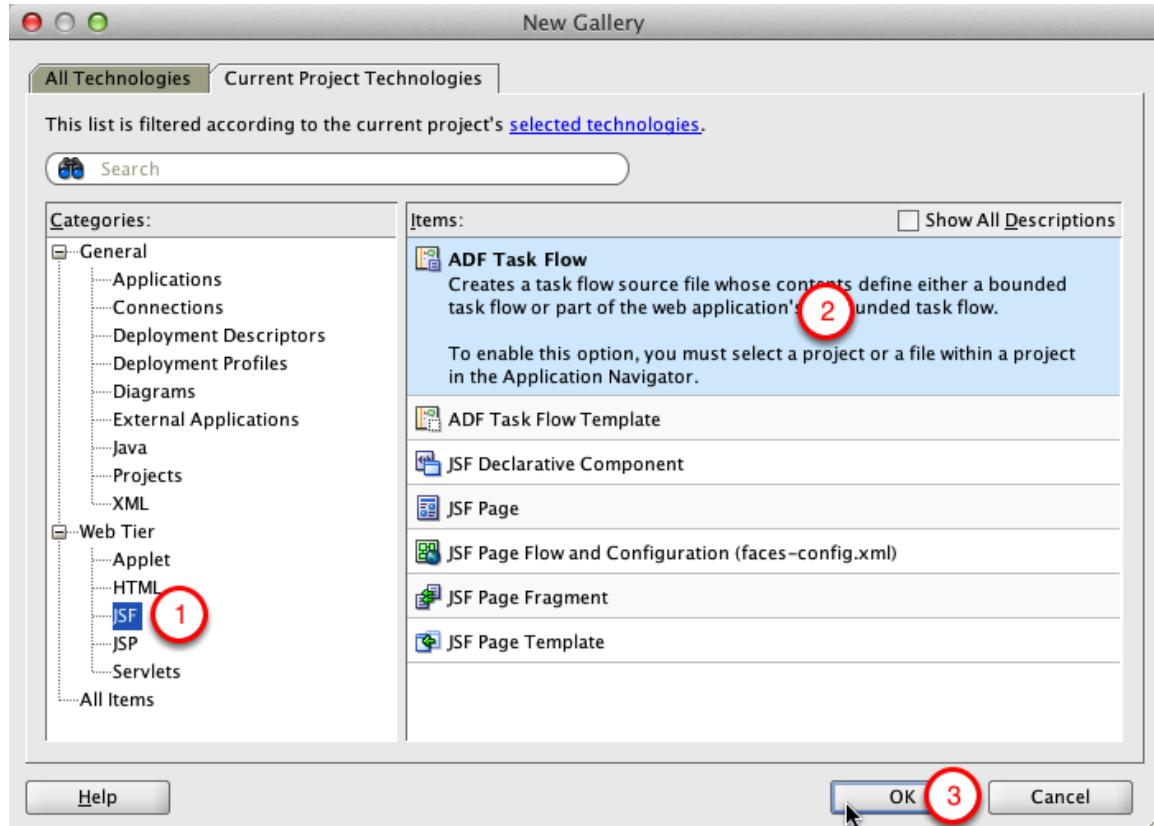
26. Account Create BTF

Create New Task Flow



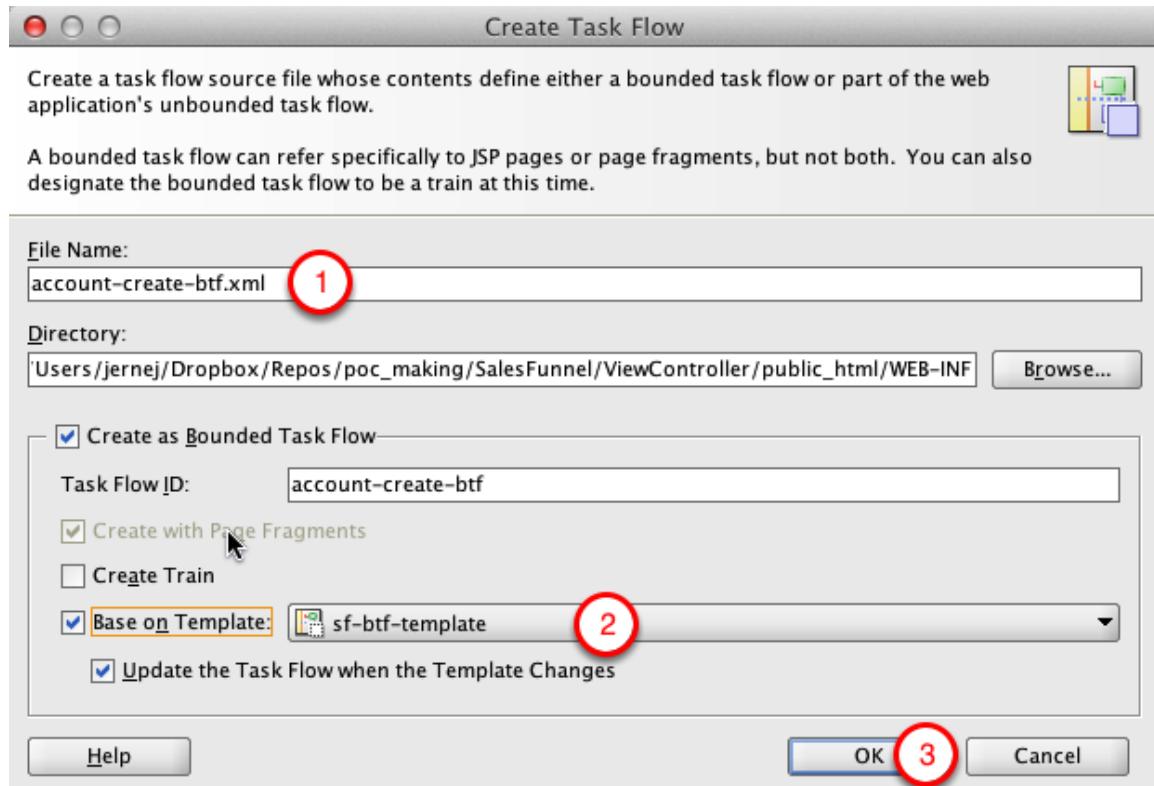
1. Right-click WEB-INF folder and select New from the context menu

New Gallery



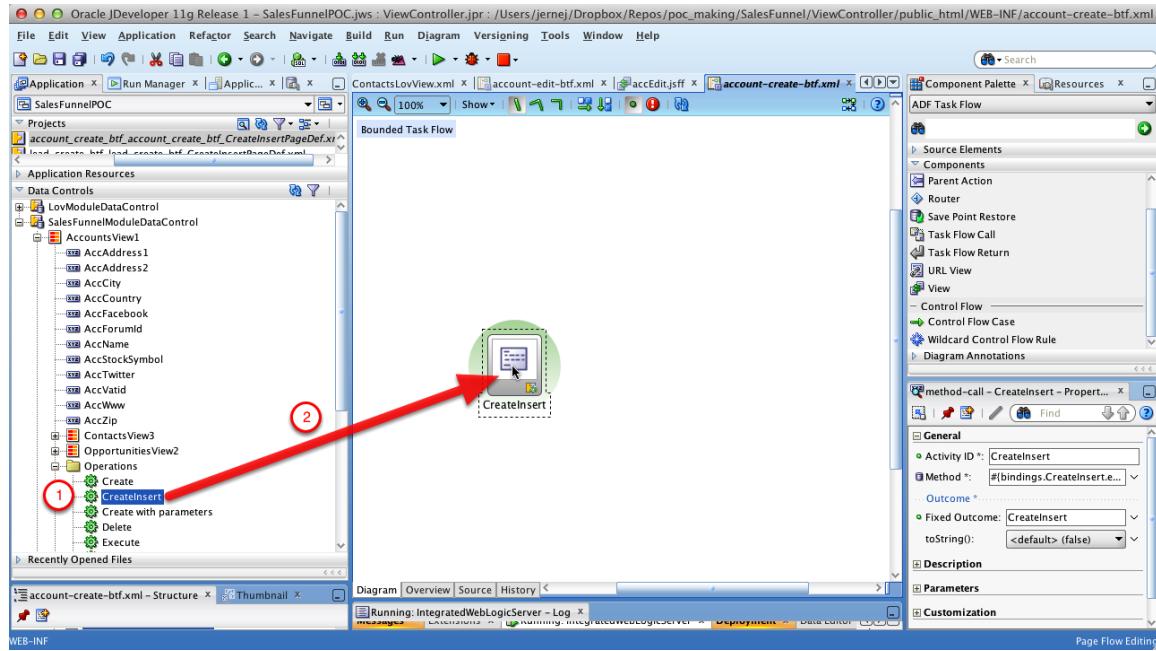
1. Select Web Tier > JSF Category
2. Select ADF Task Flow
3. Click OK

Create Task Flow



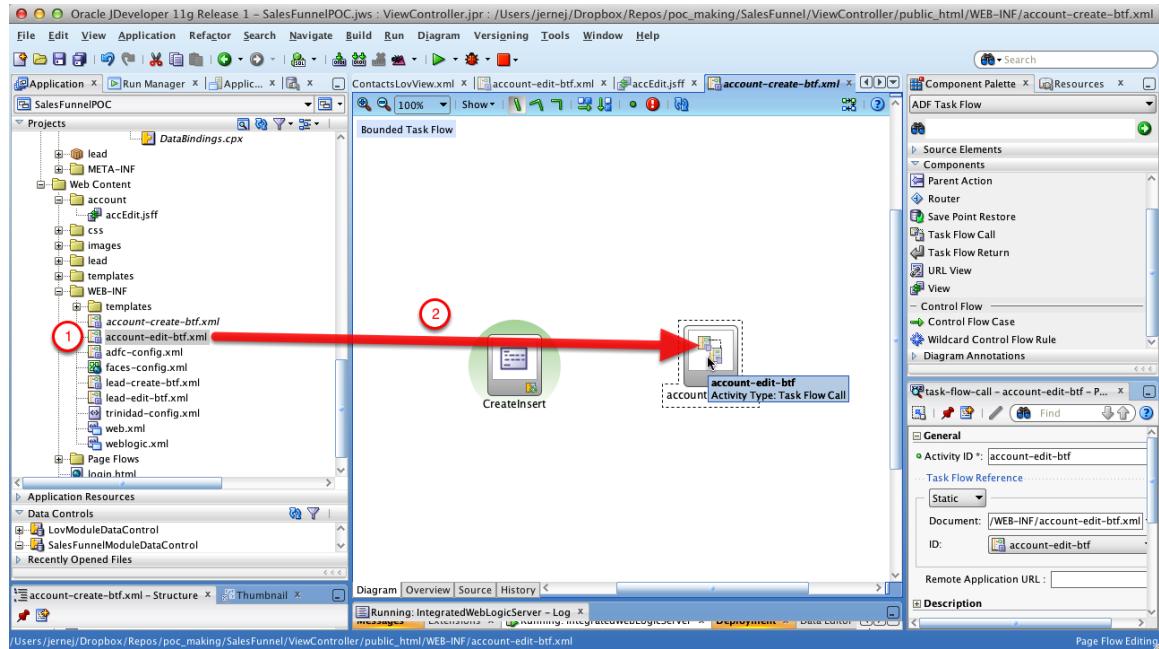
1. Name the file "account-create-btf.xml"
2. Select sf-btf-template
3. Click OK

Drag and Drop AccountsView CreateInsert operation



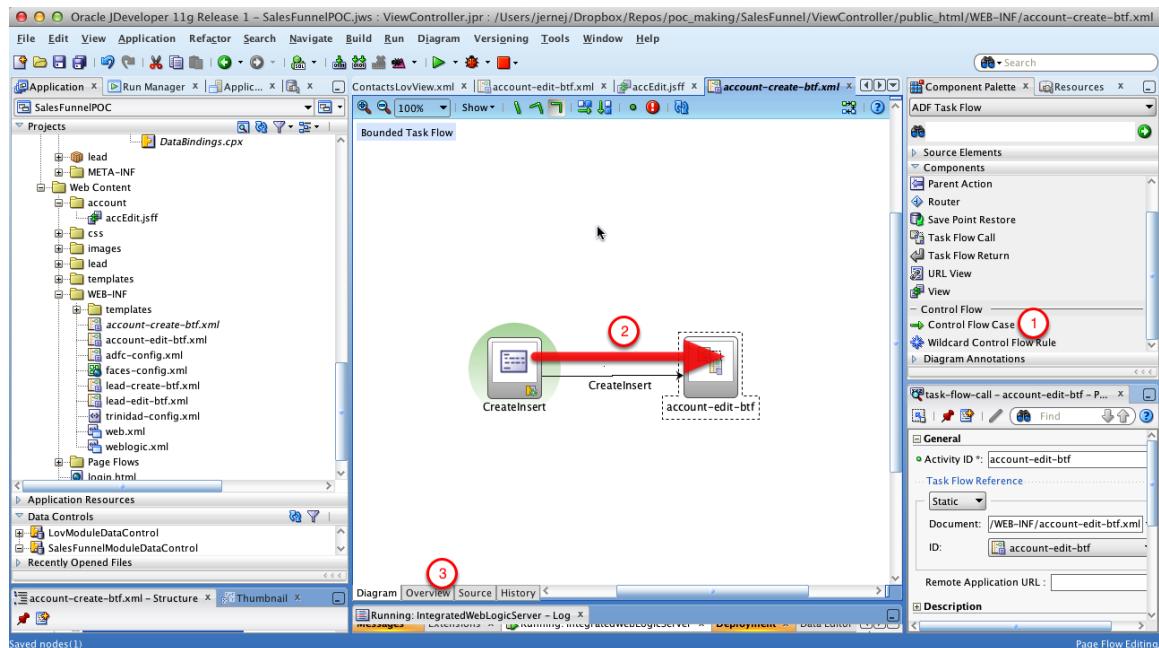
1. In Data Controls, expand SalesFunnelModuleDataControl > AccountsView1 > Operations
2. Drag and drop CreateInsert operation to the task flow diagram

Add account-edit-btf task flow



1. In the Projects window locate account-edit-btf
2. Drag and drop it to the task flow diagram

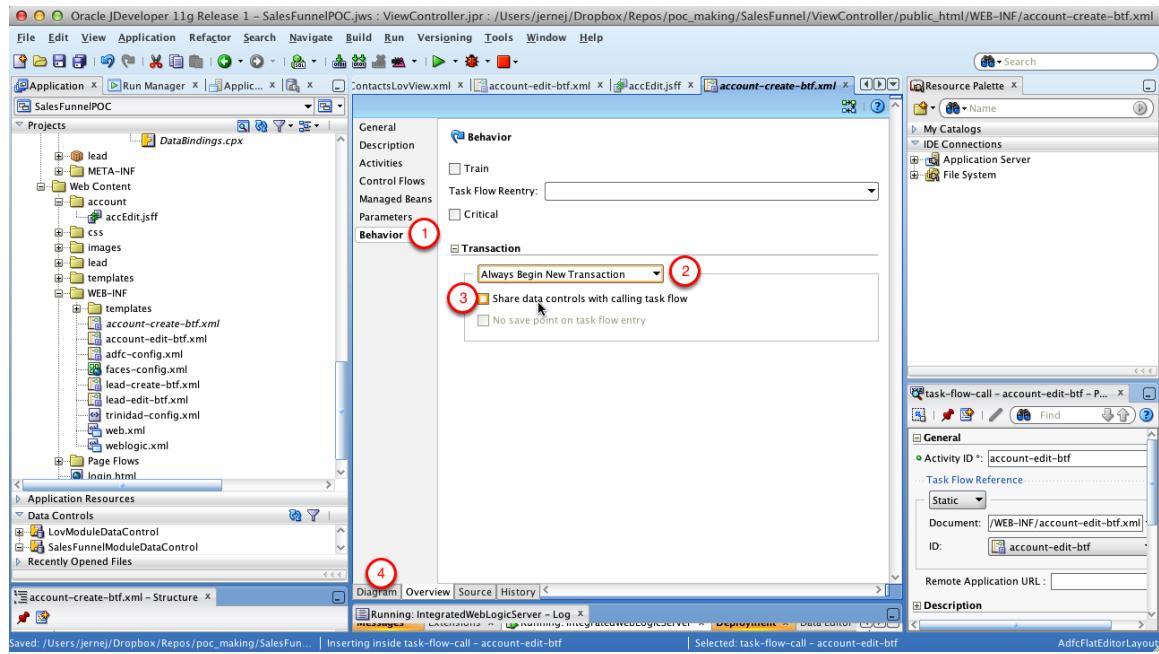
Connect CreateInsert to account-edit-btf



1. Select Control Flow Case in the Component Palette
2. Click CreateInsert action and connect it to account-edit-btf

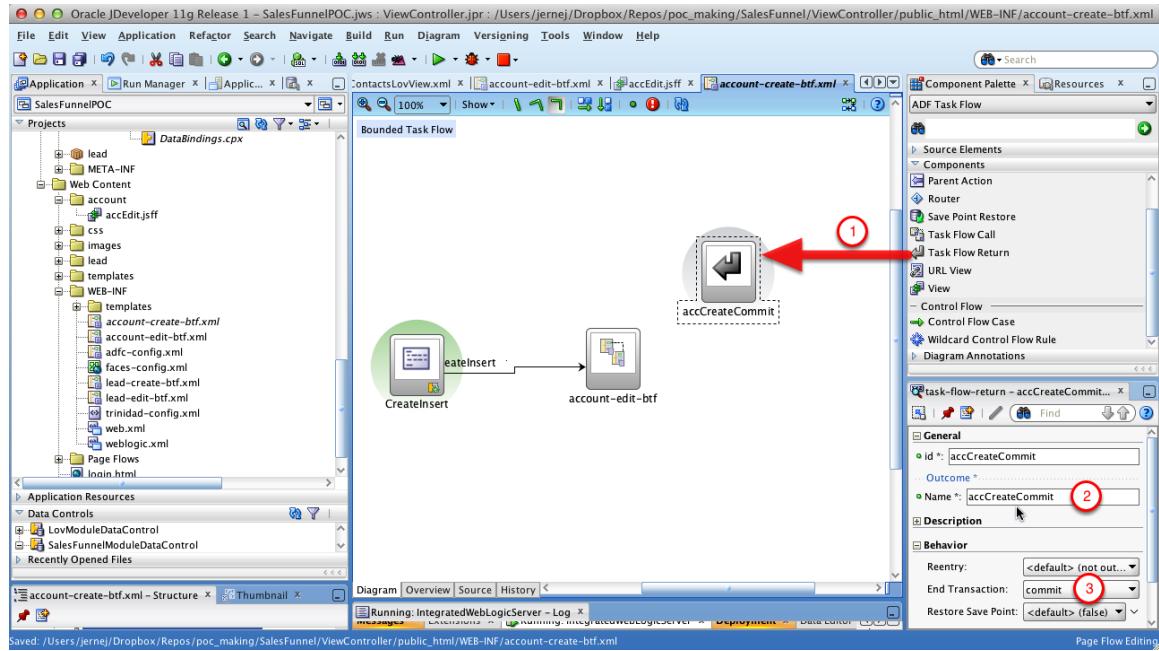
3. Open Overview tab

Set account-create-btf behavior



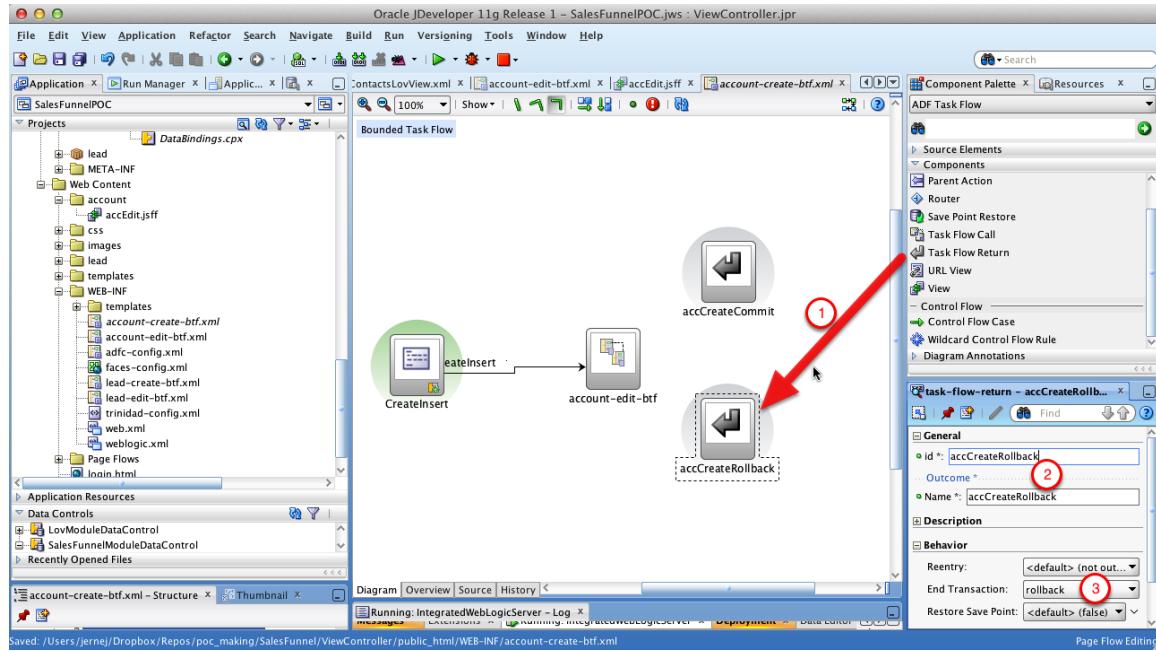
1. Select Behavior tab
2. Select Always Begin New Transaction option
3. Uncheck Share data controls checkbox
4. Return to diagram

Add accCreateCommit Task Flow Return



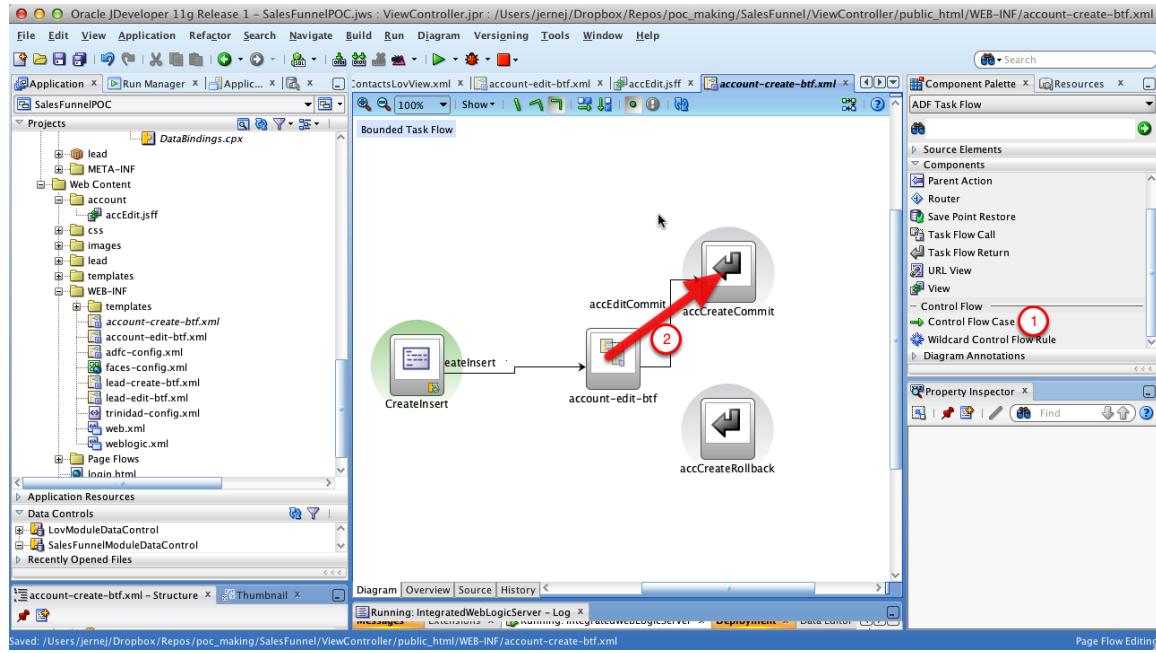
1. Drag and drop Task Flow Return action from Components Palette to the diagram
2. set id and name properties to "accCreateCommit"
3. Set End Transaction property to commit

Add accCreateRollback Task Flow Return



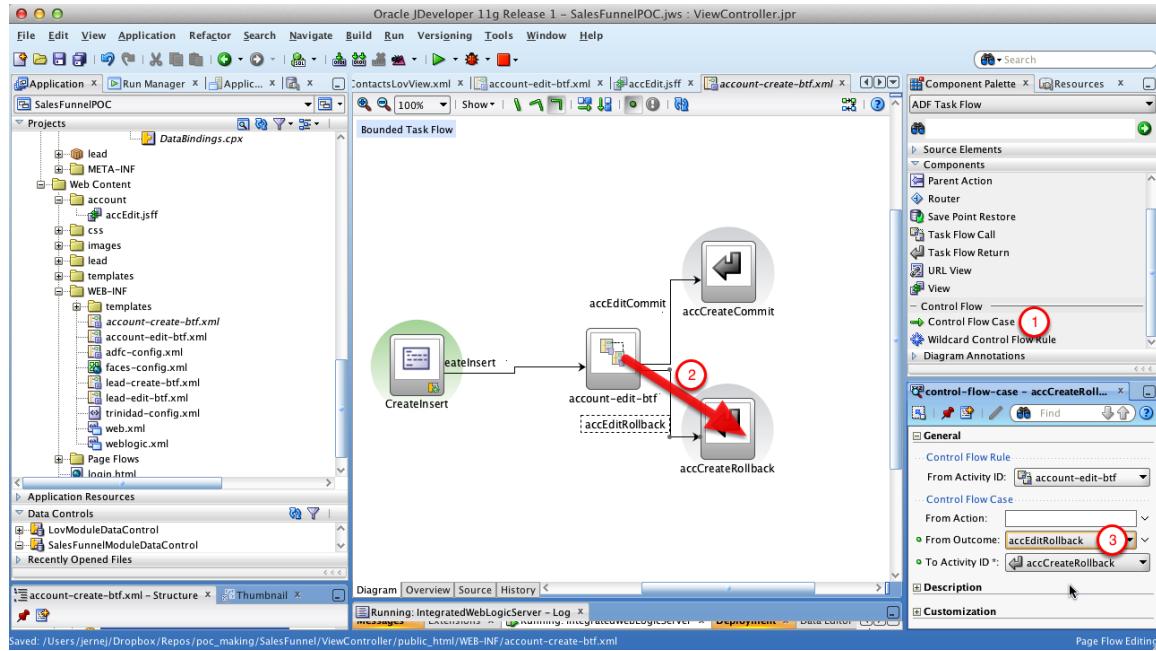
1. Drag and drop Task Flow Return action from Components Palette to the diagram
2. Set id and name properties to "accCreateRollback"
3. Set End Transaction property to rollback

Connect account-edit-btf to accCreateCommit



1. Select Control Flow Case in the Component Palette
2. Connect account-edit-btf to accCreateCommit action
3. Make sure the outcome is accEditCommit

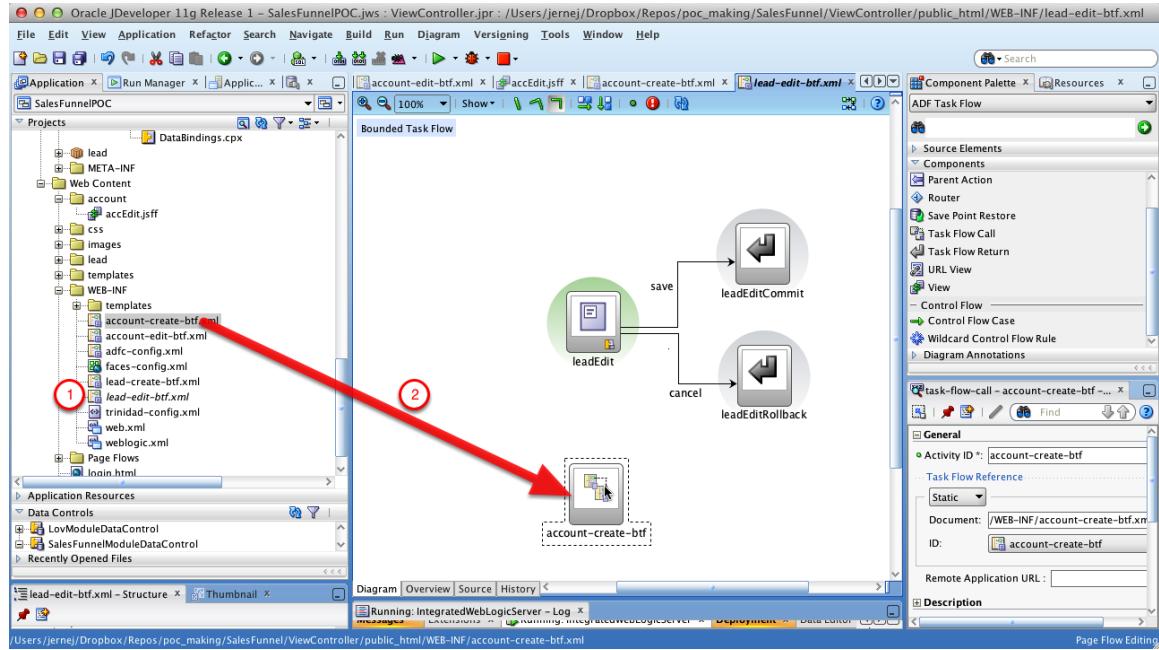
Connect account-edit-btf to accCreateRollback



1. Select Control Flow Case in the Component Palette
2. Connect account-edit-btf to accCreateRollback action
3. Set the from outcome property to accEditRollback

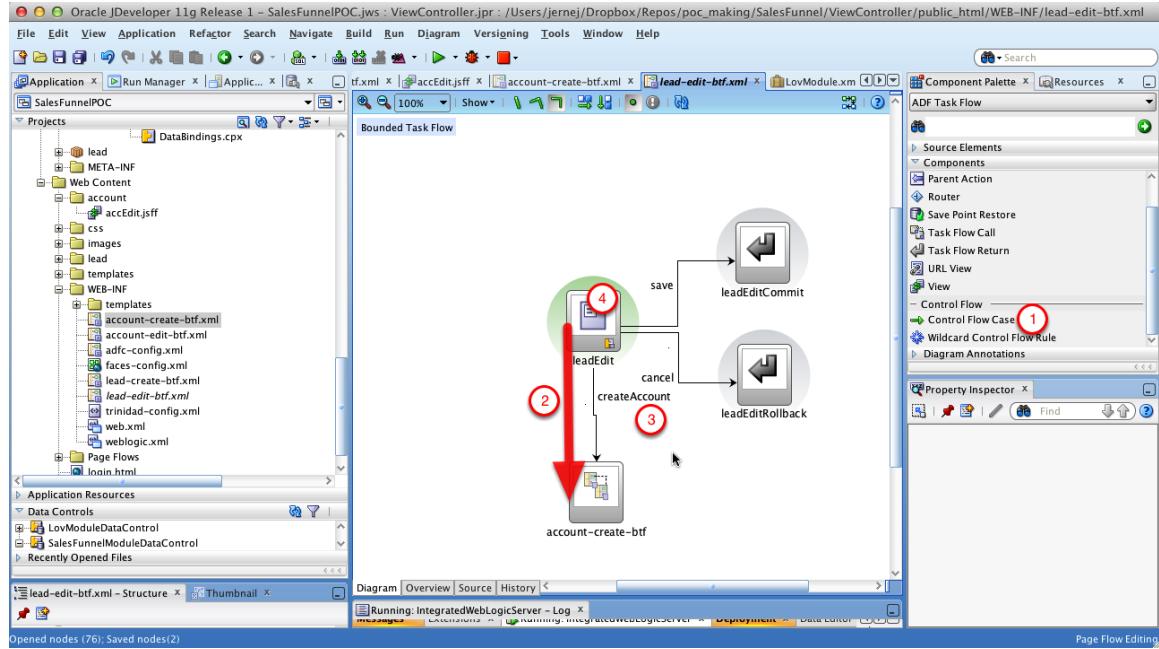
27. Adding Account Create to Lead Edit BTF

Add account-create-btf to lead-edit-btf



1. Open lead-edit-btf
2. Drag and drop account-create-btf to the diagram

Connect leadEdit to account-create-btf



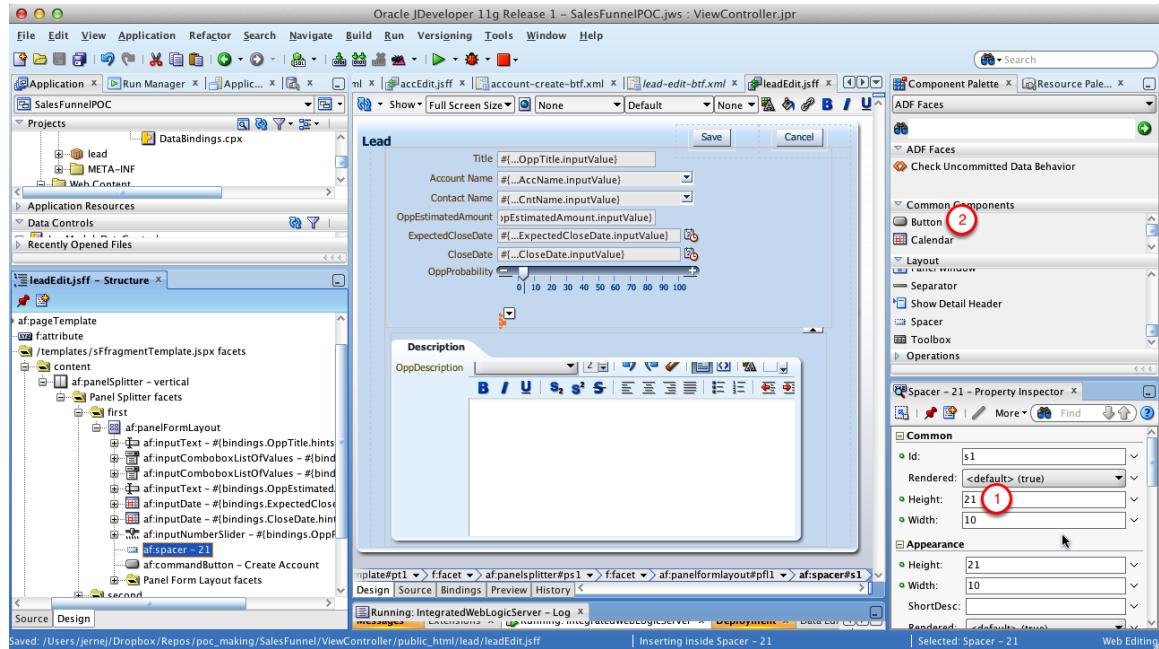
1. Select Control Flow Case in the Component Palette
2. Connect leadEdit view to account-create-btf
3. Name the action "createAccount"
4. Double click leadEdit view to open it

Change Lead Form layout

The screenshot shows the Oracle JDeveloper interface with the 'leadEdit.jsff' file open in the center. The Design tab is active. In the Component Palette on the right, the 'Spacer' component is selected (circled in red). The form layout in the center contains several input fields and a slider.

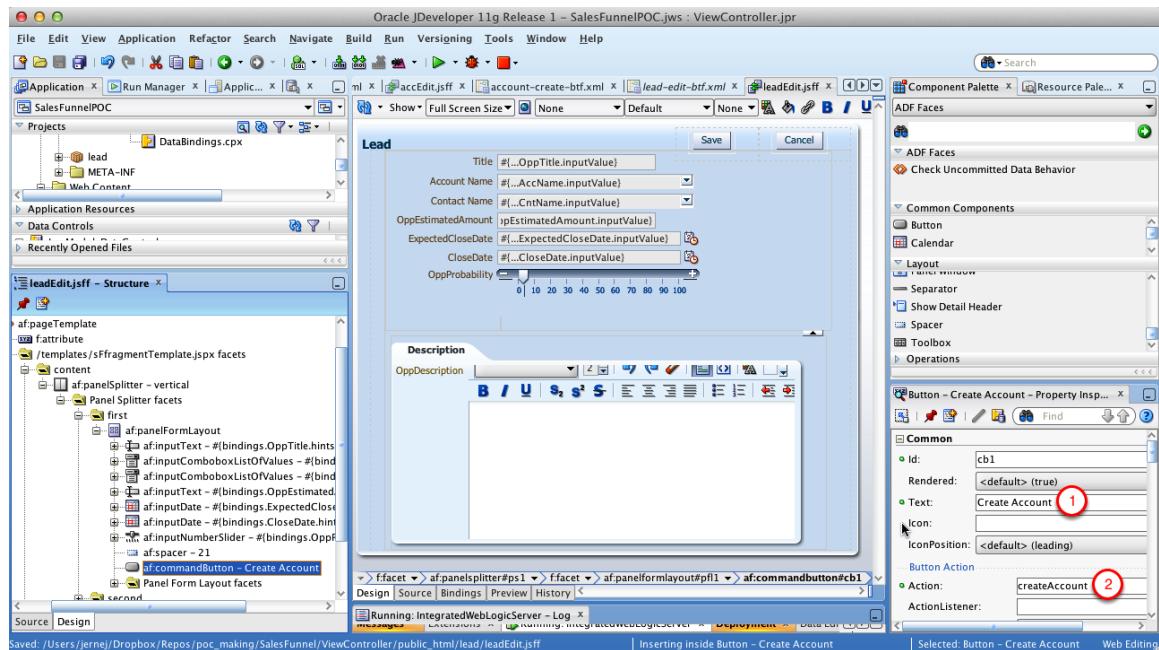
1. In the Structure window, find and select af:panelFormLayout
2. Click Spacer component in the Component Palette to append it to the form

Add Create Account Button



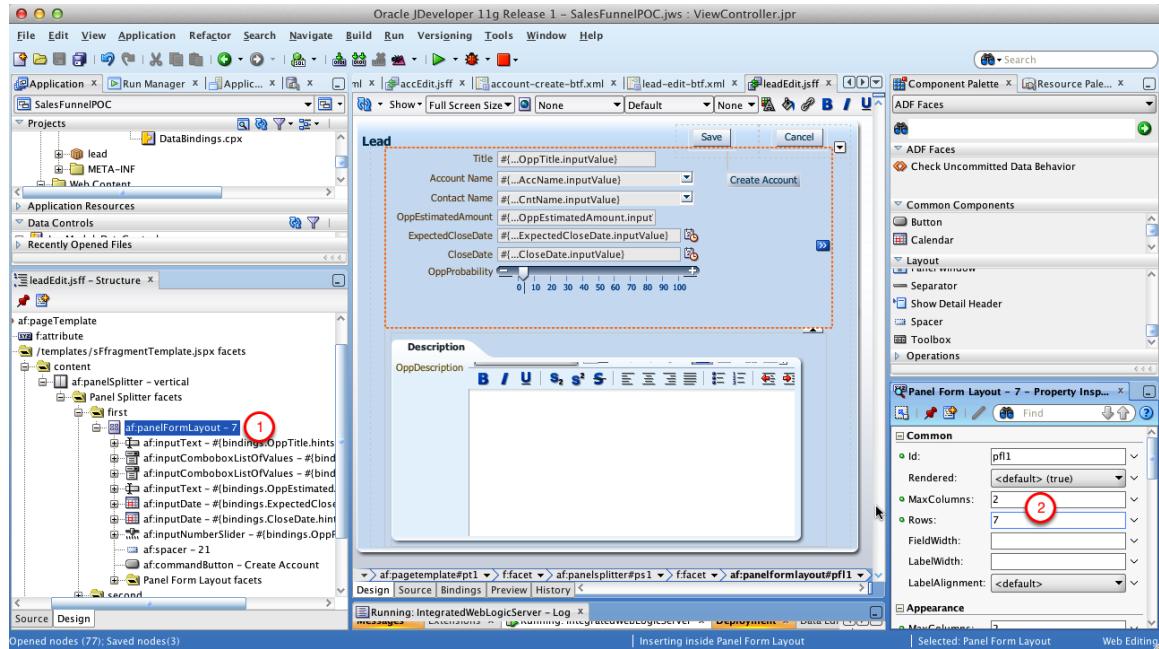
1. With the spacer selected, set Height property to 21
2. Click the Button component in Component Palette to append a button to the form

Set Create Account button's properties



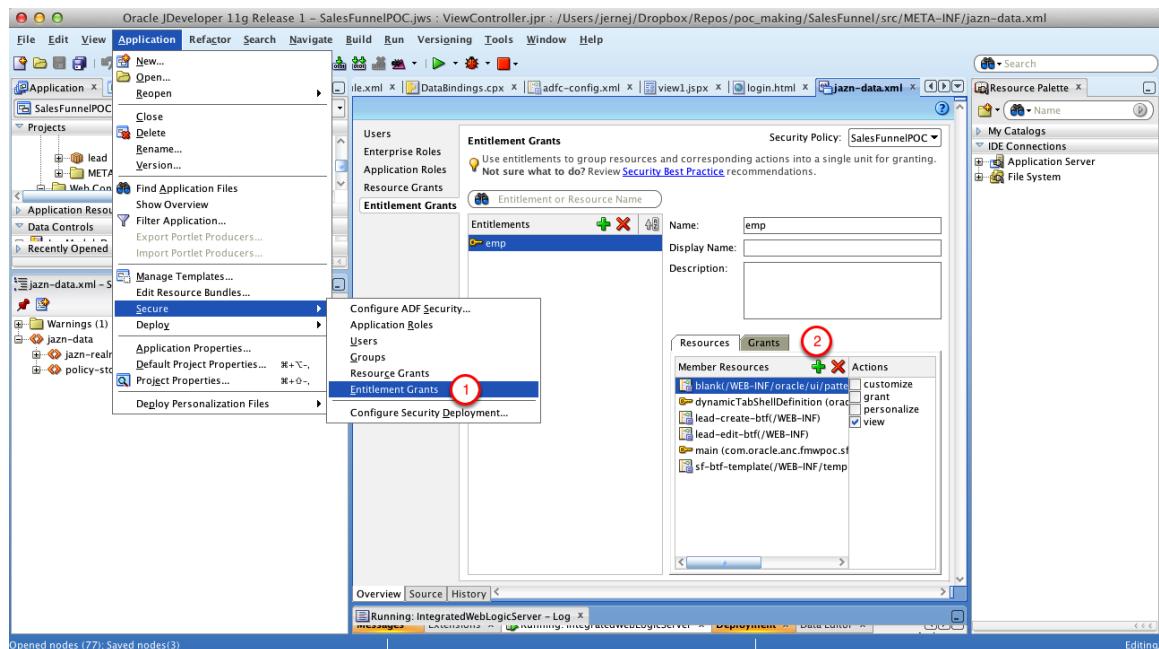
1. Select af:commandButton
2. Set Text property to Create Account
3. Set Action property to createAccount

Change Lead form's properties



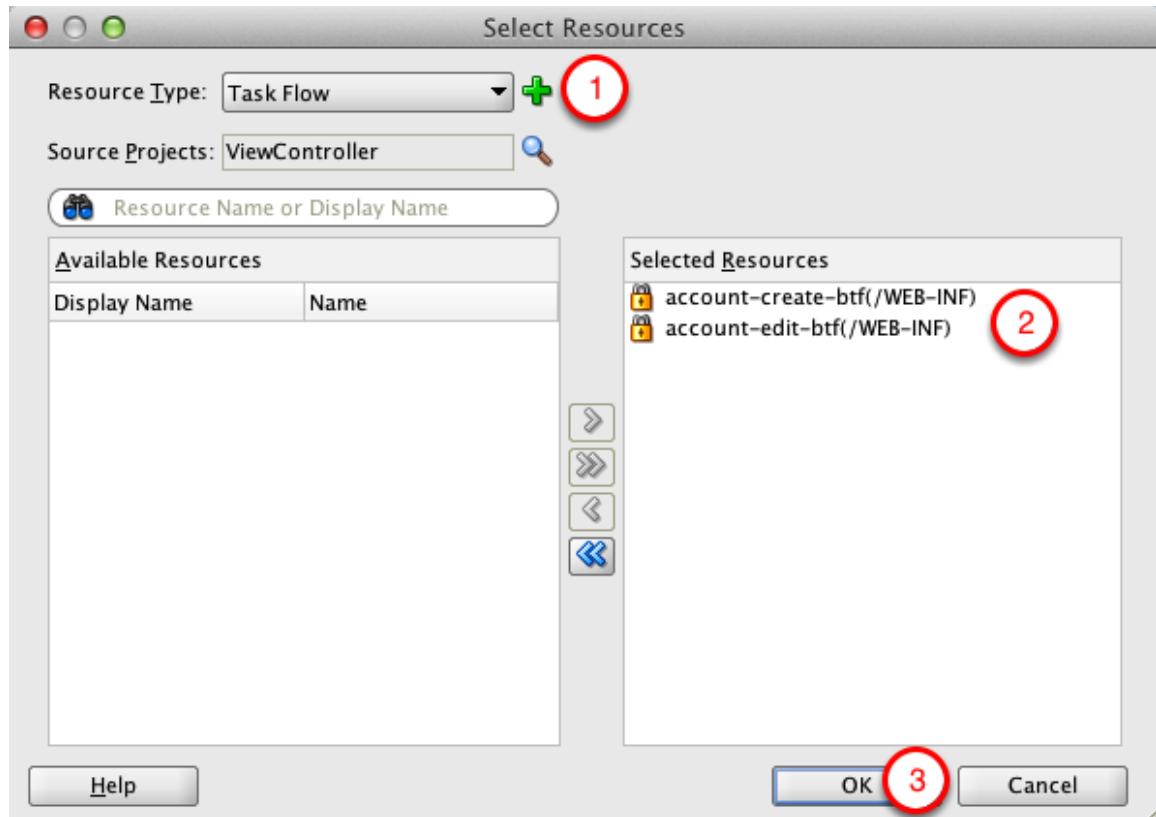
1. Select af:panelFormLayout
2. Set MaxColumns property to 2 and Rows to 7

Add Security Grants



1. In the menu, click Application > Secure > Entitlement Grants
2. Click the plus when it opens

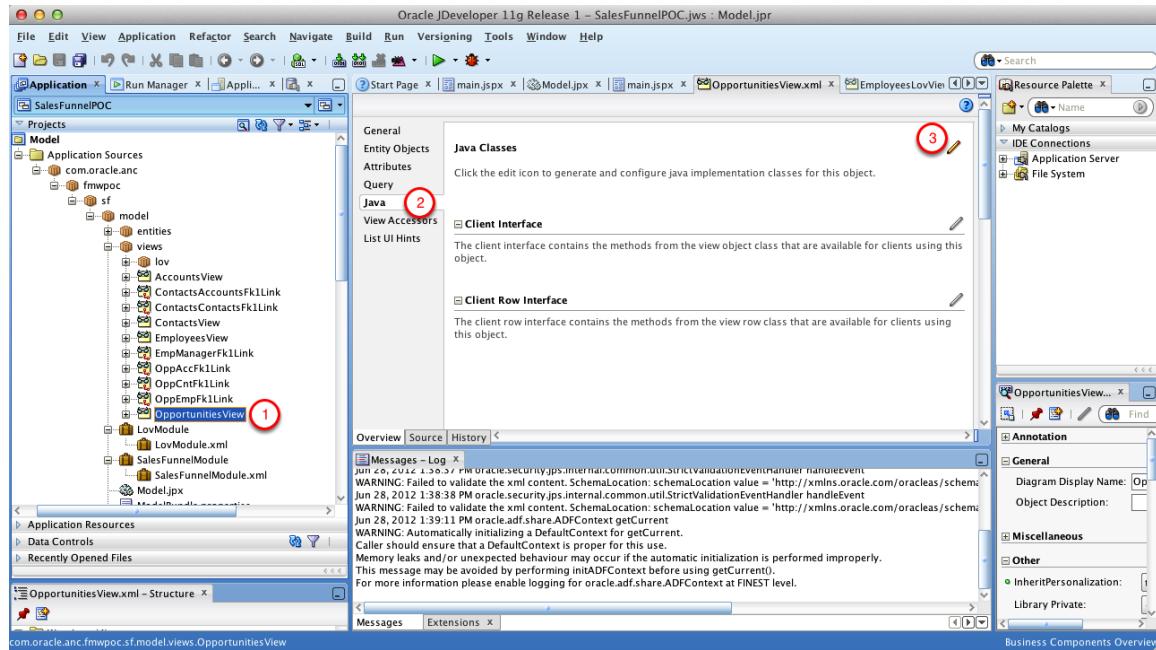
Select Resources



1. Select Task Flow as Resource Type
2. Slide account-create-btf and account-edit-btf to the right
3. Click OK

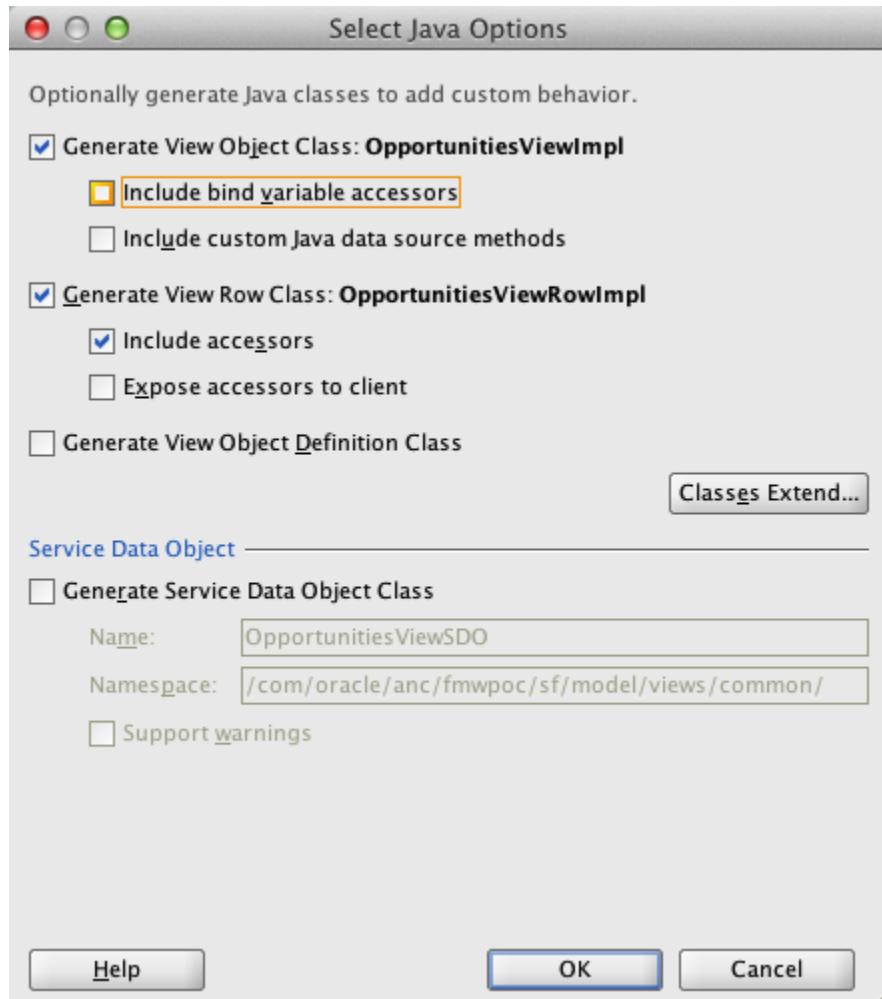
28. Refresh Accounts LOV after insert

Add OpportunitiesView Java class

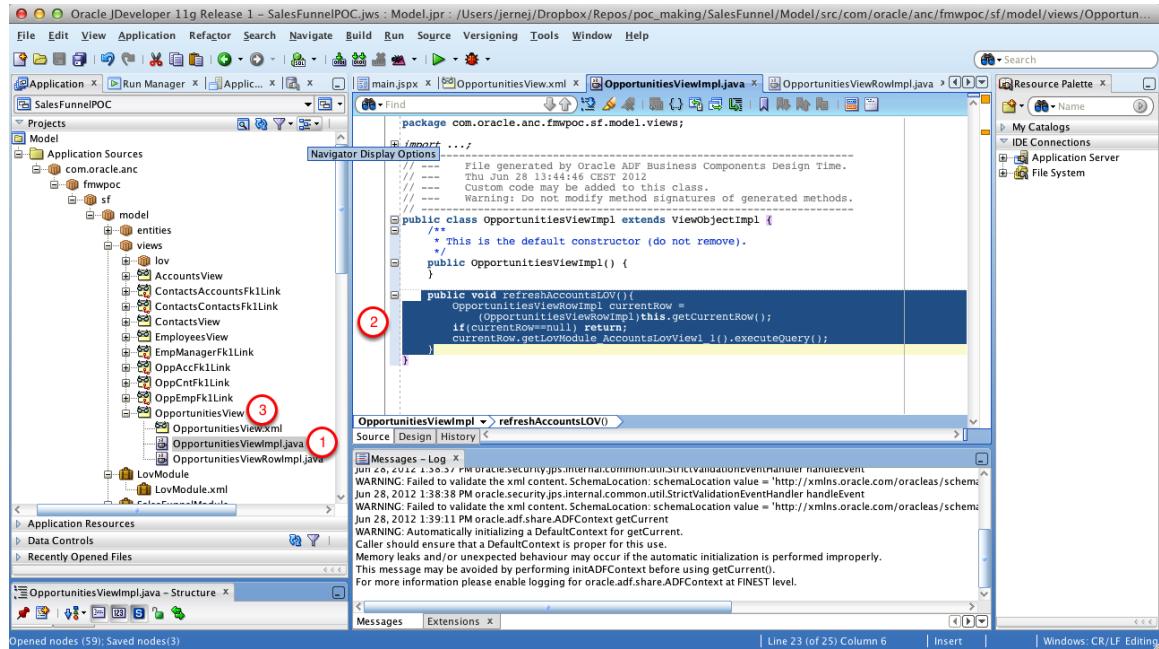


1. In the Model project, locate OpportunitiesView and open it
2. Select Java tab
3. Click edit icon next to Java Classes

Select Java Options



Paste refreshAccountsLOV method



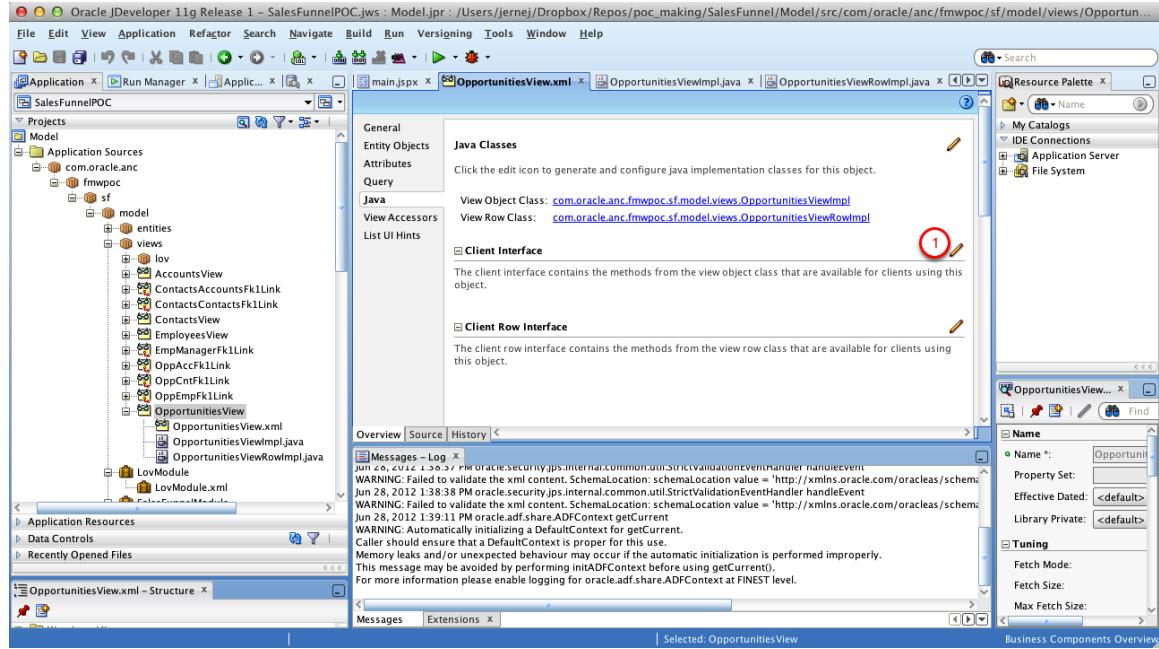
1. Open OpportunitiesViewImpl.java
2. Append the below code at the end of the class:
3. Open OpportunitiesView

```

public void refreshAccountsLOV(){
    OpportunitiesViewRowImpl currentRow =
        (OpportunitiesViewRowImpl)this.getCurrentRow();
    currentRow.getLovModule AccountsLovView1_1().executeQuery();
}

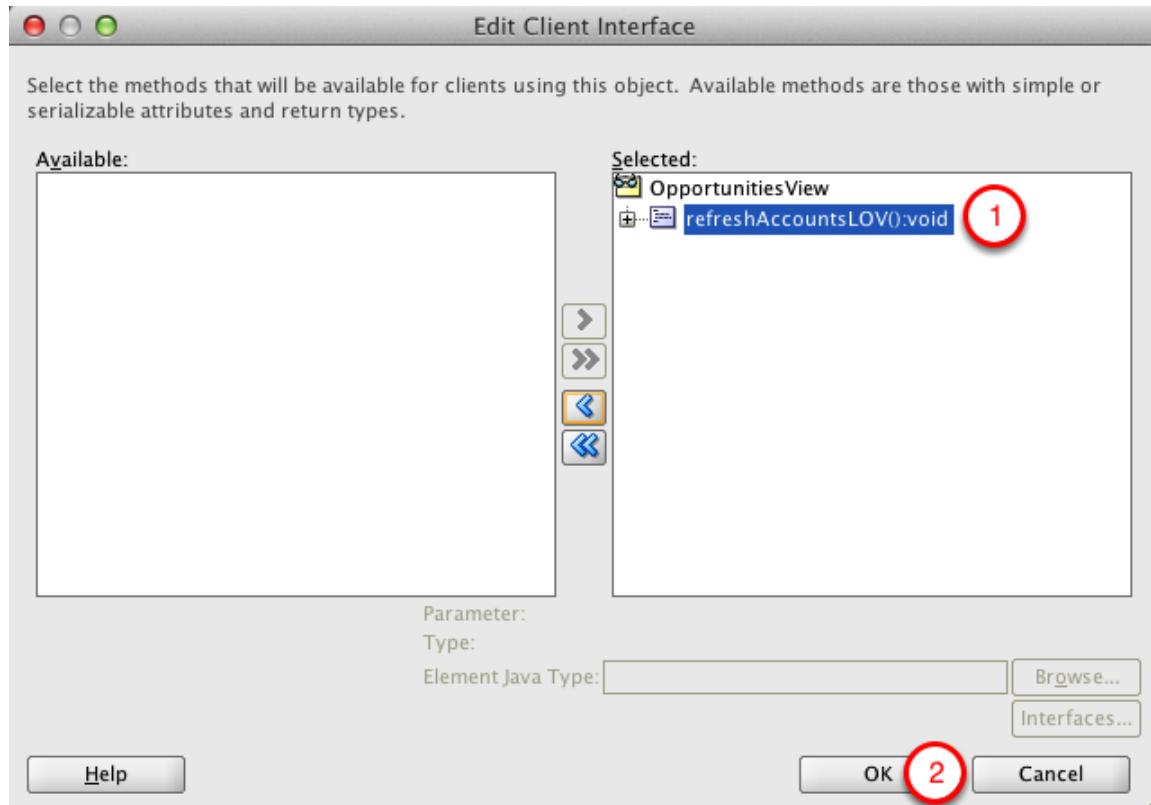
```

Add Client Interface



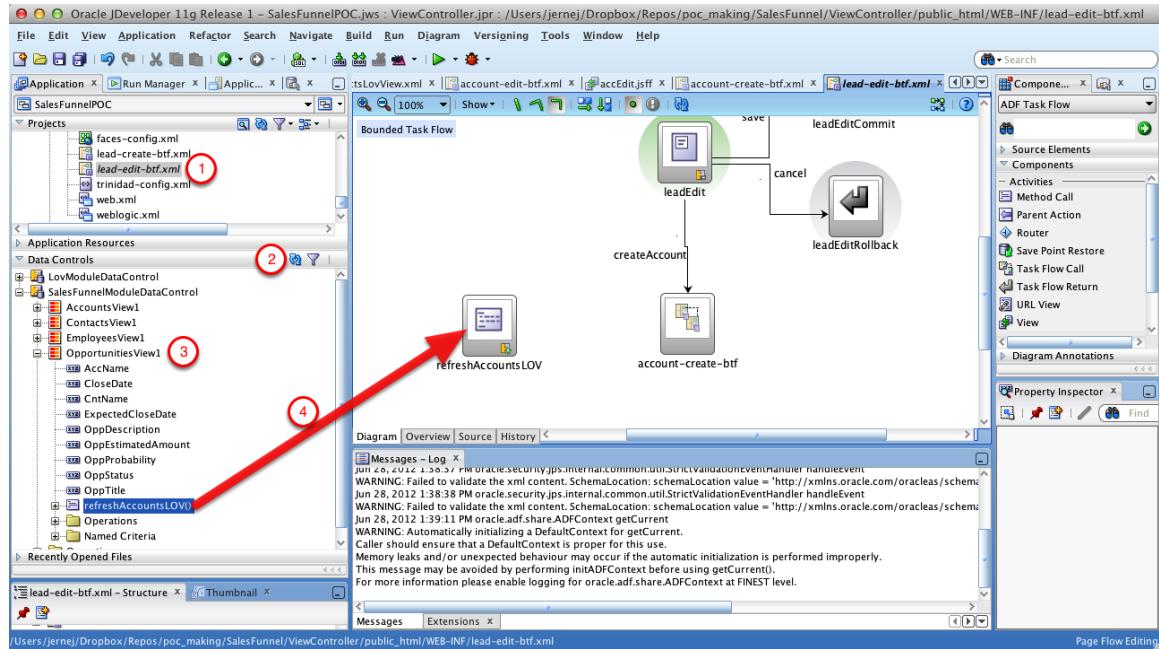
1. Click edit icon next to Client Interface

Edit Client Interface



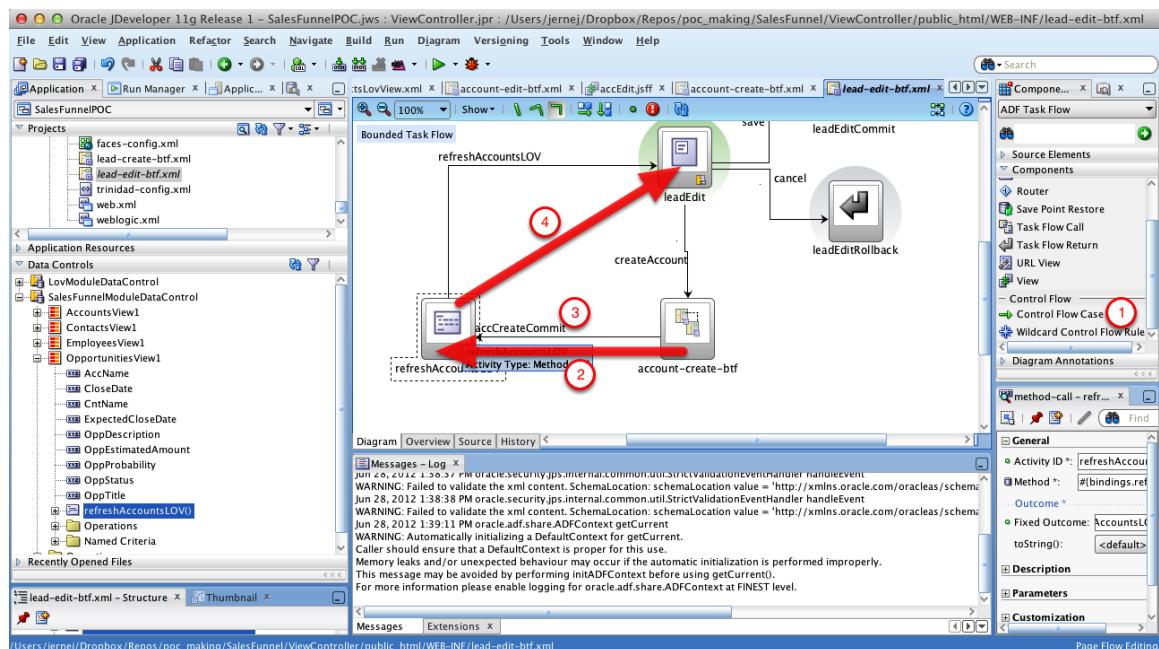
1. Slide refreshAccountsLOV to the right
2. Click OK

Add refreshAccountsLOV method to lead-edit-btf



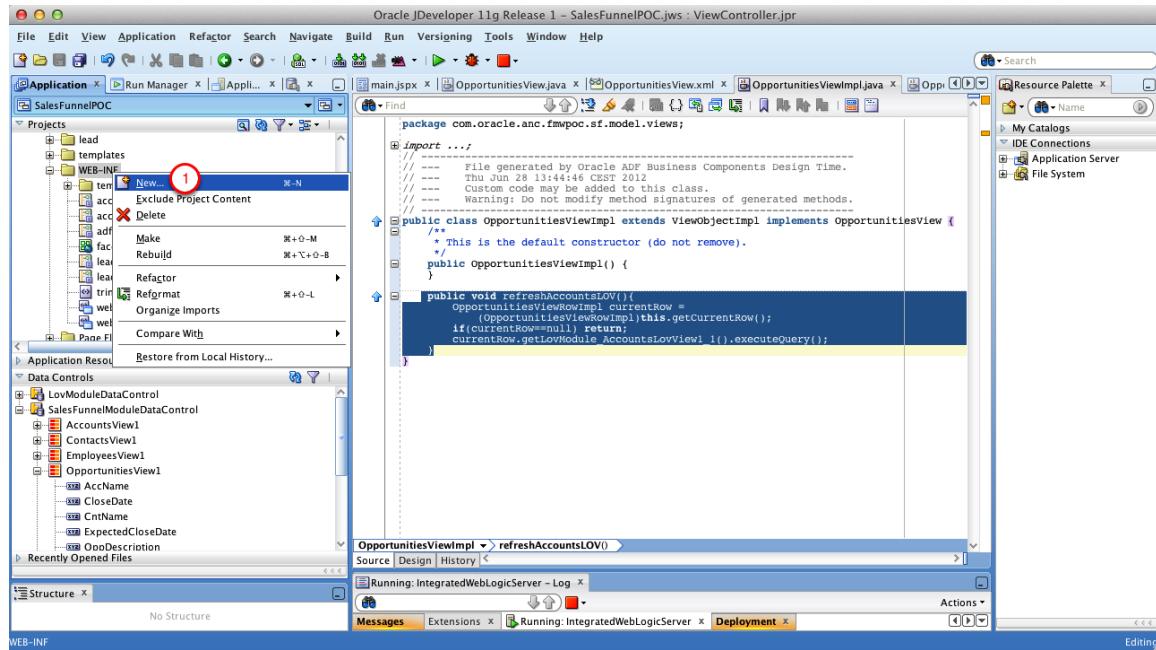
1. Open lead-edit-btf
2. Click the refresh icon in Data Controls
3. Expand OpportunitiesView1
4. Drag and drop refreshAccountsLOV to the diagram

Connect refreshAccountsLOV



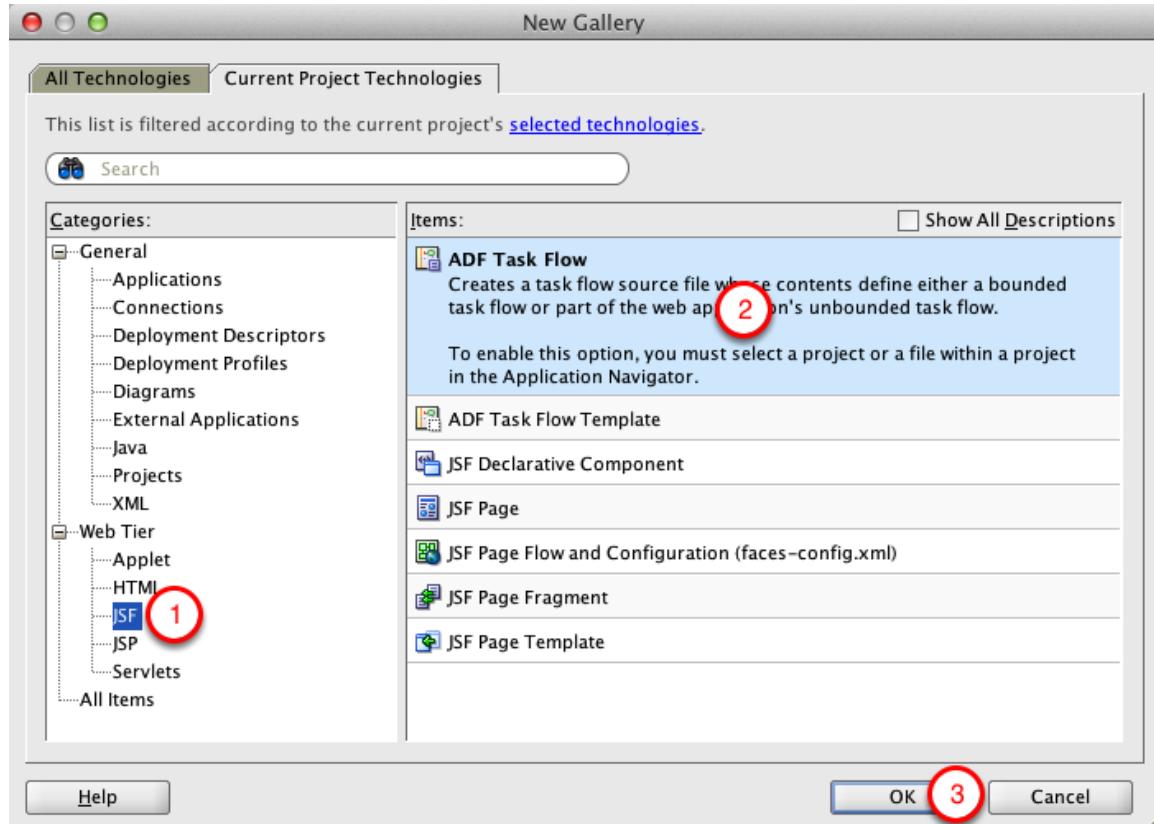
29. Contact Edit BTF

Create Contact Edit BTF



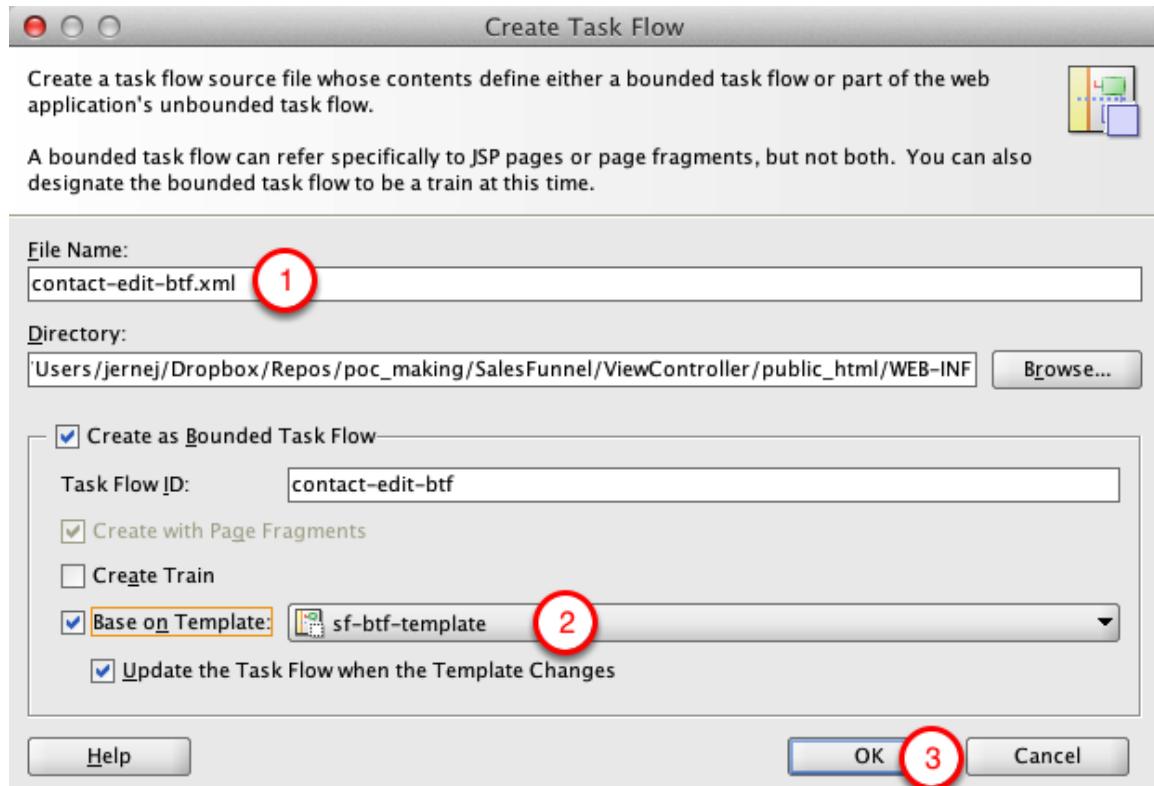
1. Right-click WEB-INF, select New

New Gallery



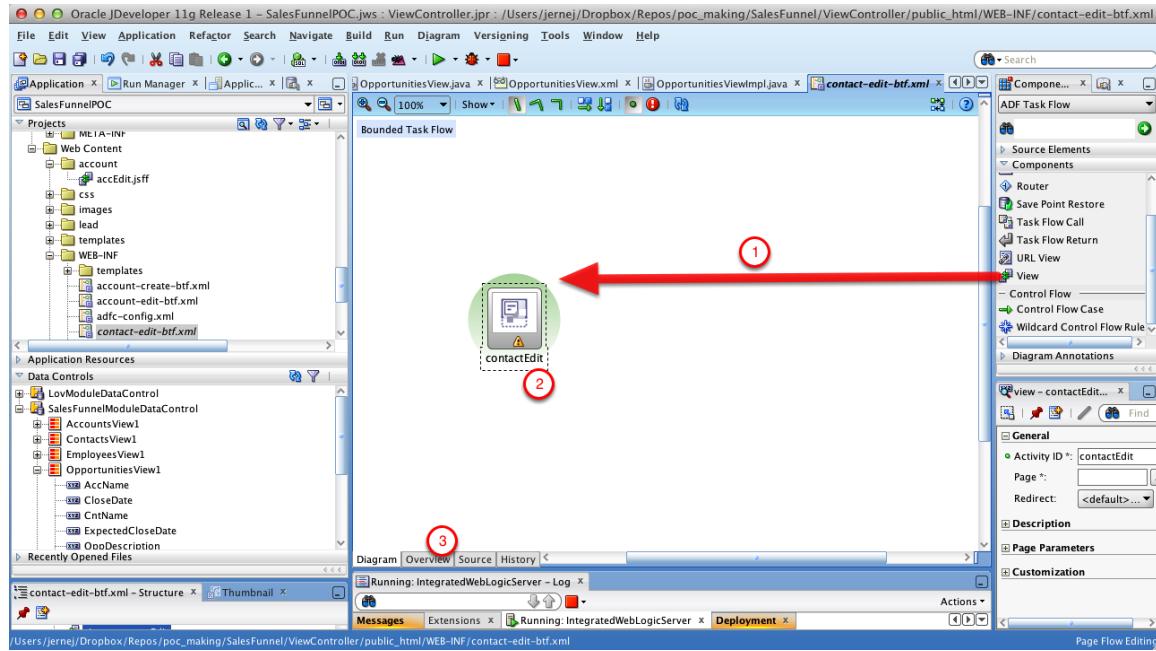
1. Select Web Tier > JSF
2. Select ADF Task Flow
3. Click OK

Create Task Flow



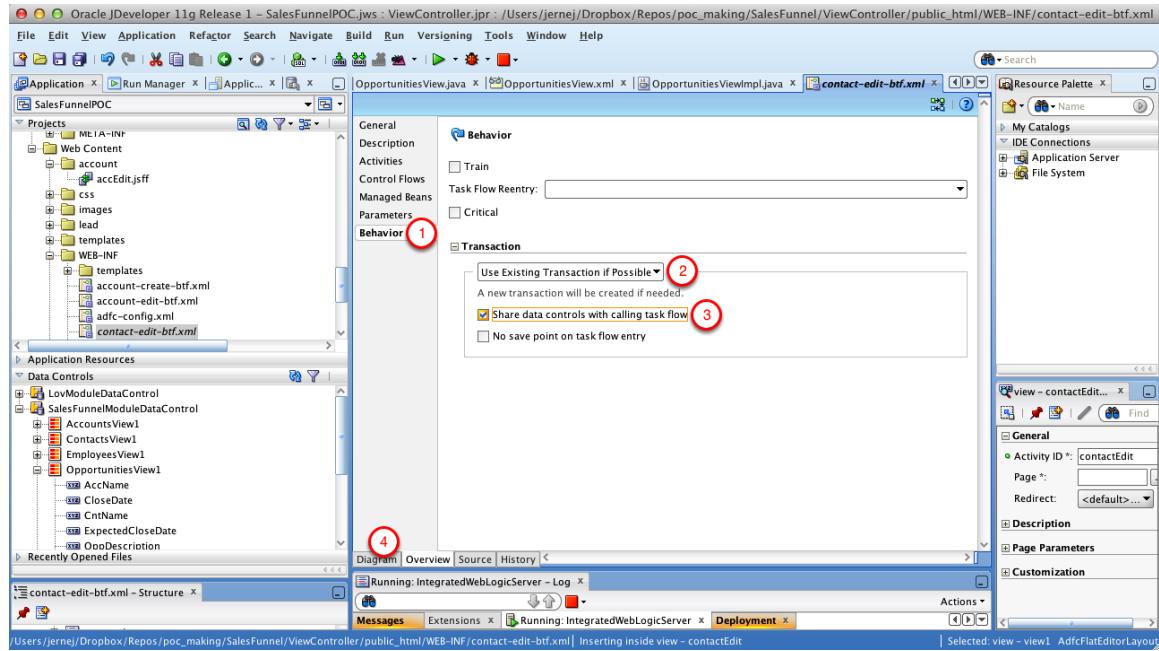
1. Name it contact-edit-btf
2. Base it on sf-btf-template
3. Click OK

Add contactEdit View



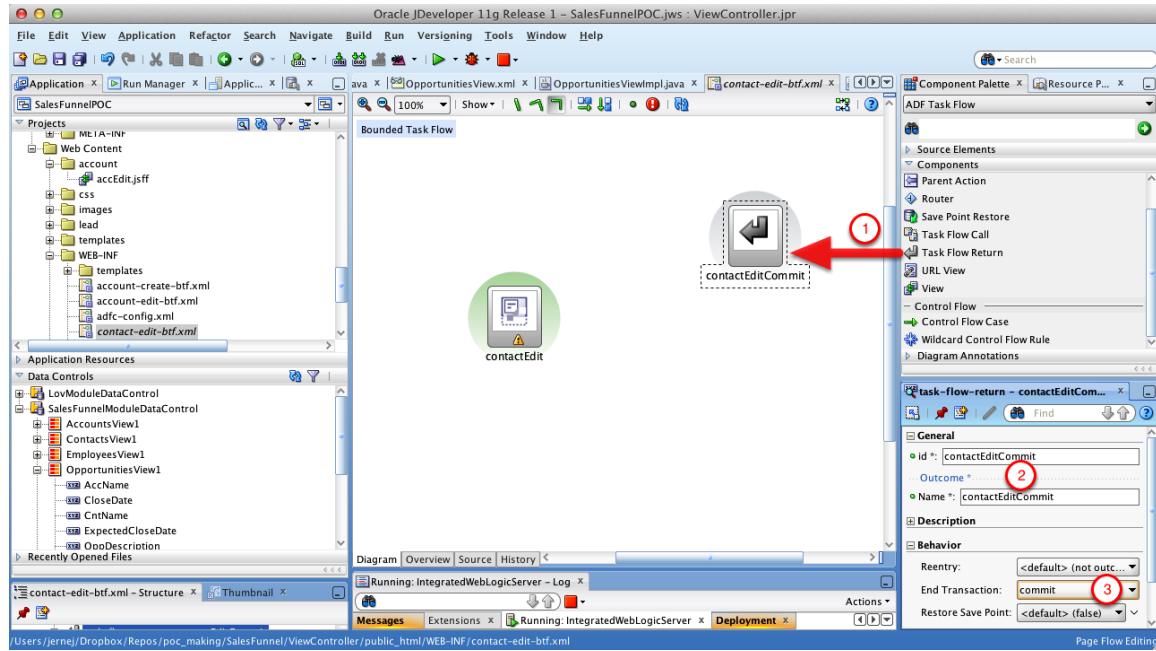
1. Drag and drop View activity on the diagram
2. Name it contactEdit
3. Open Overview Tab

Set contact-edit-btf behavior



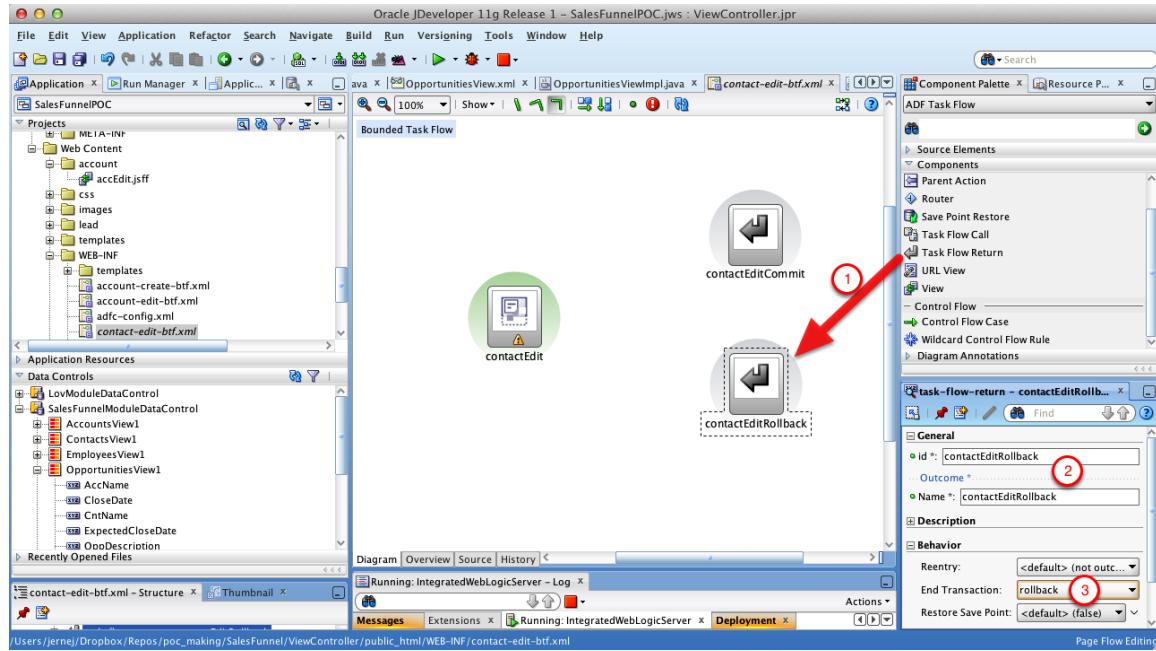
1. Open the Behavior tab
2. Select Use Existing Transaction if Possible option
3. Check Share Data Controls checkbox
4. Return to the diagram

Add contactEditCommit TF Return



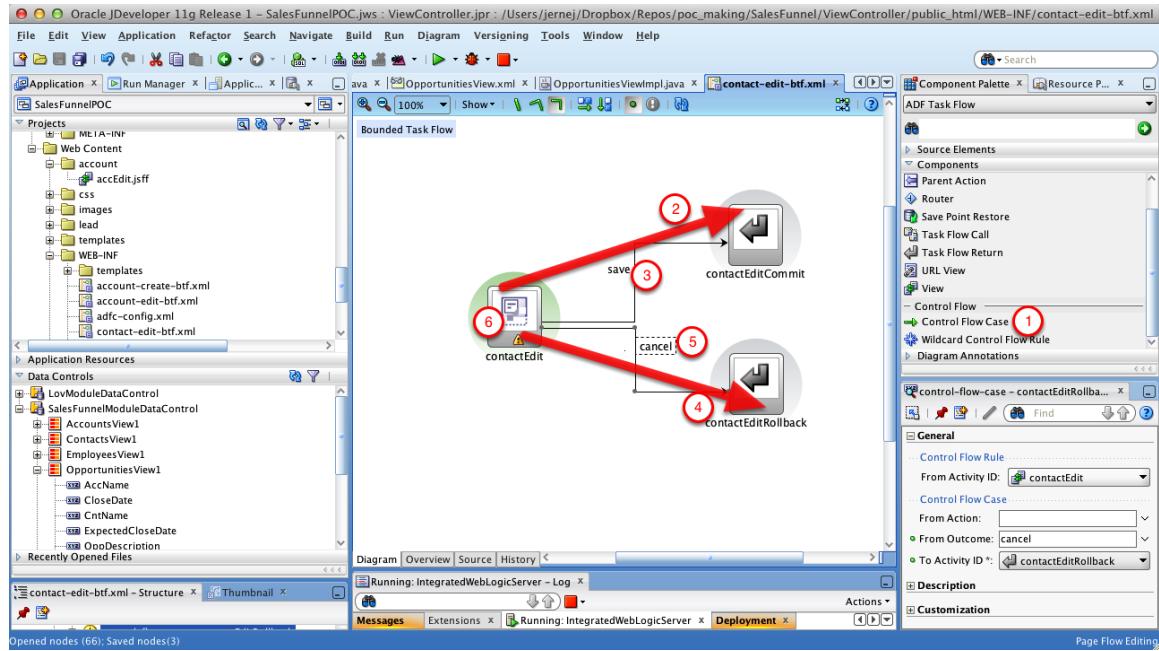
1. Drag and drop Task Flow Return to the diagram
2. Set id and name to contactEditCommit
3. Set End Transaction property to commit

Add contactEditRollback TF Return



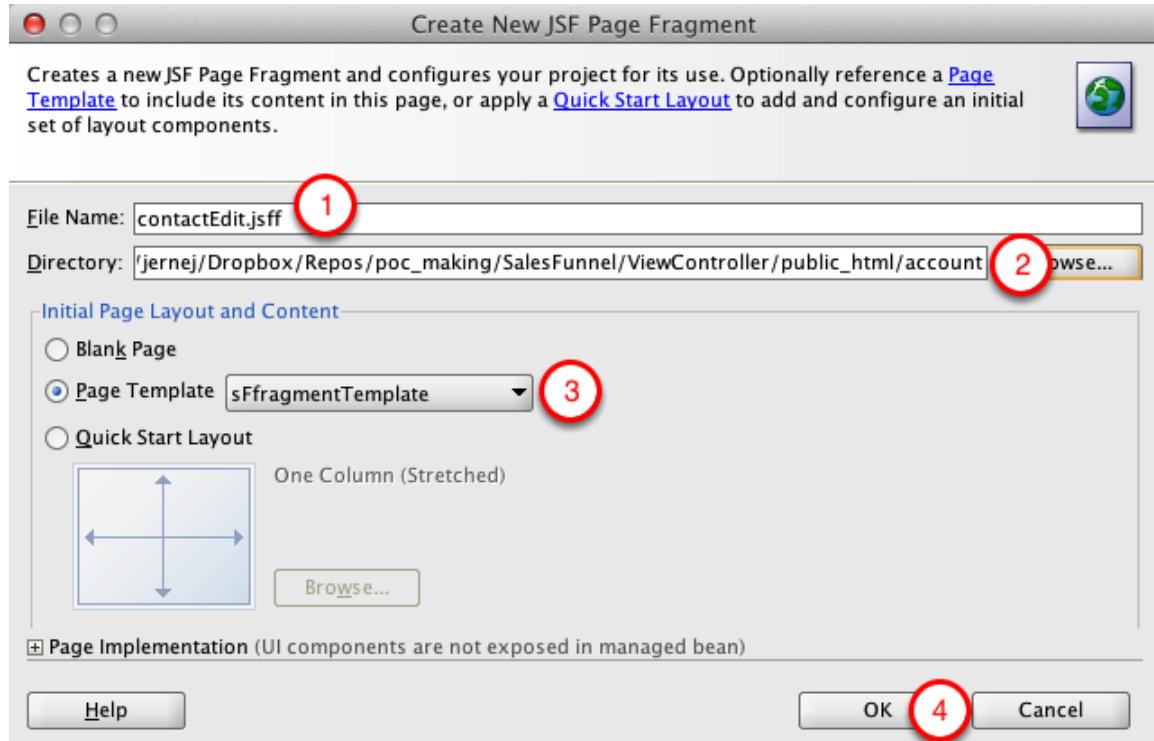
1. Drag and drop Task Flow Return to the diagram
2. Set id and name to contactEditRollback
3. Set End Transaction property to rollback

Connect contactEdit to task flow returns



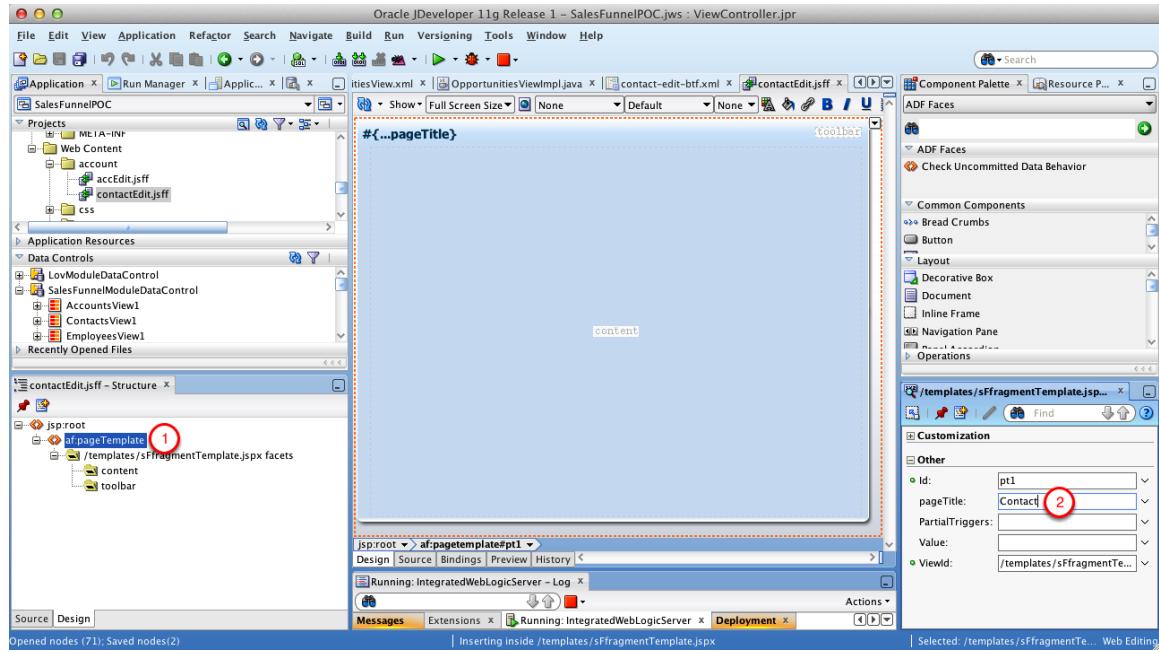
1. Select Control Flow Case in the Component Palette
2. Connect contactEdit to contactEditCommit
3. Name the action "save"
4. Connect contactEdit to contactEditRollback
5. Name the action "cancel"
6. Double click contactEdit view

Create New JSF Page Fragment



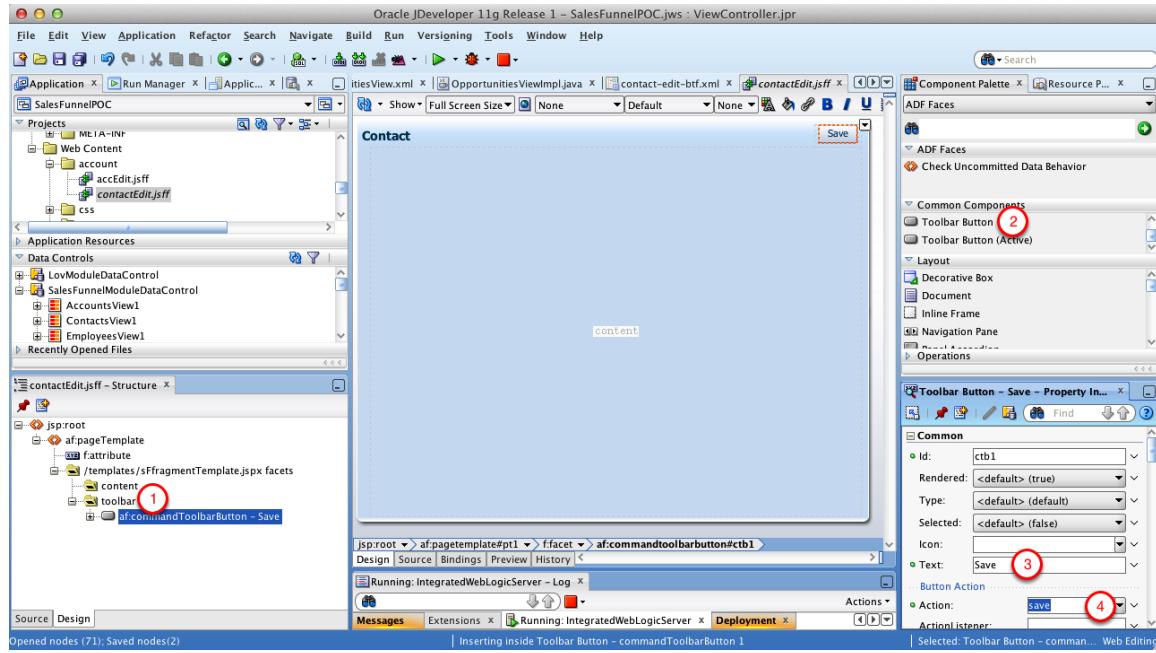
1. Set name to contactEdit.jsff
2. Append /account to the directory
3. Select sFfragmentTemplate
4. Click OK

Set page template attributes



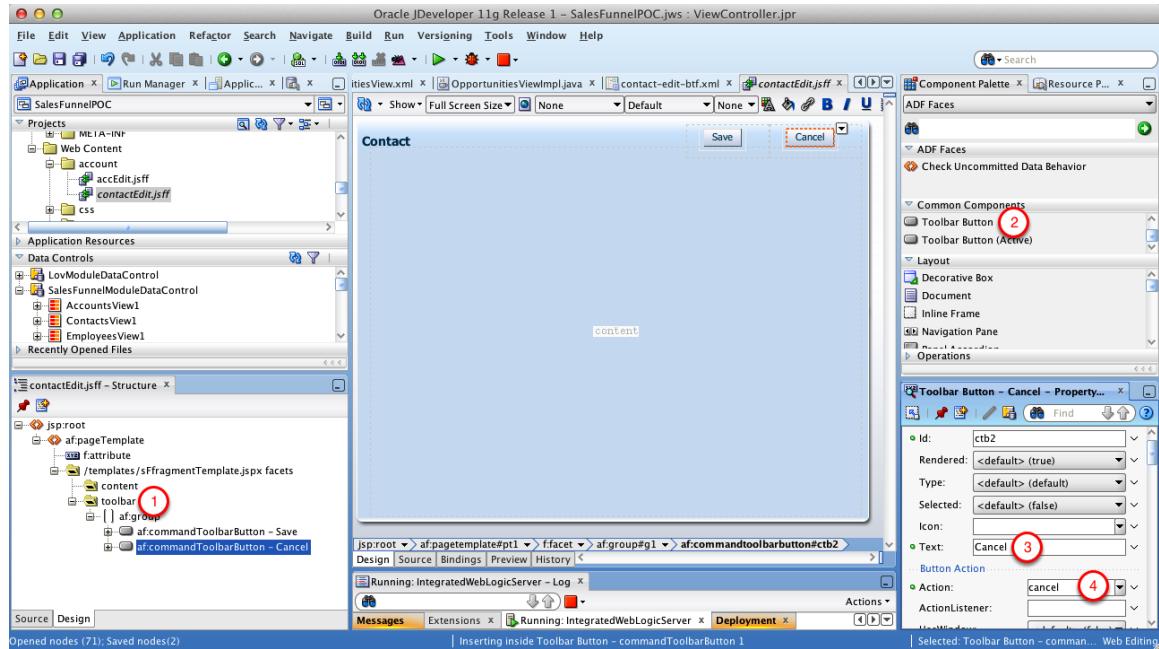
1. Select `af:pageTemplate` in the Structure window
2. Set `pageTitle` property to "Contact"

Add Save button



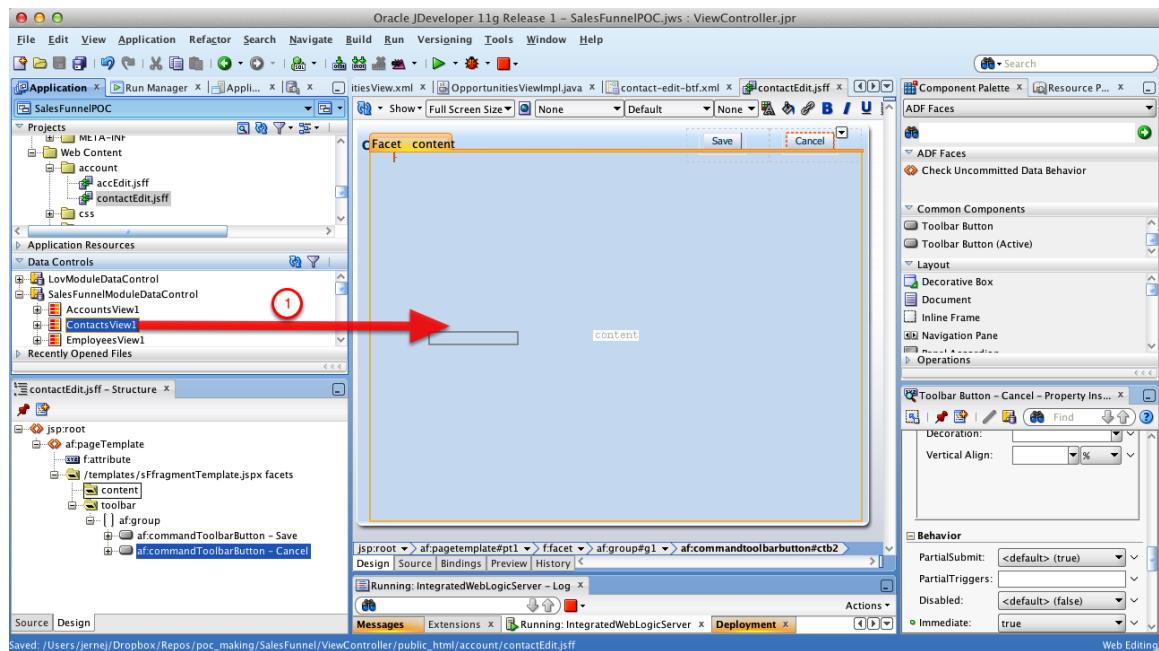
1. Select toolbar facet
2. Click Toolbar Button component in Component Palette
3. Set Text property to "Save"
4. Set Action property to "save"

Add Cancel button



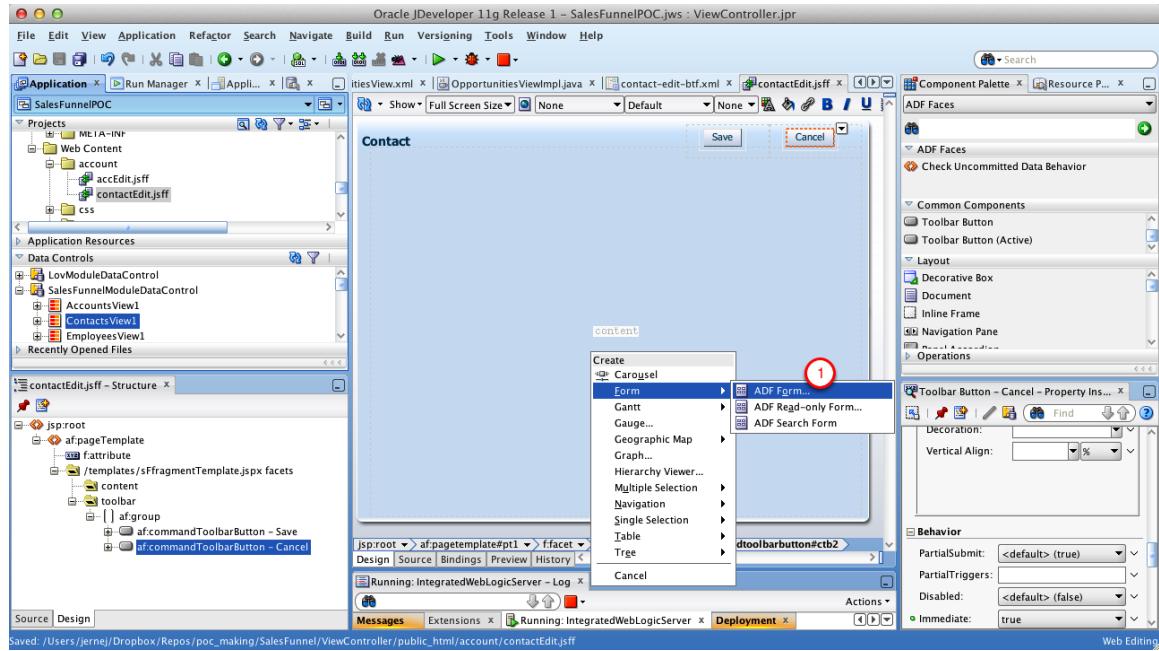
1. Select toolbar facet
2. Click Toolbar Button component in Component Palette
3. Set Text property to "Cancel"
4. Set Action property to "cancel"
5. Set Immediate property to true (not shown on the screenshot)

Drag and drop ContactsView1 to the form



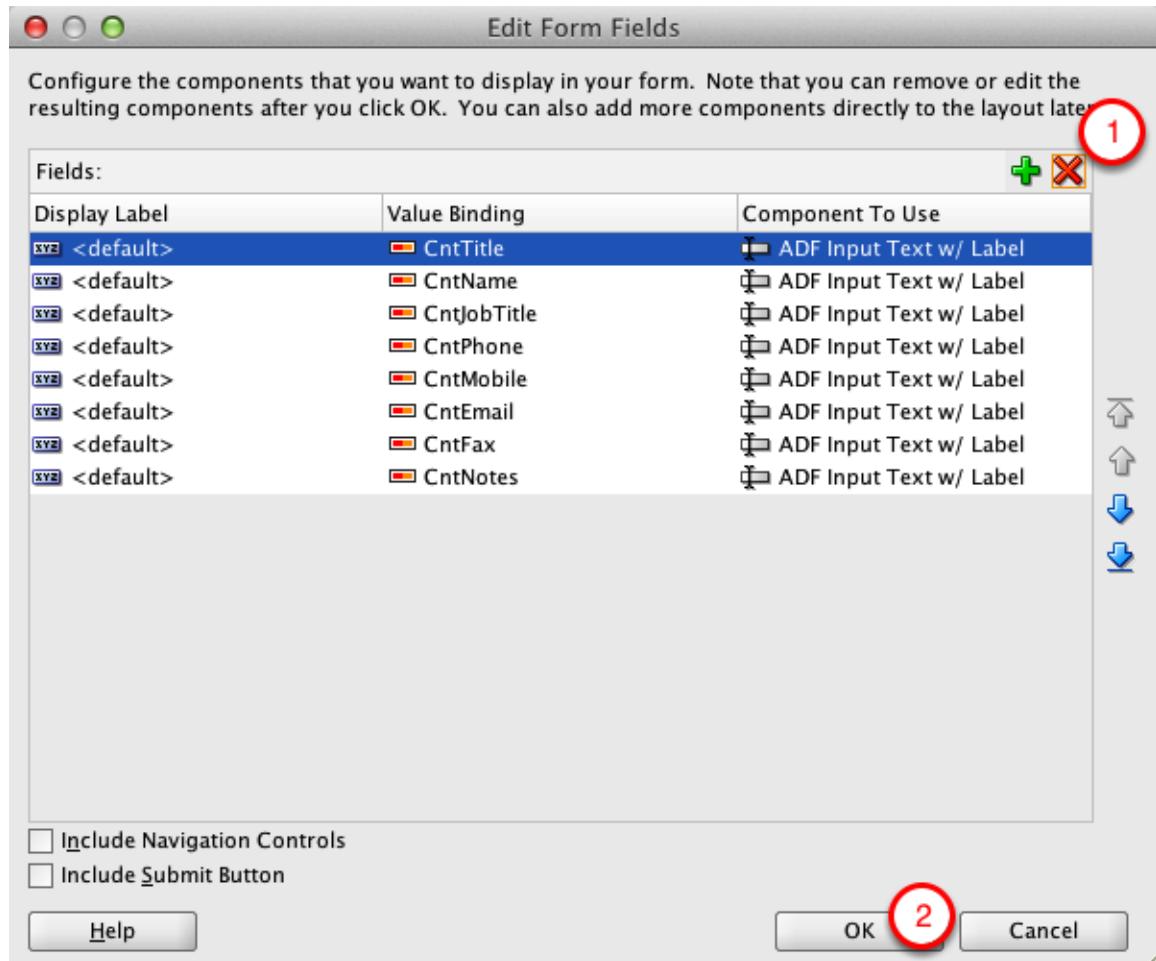
1. Drag and drop ContactsView1 to the content facet

Create Adf Form



1. Select Form > ADF Form from the context menu

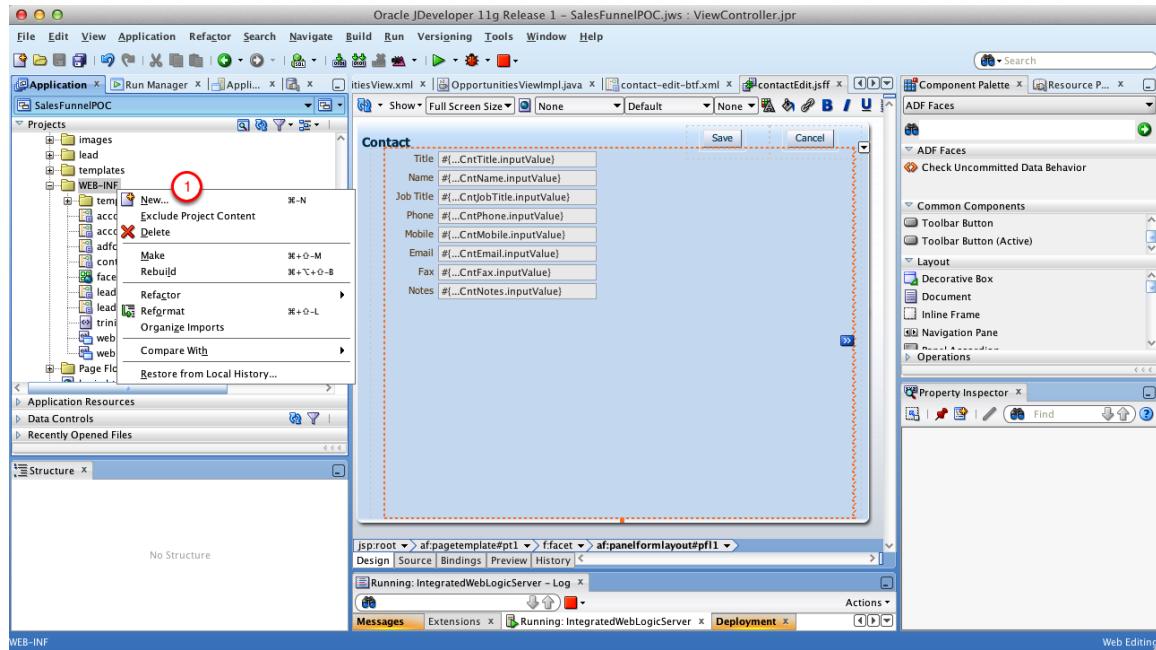
Edit Form Fields



1. Remove AccId from the field list
2. Click OK

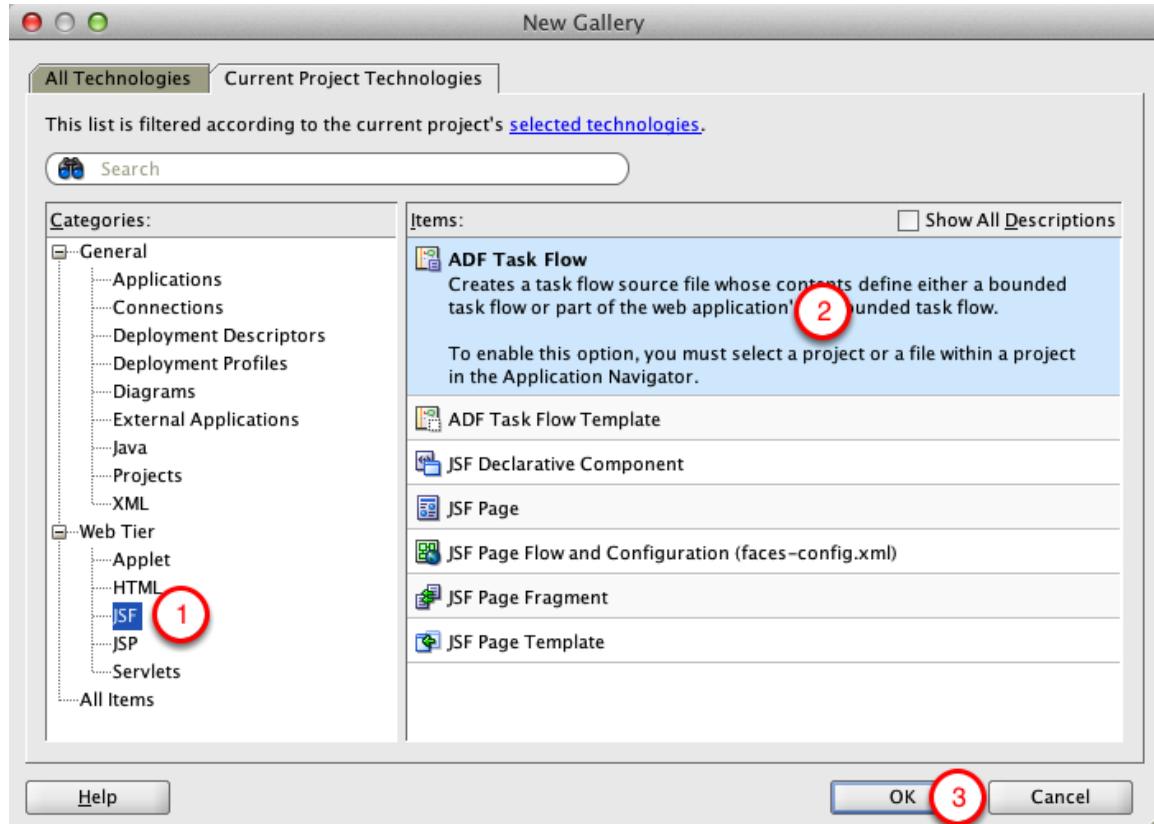
30. Contact Create BTF

Create Contact Create BTF



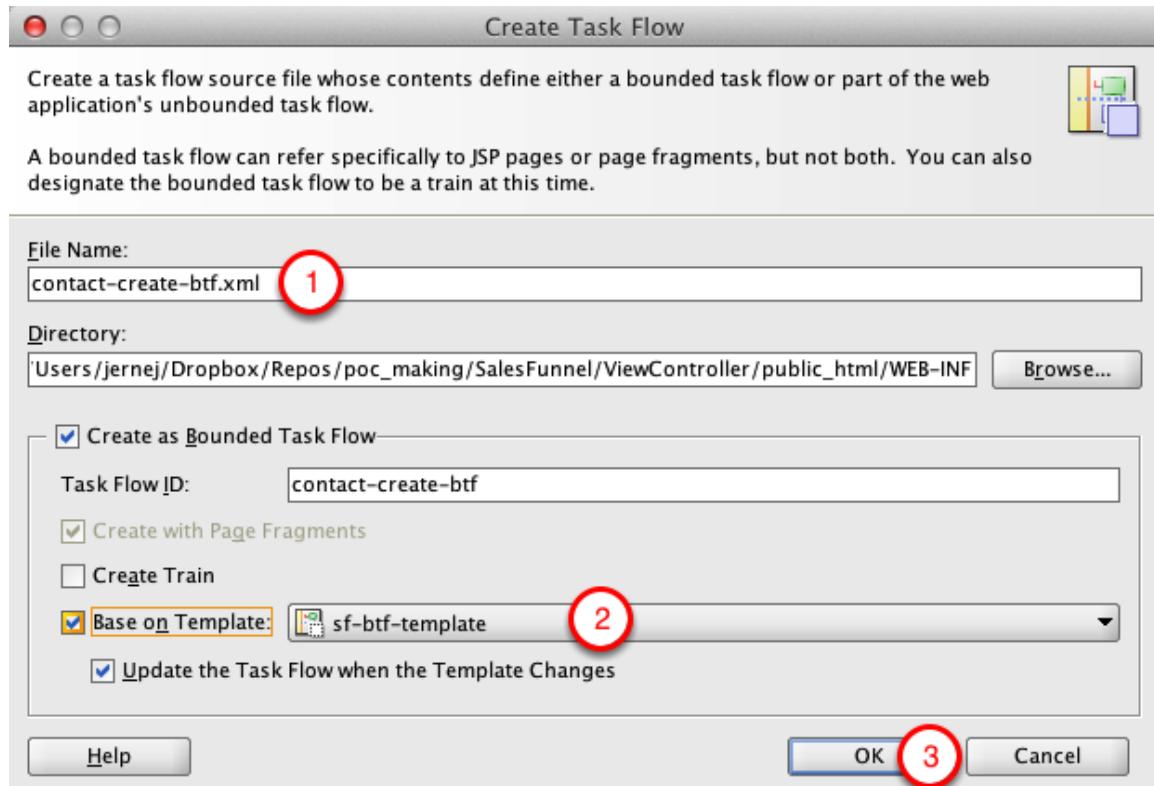
1. Right-click WEB-INF, select new

New Gallery



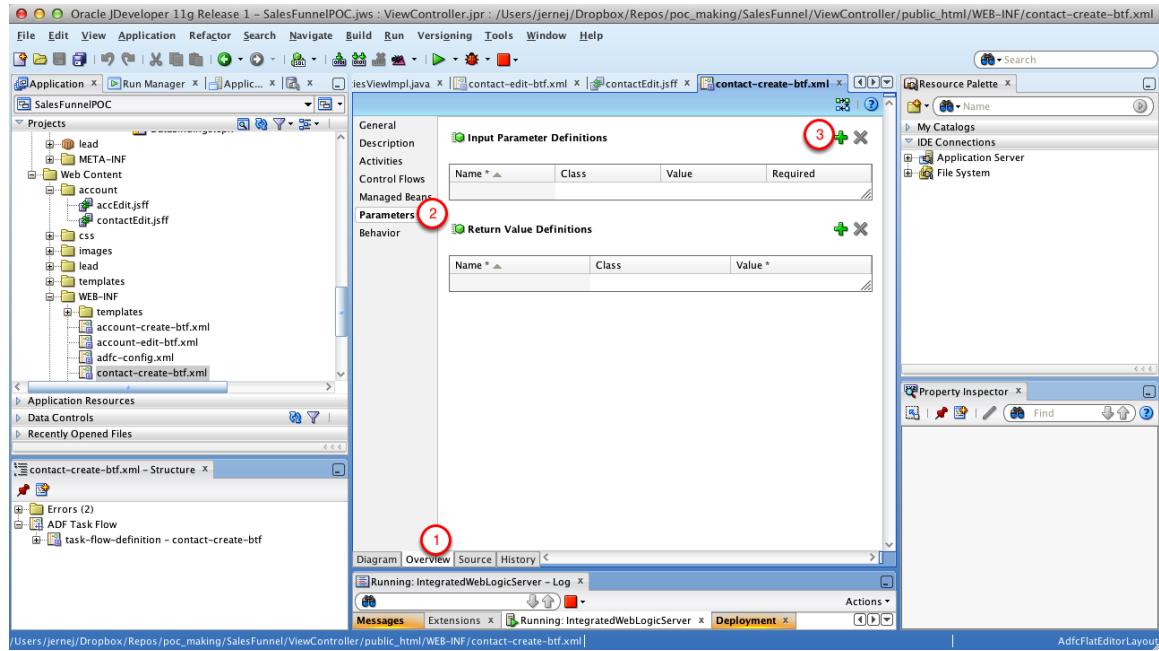
1. Select JSF under Web Tier
2. Select ADF Task Flow
3. Click OK

Create Task Flow



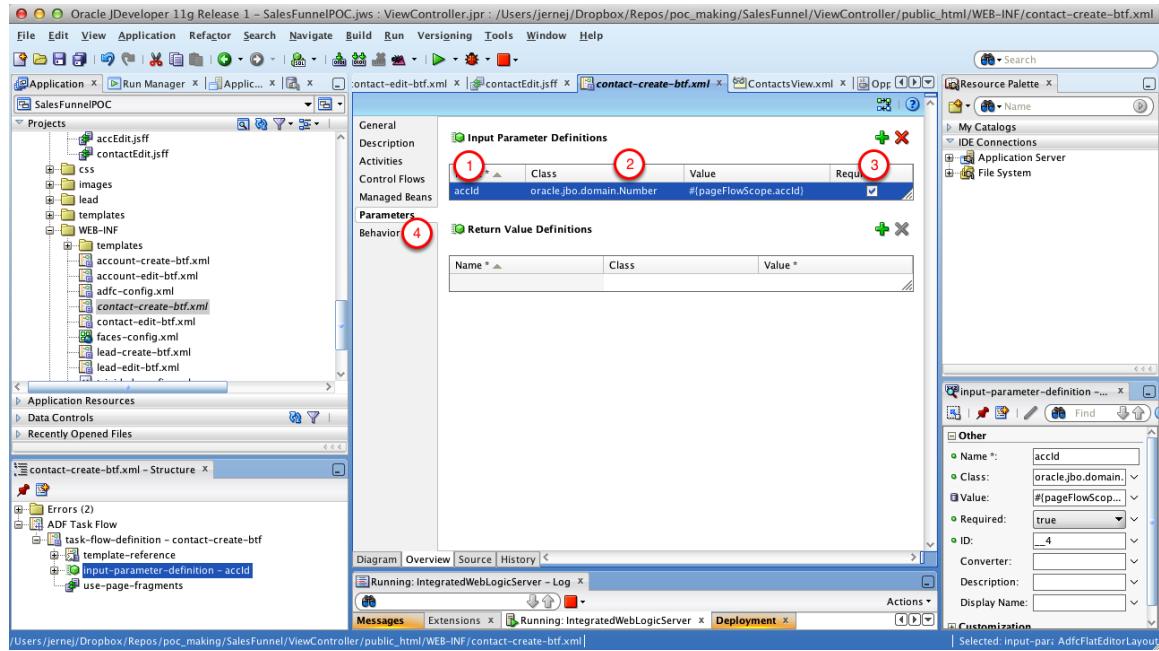
1. Name it contact-create-btf
2. Base it on sf-btf-template
3. Click OK

Add Task Flow Parameter



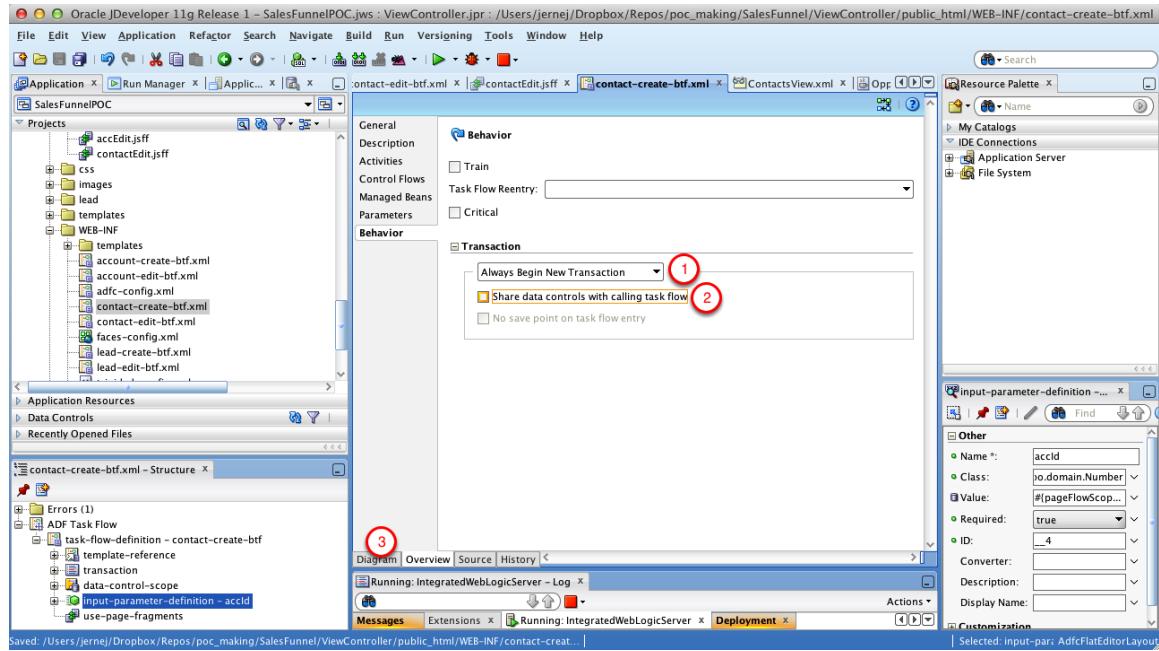
1. Open Overview Tab
2. Open Parameters
3. Click the Plus next to Input Parameter Definitions

Configure accId Parameter



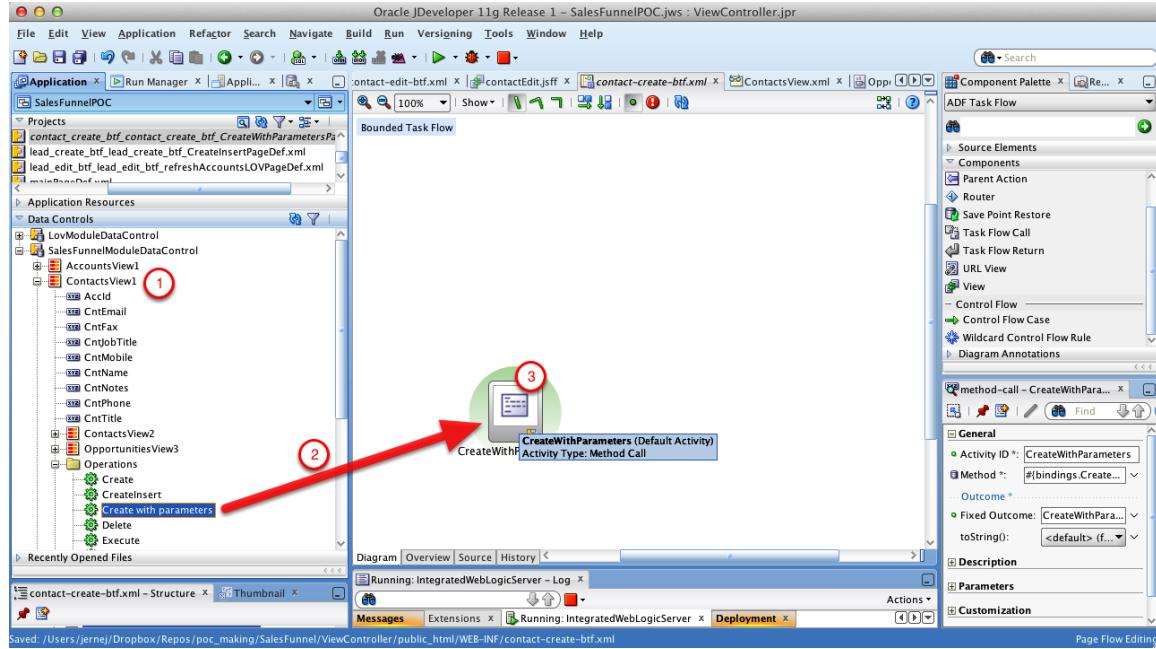
1. Set Name to accId
2. Set Class to oracle.jbo.domain.Number
3. Check Required Checkbox
4. Open Behavior tab

Configure contact-create-btf behavior



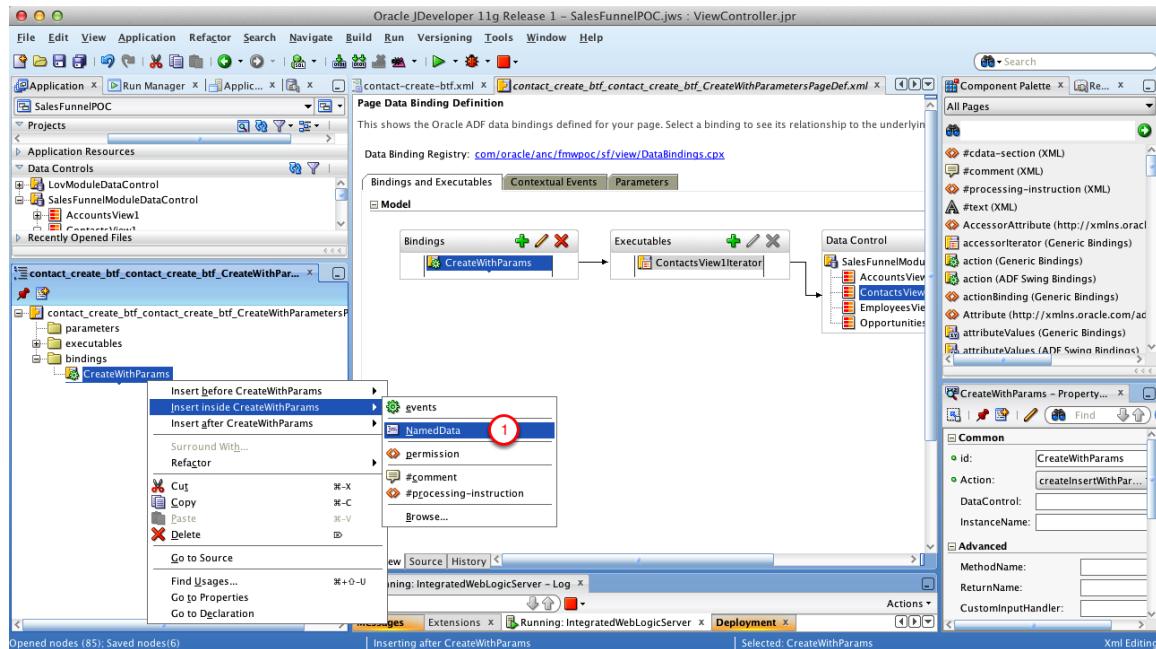
1. Set Always Begin New Transaction
2. Uncheck Share data controls
3. Open Diagram tab

Add ContactsView CreateWithParameters method



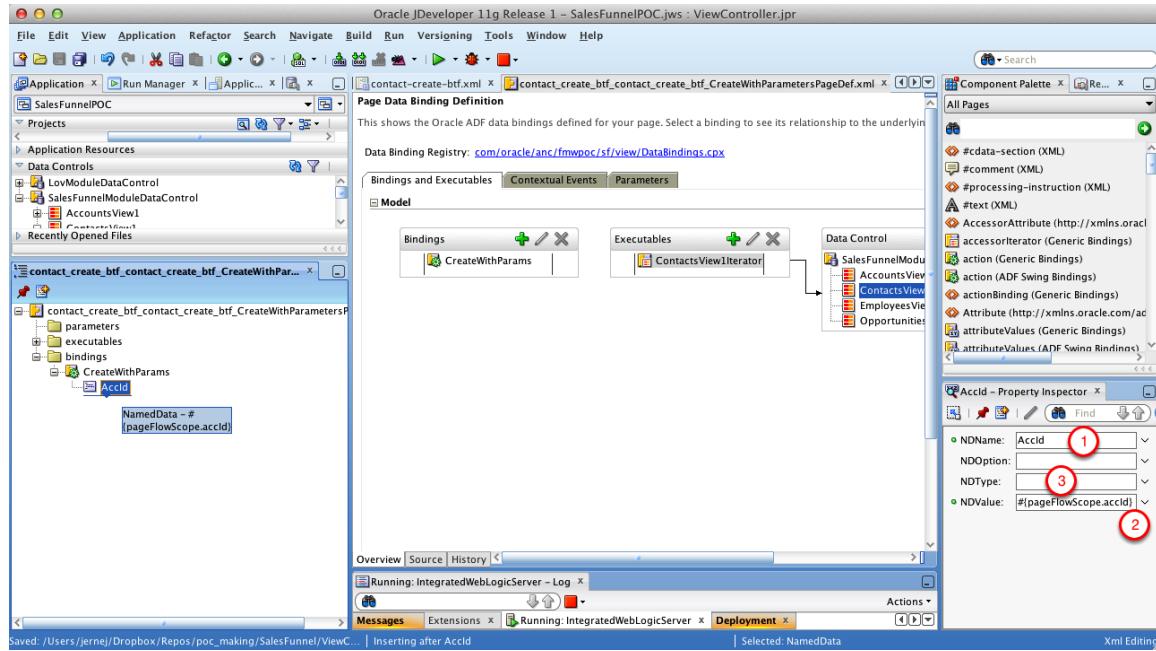
1. Expand ContactsView1 > Operations
2. Drag and drop Create with parameters to the diagram
3. Double-click CreateWithParameters activity

Set CreateWithParams Parameters



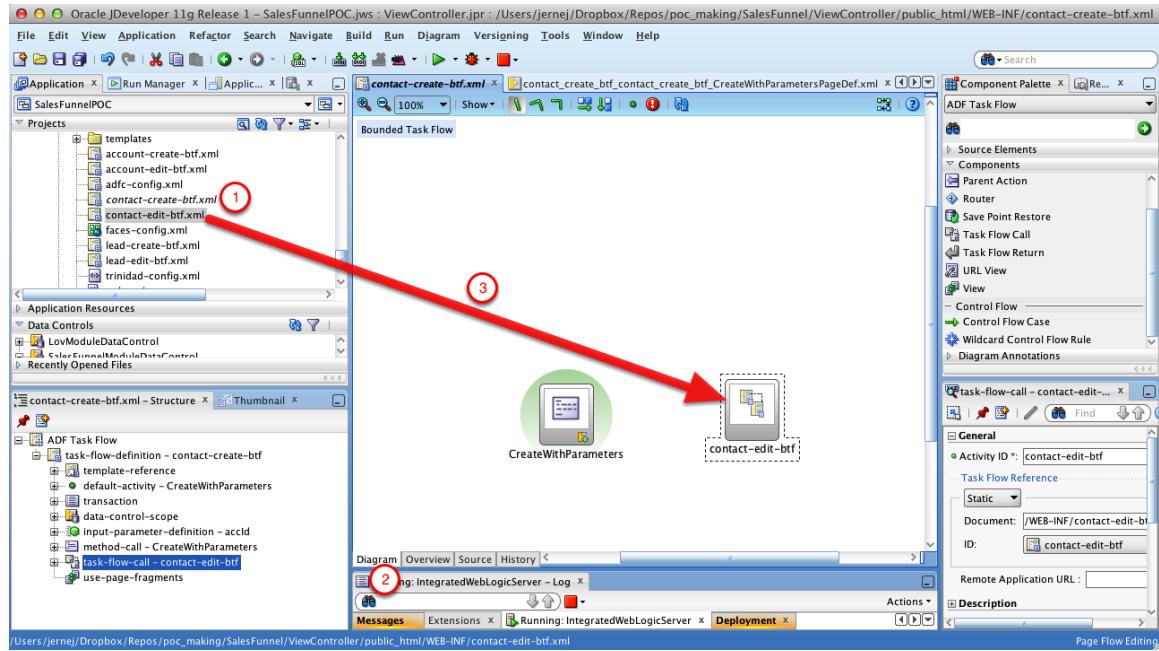
1. In the structure window, right-click CreateWithParams and select Insert inside CreateWithParams > NamedData

Set AccId Parameter



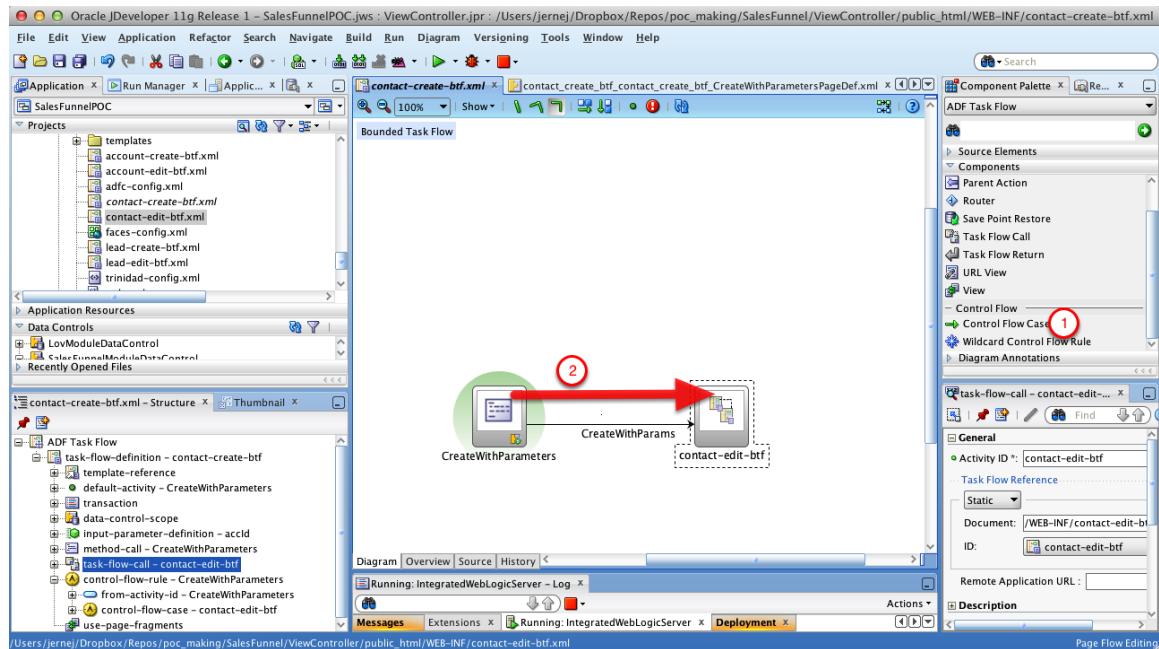
1. Set NDName property to AccId
2. Set NDValue property to #{pageFlowScope.accid}
3. Set NDType property to oracle.jbo.domain.Number (not shown on the screenshot)

Add contact-edit-btf task flow



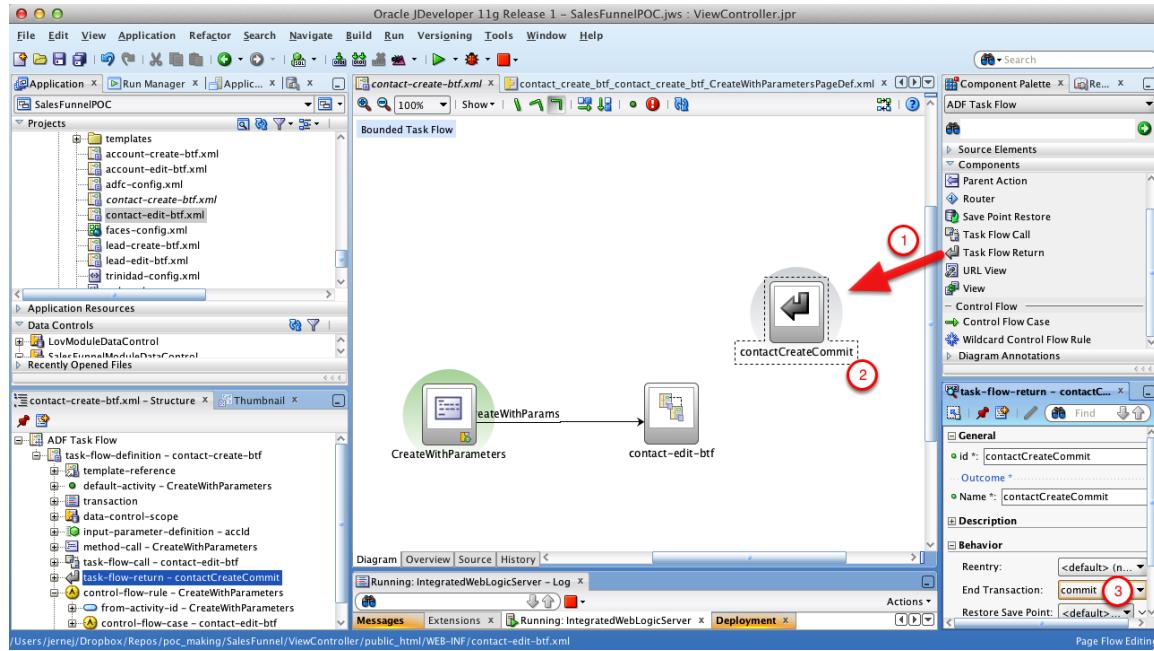
1. Open contact-edit-btf
2. Open Diagram tab if needed
3. Drag and drop contact-edit-btf to the diagram

Connect CreateWithParameters to contact-edit-btf



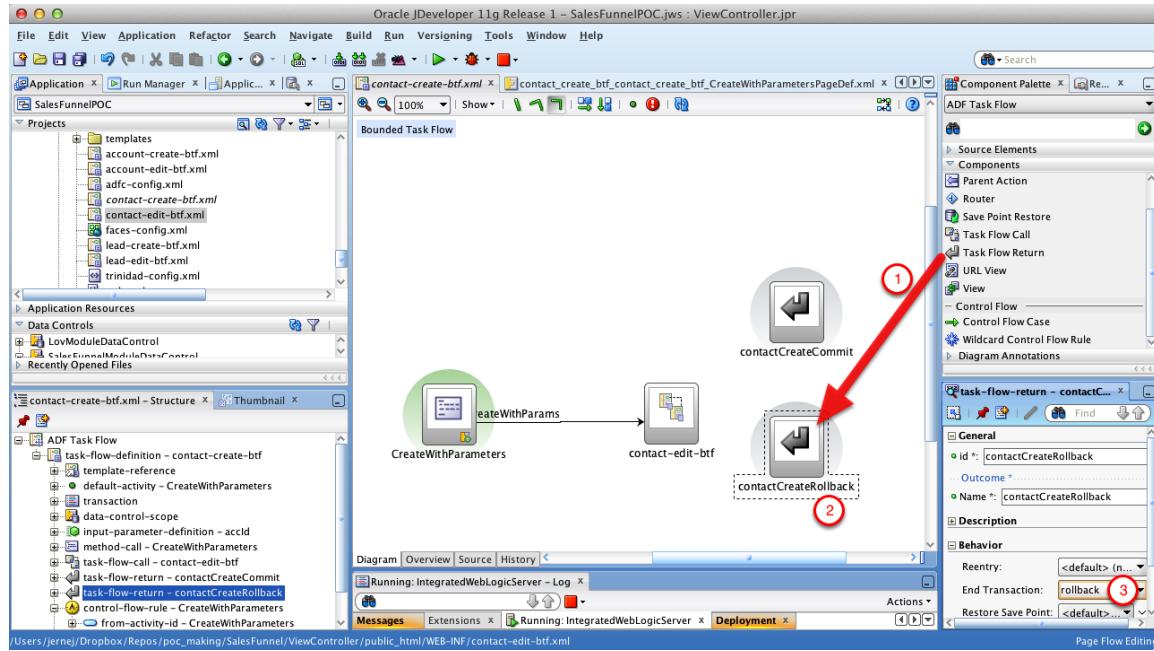
1. Select Control Flow Case from the Component Palette
2. Connect CreateWithParameters to contact-edit-btf

Add contactCreateCommit task flow return



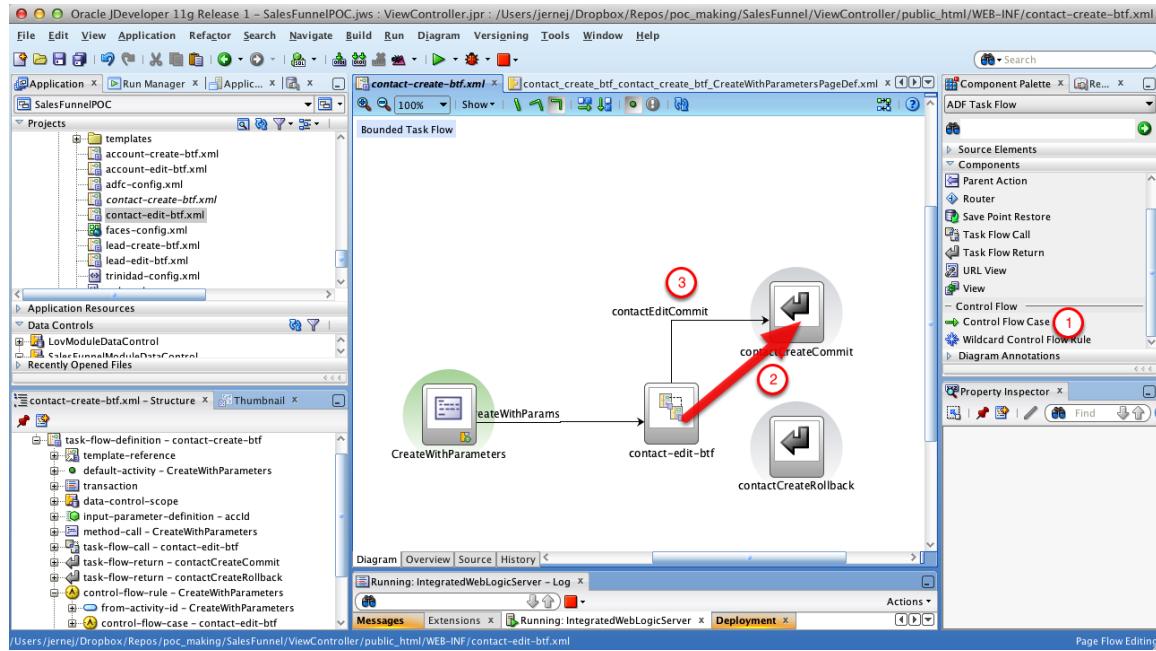
1. Drag and drop Task Flow Return to the diagram
2. Name it contactCreateCommit
3. Set End Transaction Property to commit

Add contactCreateRollback task flow return



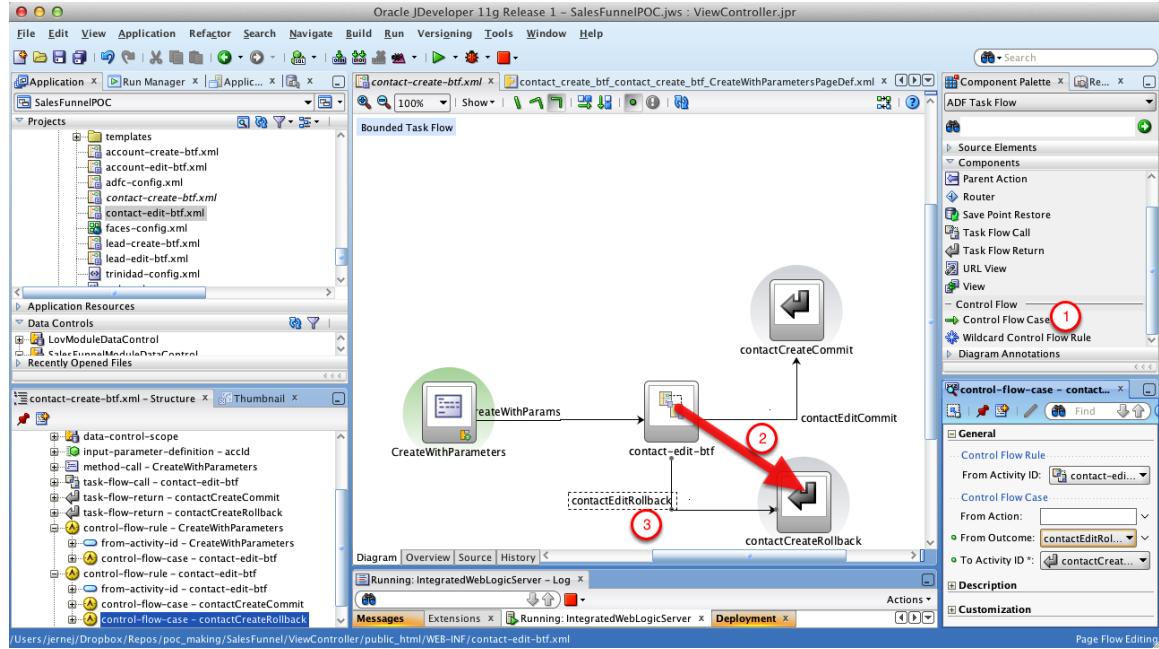
1. Drag and drop Task Flow Return to the diagram
2. Name it contactCreateRollback
3. Set End Transaction Property to rollback

Connect contact-edit-btf to contactCreateCommit



1. Select Control Flow Case in the Component Palette
2. Connect contact-edit-btf to contactCreateCommit
3. Make sure from outcome is contactEditCommit

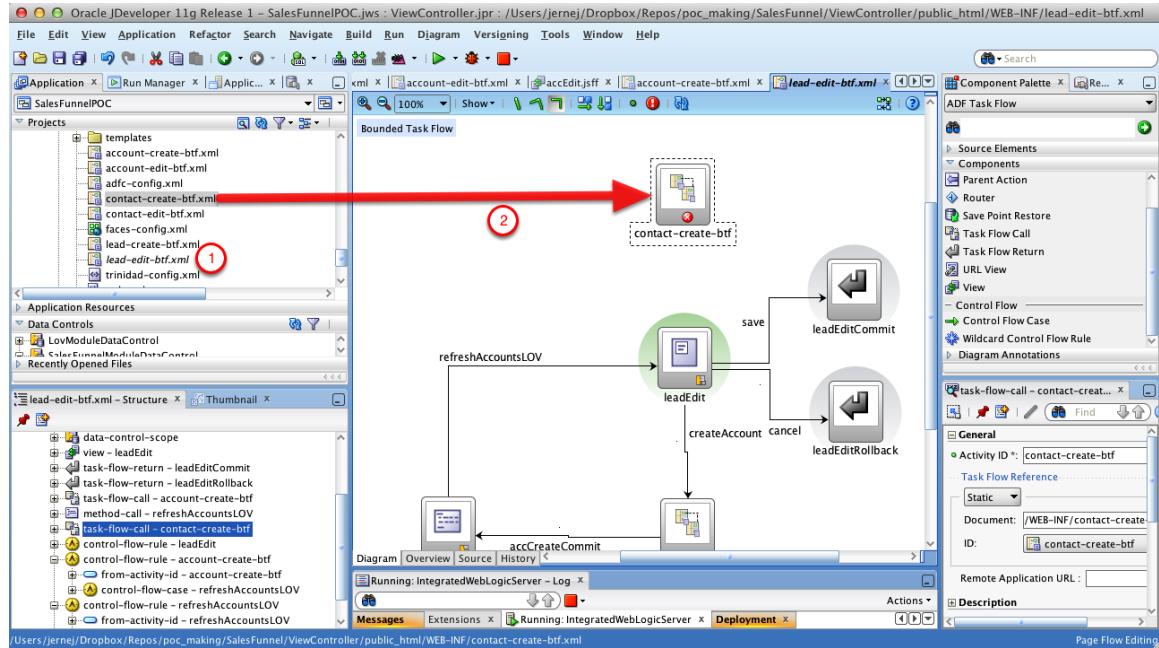
Connect contact-edit-btf to contactCreateRollback



1. Select Control Flow Case in the Component Palette
2. Connect contact-edit-btf to contactCreateRollback
3. Make sure from outcome is contactEditRollback

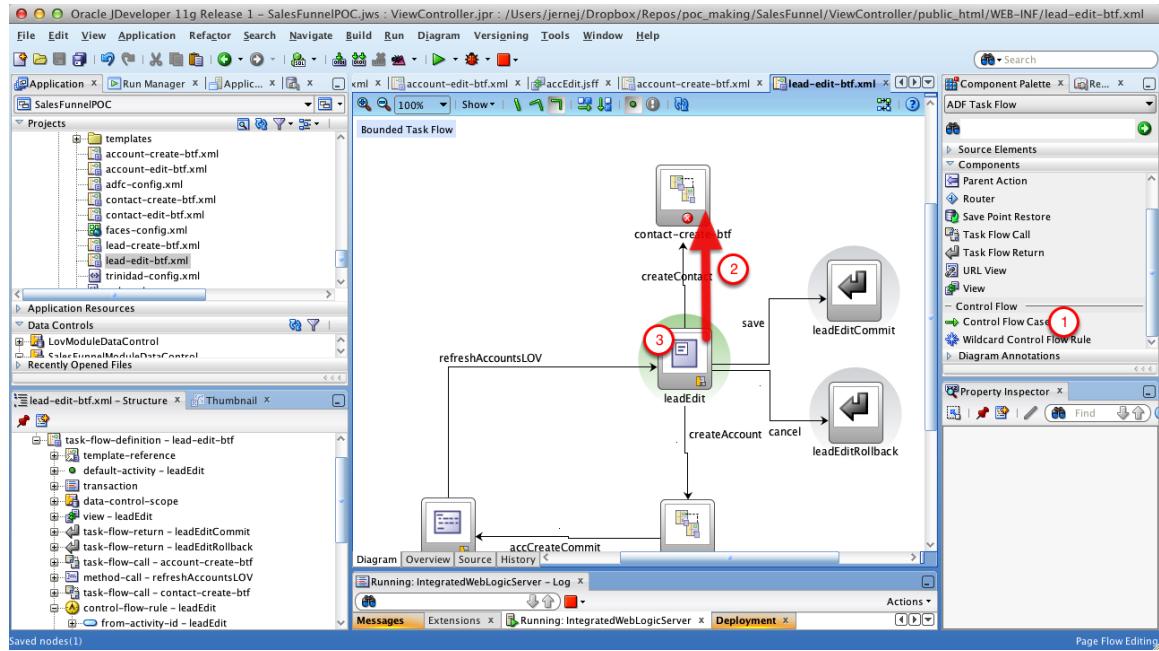
31. Adding Contact Create to Lead Edit

Adding contact-create-btf to lead-edit-btf



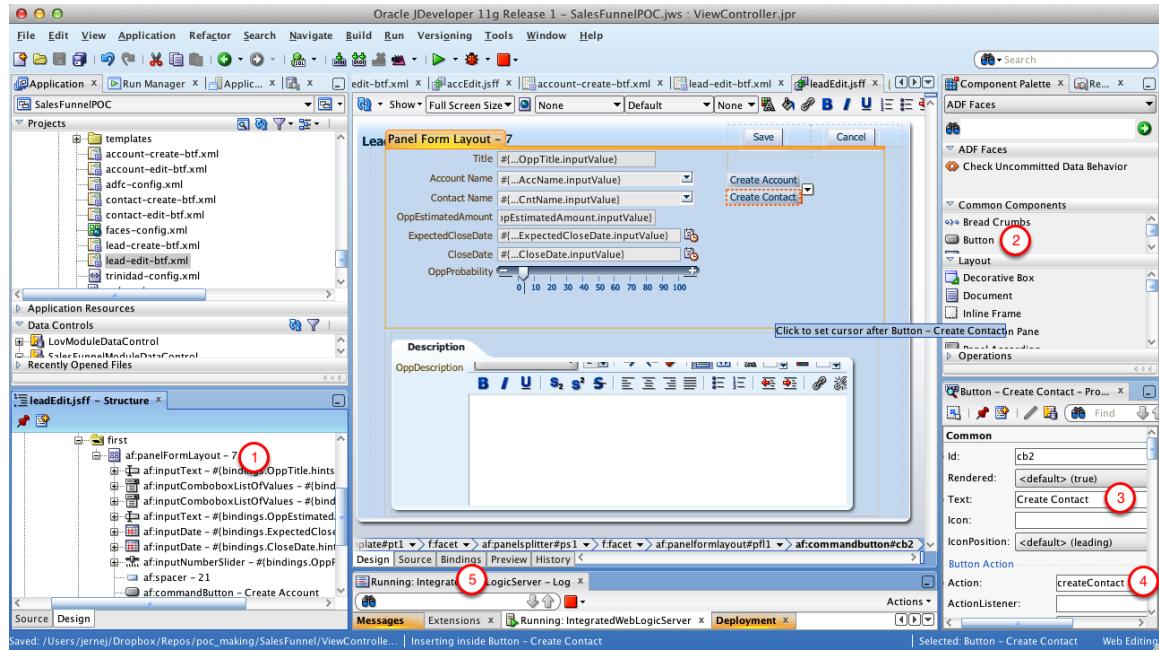
1. Open lead-edit-btf
2. Drag and drop contact-create-btf to the diagram

Connect leadEdit to contact-create-btf



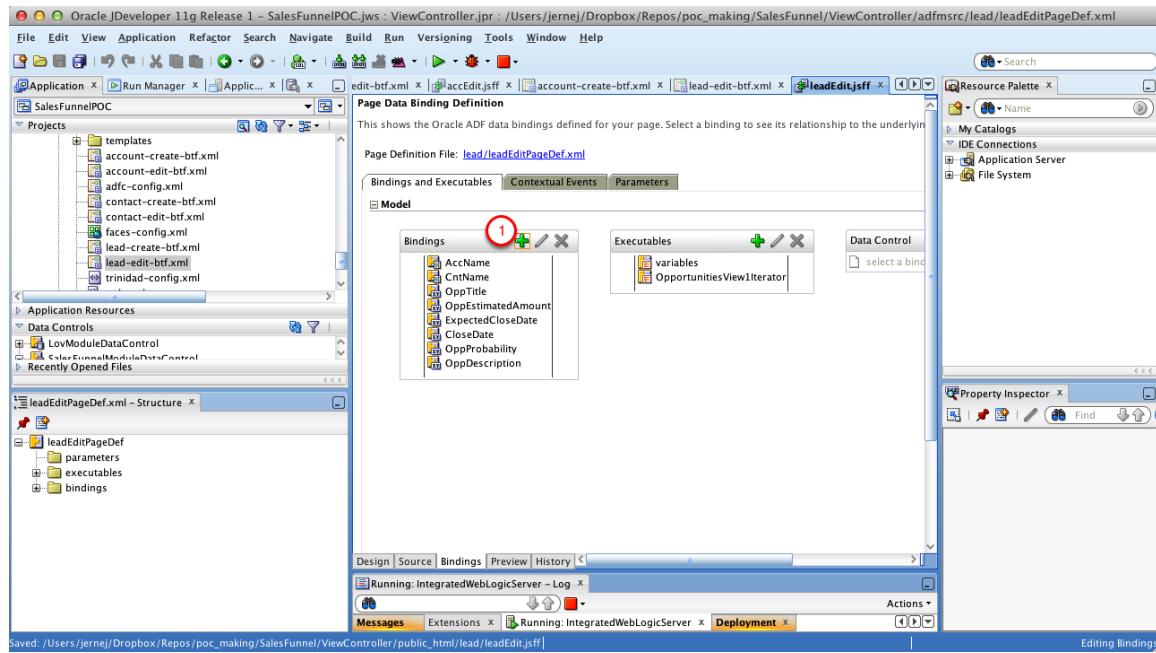
1. Select Control Flow Case from the Component Palette
2. Connect leadEdit to contact-create-btf
3. Double-click leadEdit to open it

Add CreateContact button



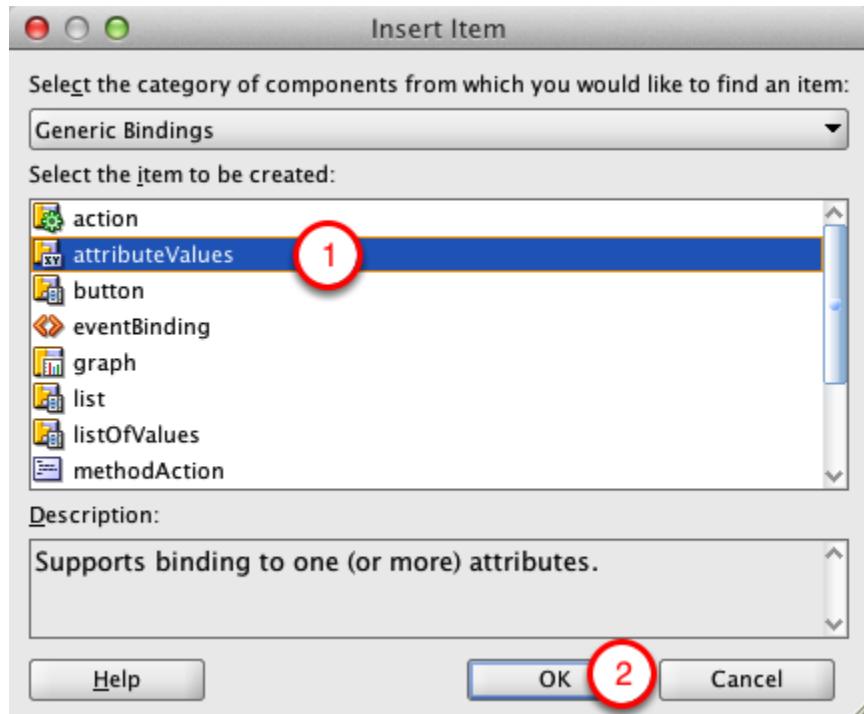
1. Select panelFormLayout in the Structure window
2. Click Button in the Component Palette
3. Set Text property of the button to "Create Contact"
4. Set Action property of the button to "createContact"
5. Set Immediate property to true (not visible on the screenshot)
6. Open Bindings tab

Add AccId attribute binding



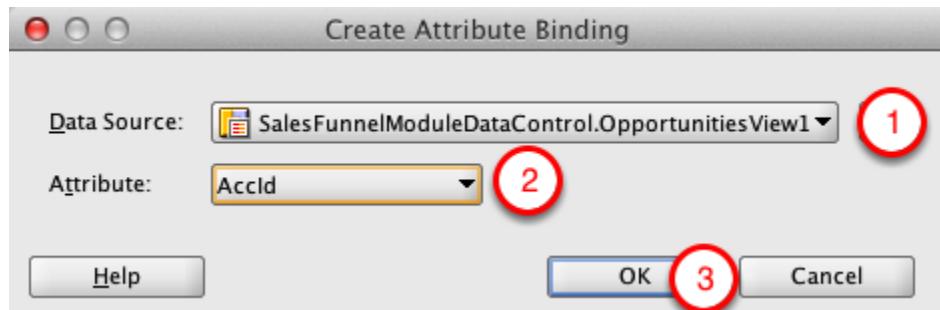
1. Click the plus in Bindings section

Insert Item



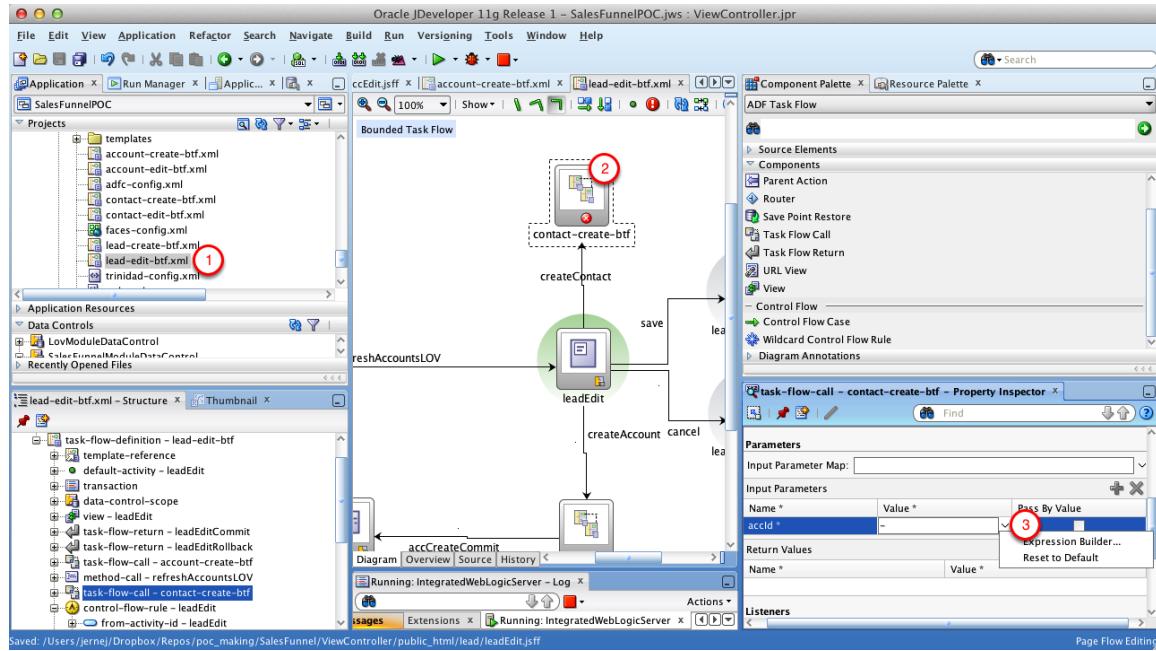
1. Select attributeValues
2. Click OK

Create Attribute Binding



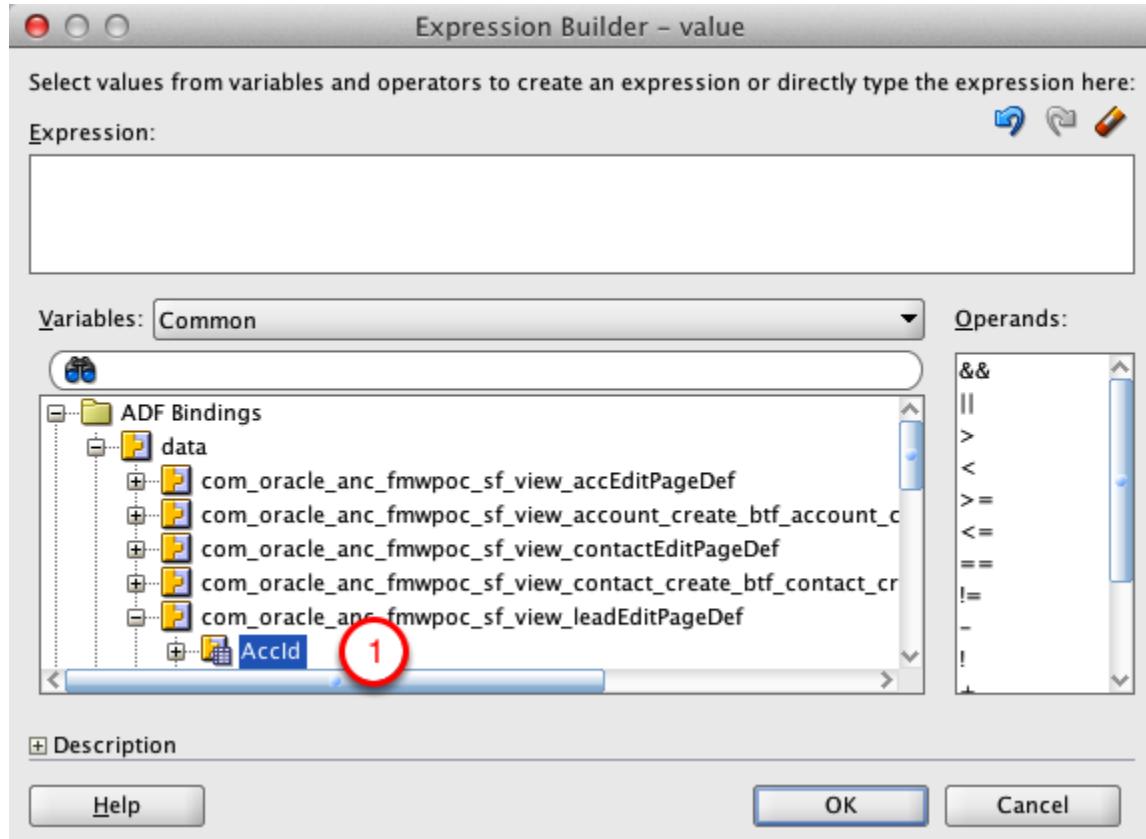
1. Select OpportunitiesView1 as Data Source
2. Select AcId as Attribute
3. Click OK

Set contact-create-btf accId parameter value



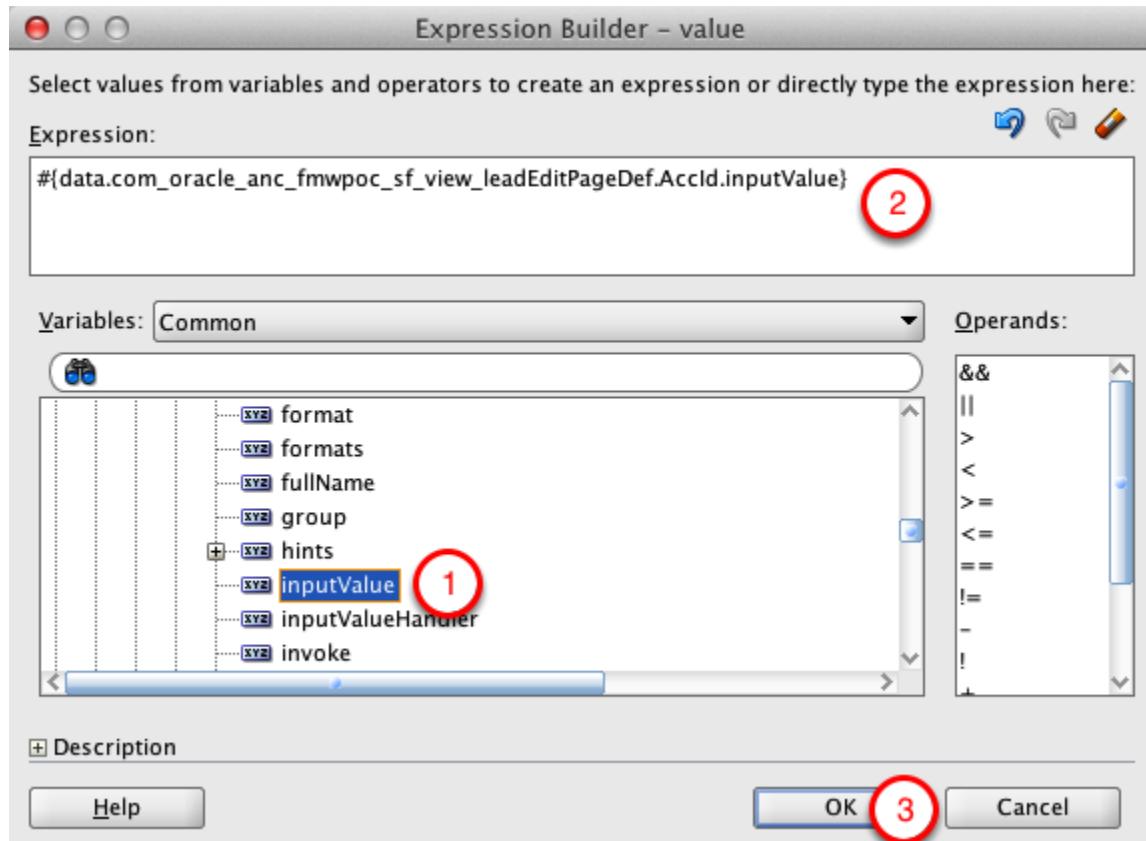
1. Open lead-edit-btf
2. Click on contact-create-btf to select it
3. Click on the down arrow next to accId input parameter Value and click Expression Builder

Expression Builder - value



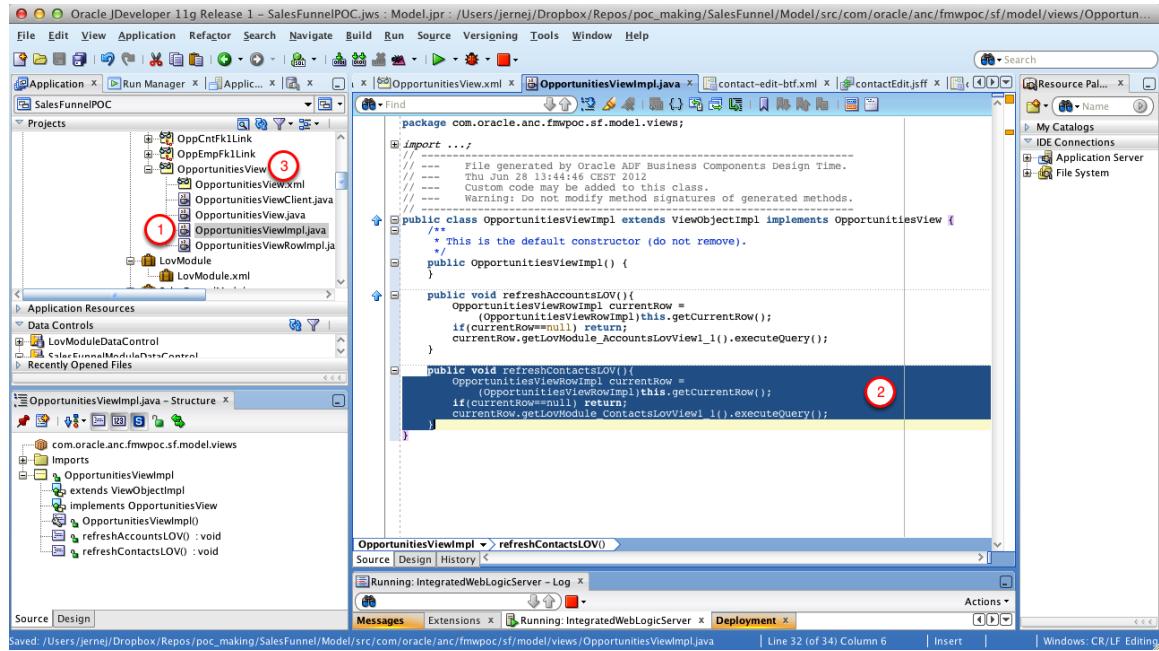
1. Expand ADF Bindings > data > com_oracle_anc_fmw poc_sf_view_leadEditPageDef > AcclId

Expression Builder - value



1. Select inputValue
2. Check the expression is correct
3. Click OK

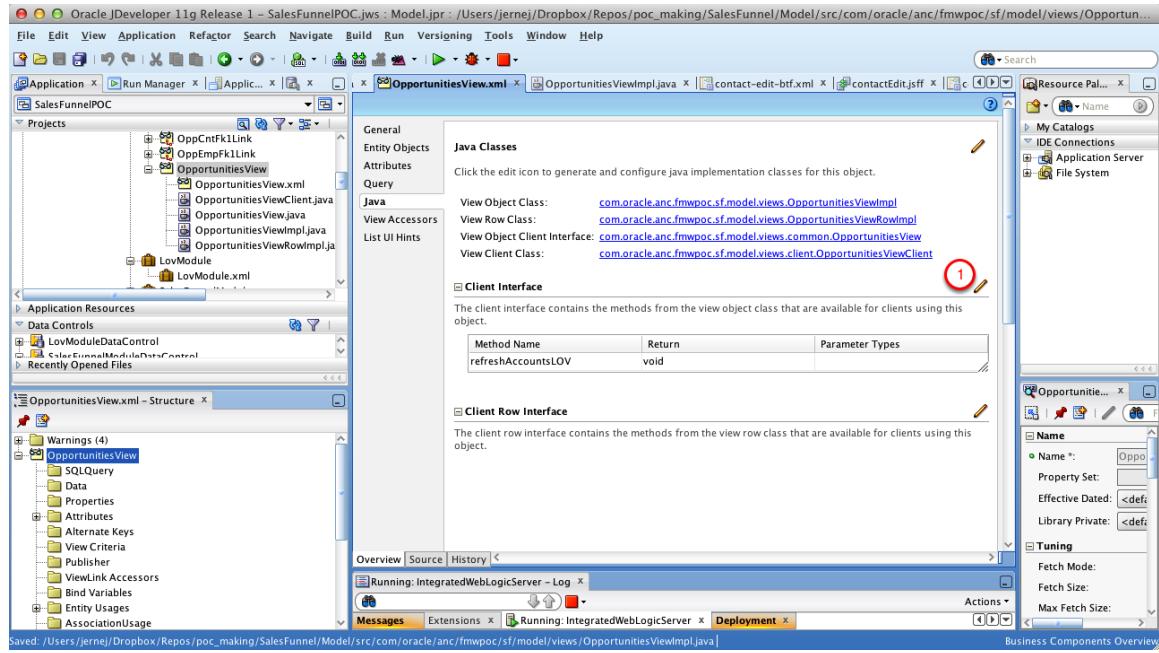
Append refreshContactsLOV method



1. Open OpportunitiesViewImpl.java
2. Append the method below
3. Open OpportunitiesView

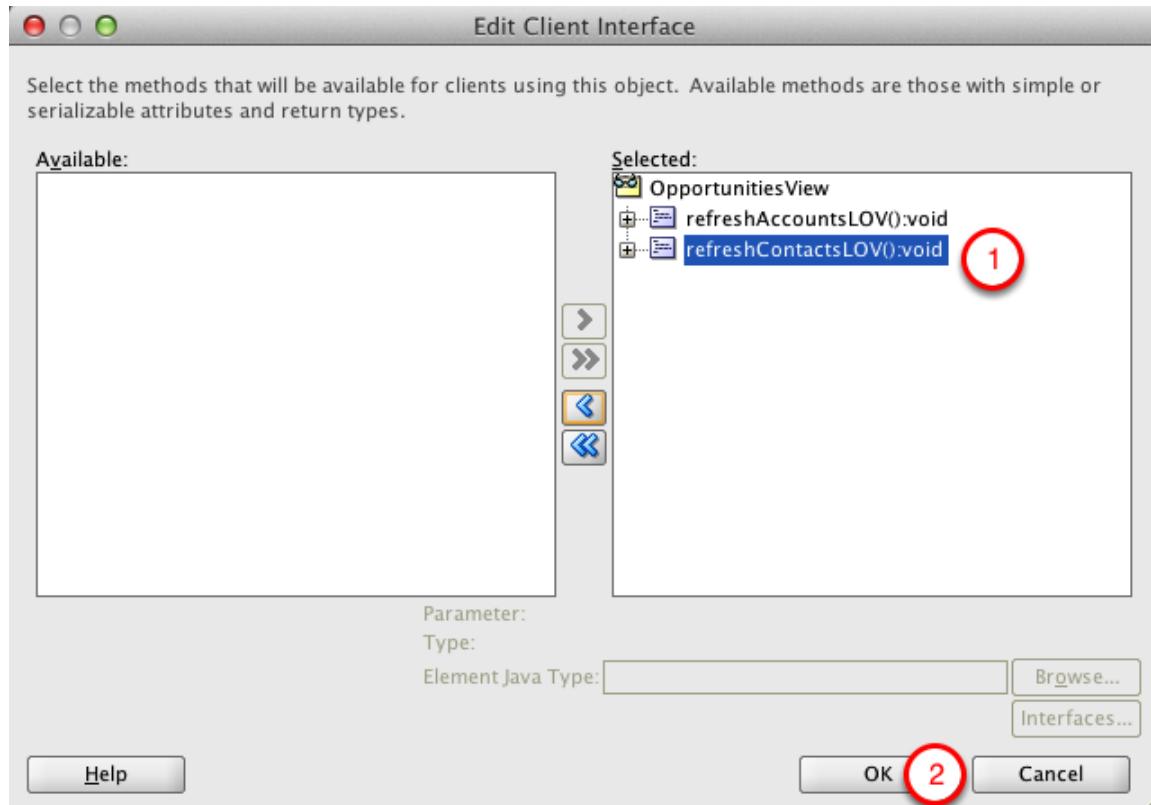
```
public void refreshContactsLOV(){
    OpportunitiesViewRowImpl currentRow =
        (OpportunitiesViewRowImpl)this.getCurrentRow();
    if(currentRow==null) return;
    currentRow.getLovModule_ContactsLovView1_1().executeQuery();
}
```

Add refreshContactsLOV to the interface



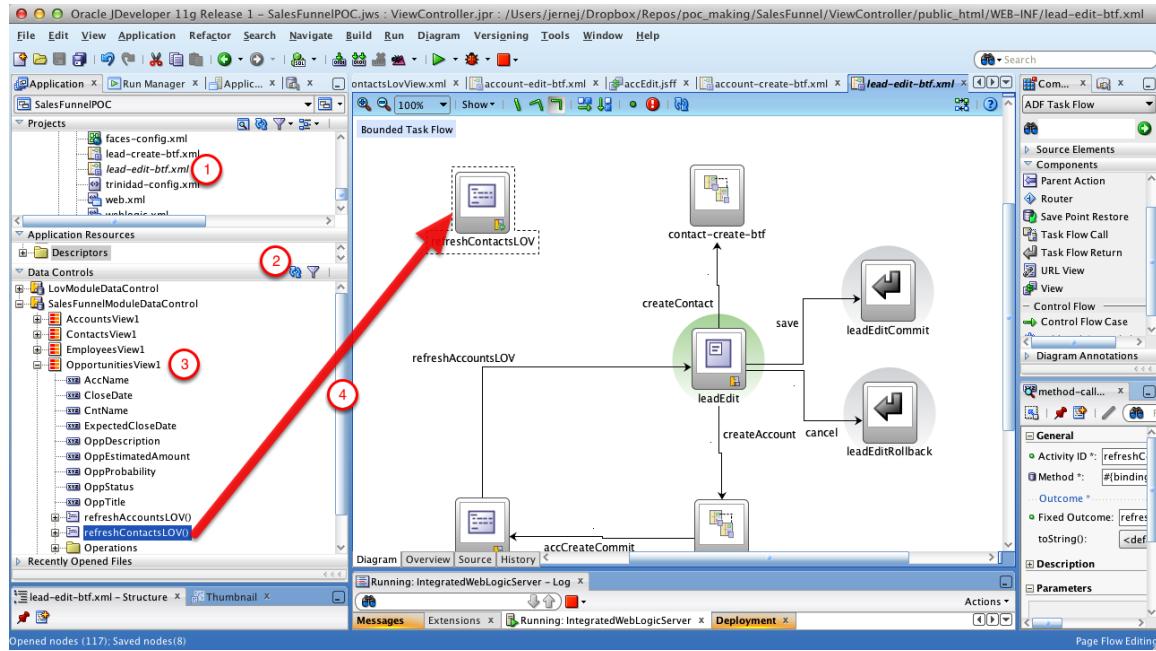
1. Click edit next to Client Interface

Edit Client Interface



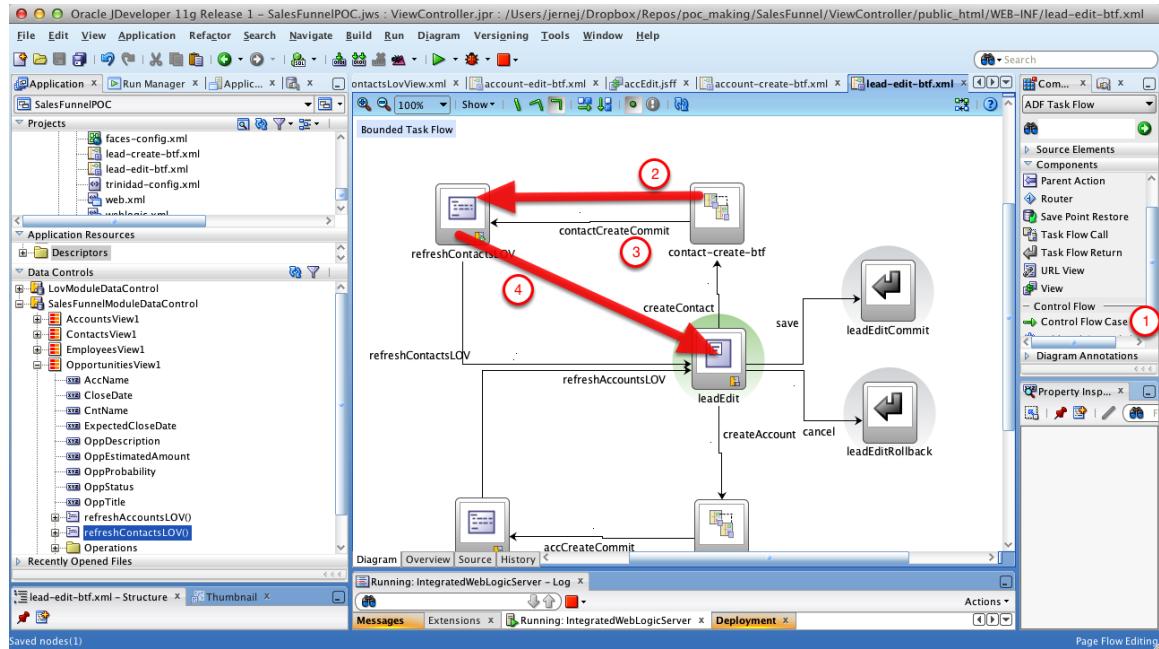
1. Slide refreshContactsLOV to the right
2. Click OK

Add refreshContactsLOV to lead-edit-btf



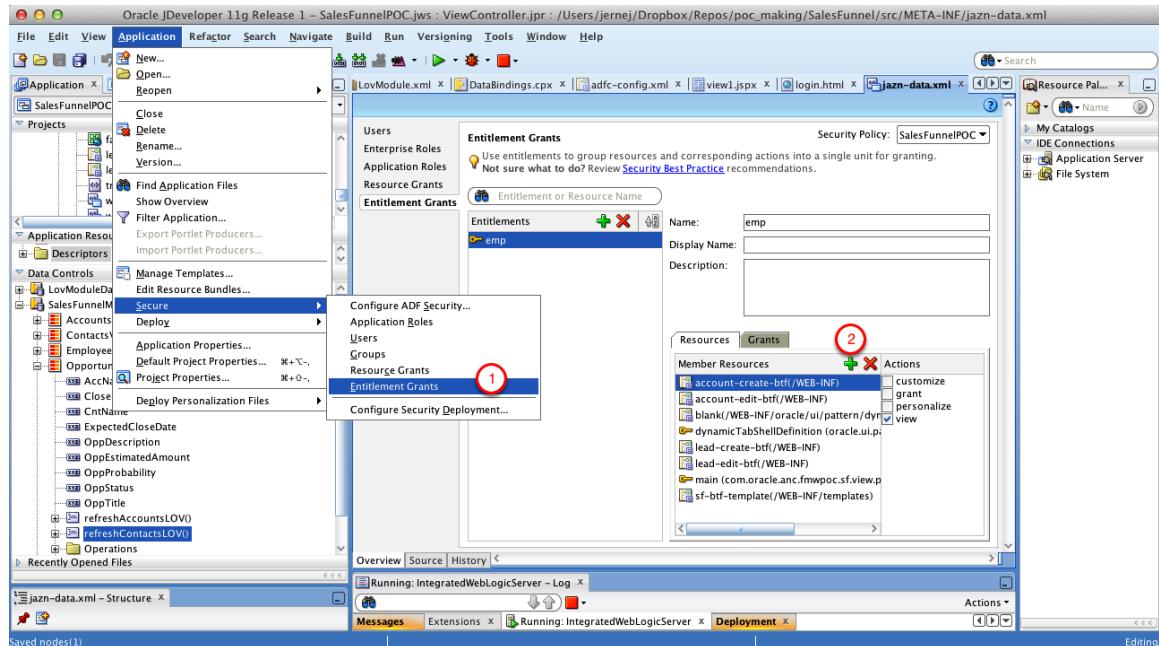
1. Open lead-edit-btf
2. Refresh Data Controls
3. Expand OpportunitiesView1
4. Drag and drop refreshContactsLOV to the diagram

Connect refreshContactsLOV



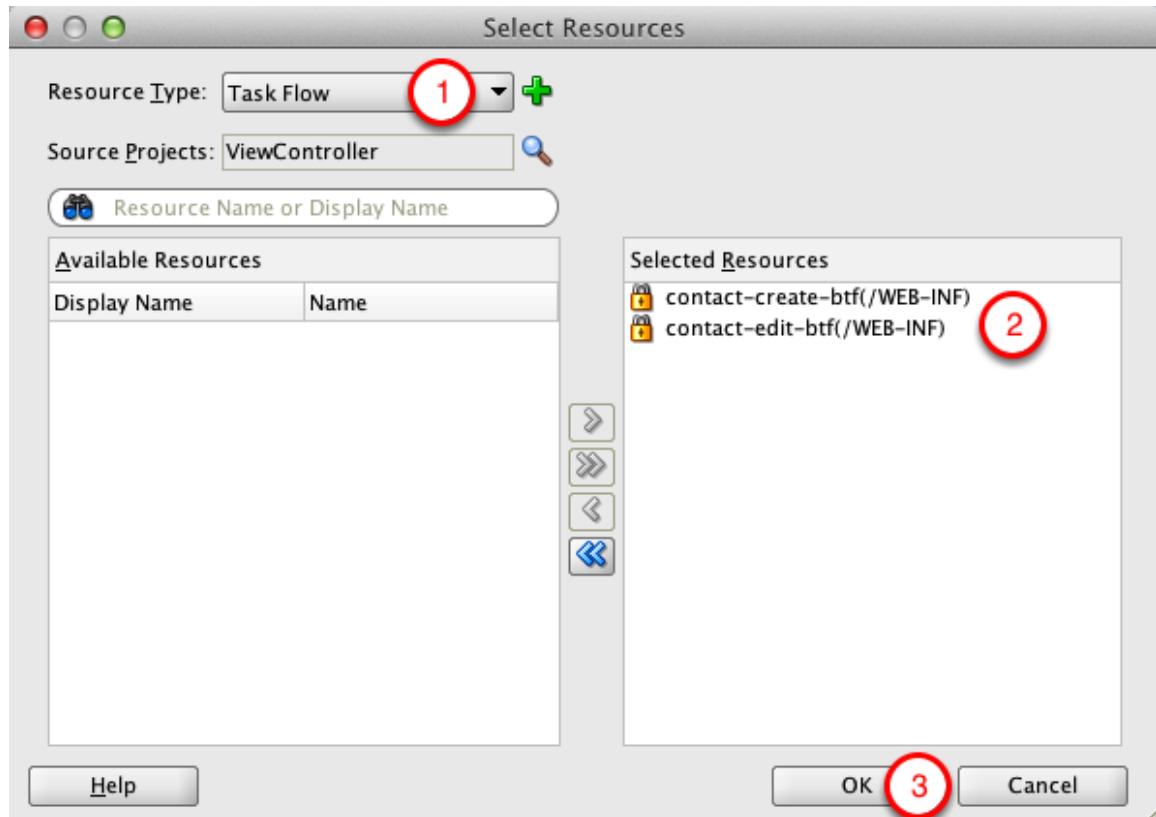
1. Select Control Flow Case in the Component Palette
2. Connect contact-create-btf to refreshContactsLOV
3. Make sure from outcome is contactCreateCommit
4. Connect refreshContactsLOV to leadEdit

Configure Security Grants



1. From the menu select Application > Secure > Entitlement Grants
2. When the page opens, click the plus in the Resources

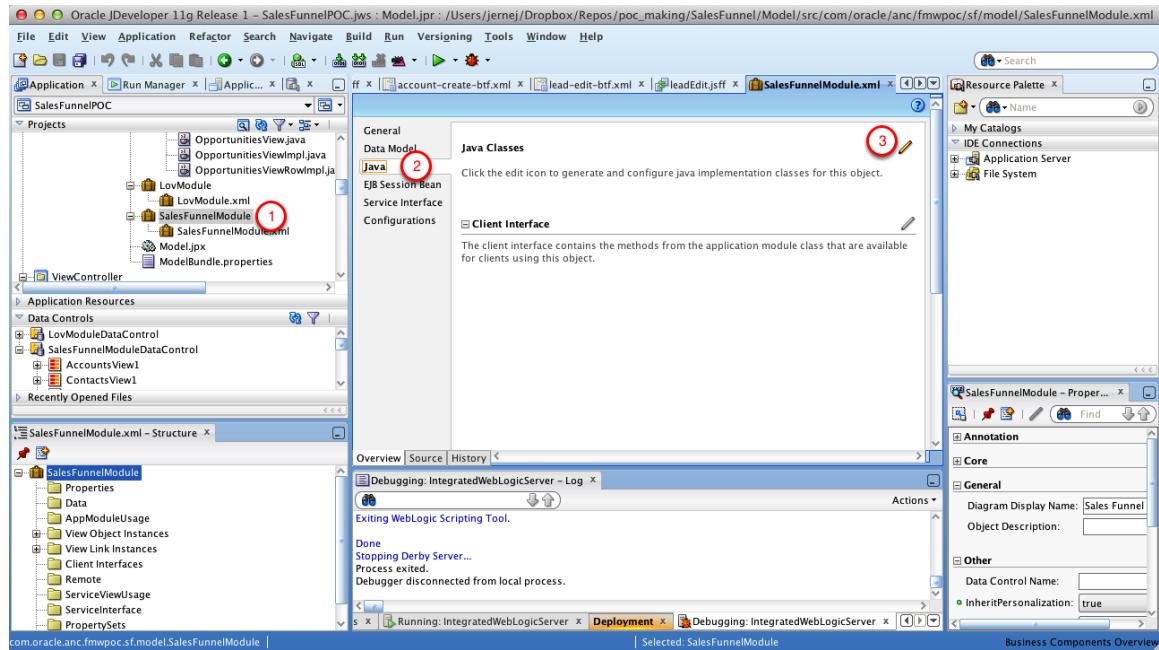
Select Resources



1. Select Task Flow as type
2. Slide contact-create-btf and contact-edit-btf to the right
3. Click OK

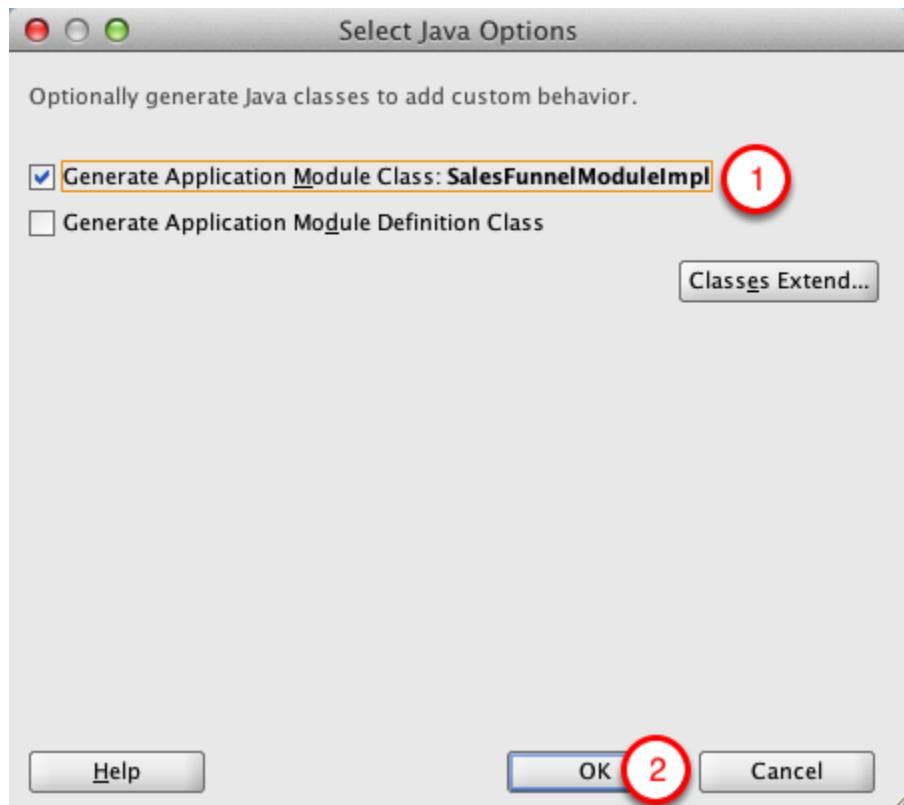
32. EmpId in Session

Create Java class for SalesFunnelModule



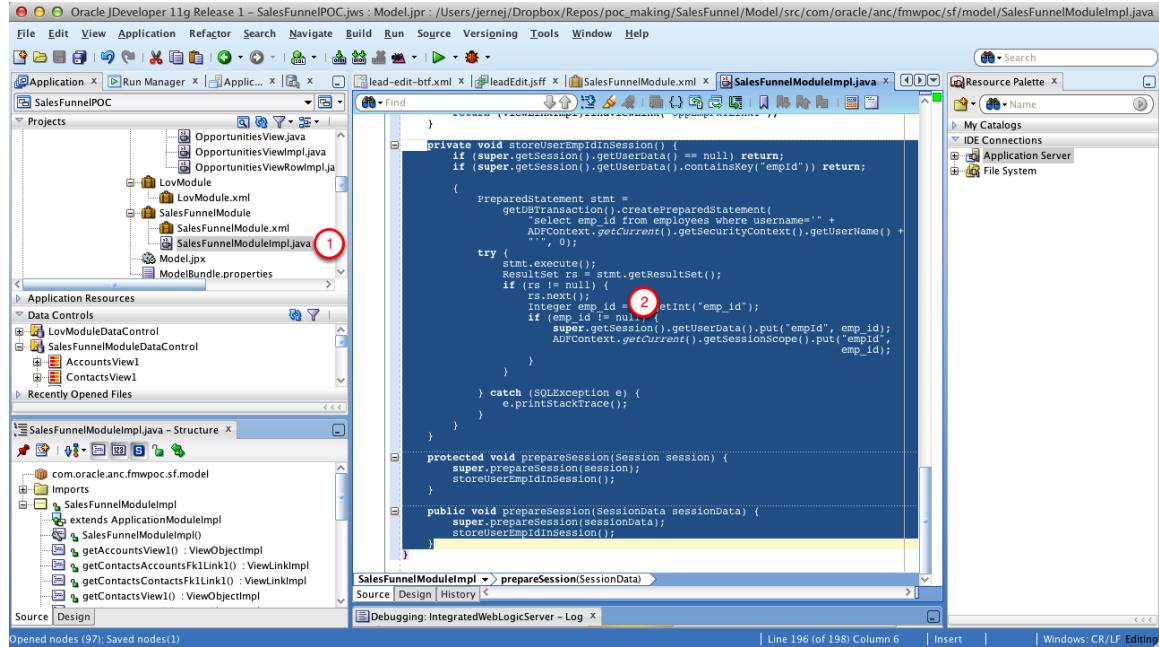
1. Open SalesFunnelModule
2. Open Java Tab
3. Click Edit Icon

Select Java Options



1. Check "Generate Application Module Class"
2. Click OK

Add implementation



1. Open SalesFunnelModuleImpl.java
2. Append the code below at the end of class implementation

```

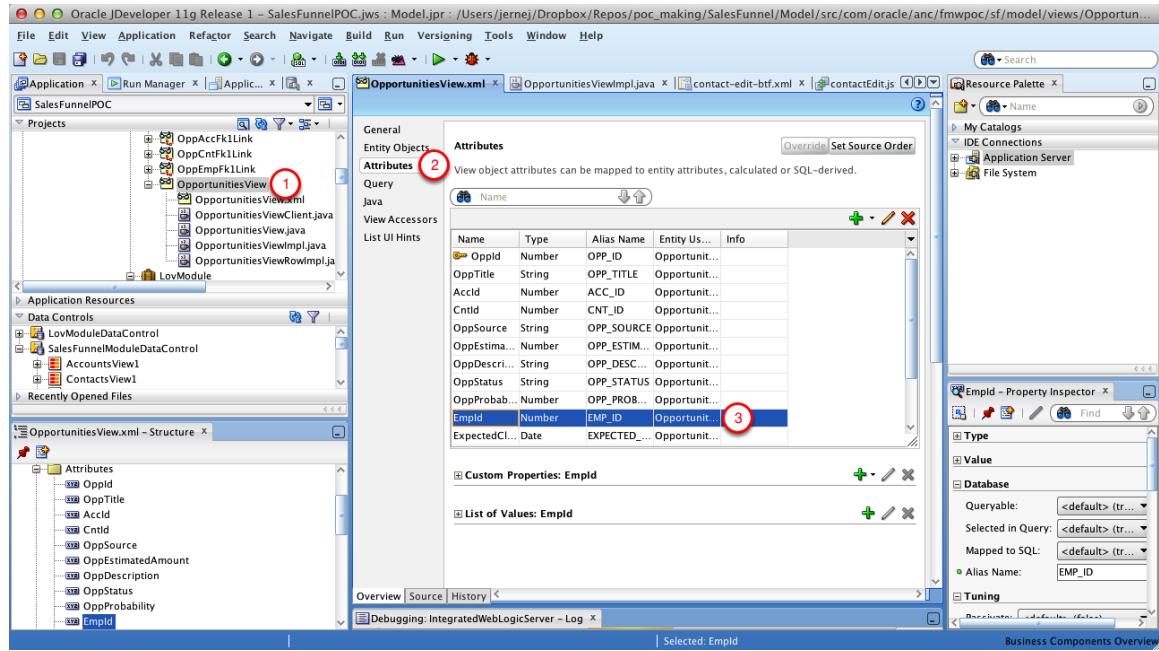
private void storeUserEmpIdInSession() {
    if (super.getSession().getUserData() == null) return;
    if (super.getSession().getUserData().containsKey("empId")) return;

    {
        PreparedStatement stmt =
            getDBTransaction().createPreparedStatement(
                "select emp_id from employees where username='"
                    + ADFContext.getCurrent().getSecurityContext().getUserName() +
                "'", 0);
        try {
            stmt.execute();
            ResultSet rs = stmt.getResultSet();
            if (rs != null) {
                rs.next();
                Integer emp_id = rs.getInt("emp_id");
                if (emp_id != null) {
                    super.getSession().getUserData().put("empId", emp_id);
                    ADFContext.getCurrent().getSessionScope().put("empId",
                        emp_id);
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    protected void prepareSession(Session session) {
        super.prepareSession(session);
        storeUserEmpIdInSession();
    }
}

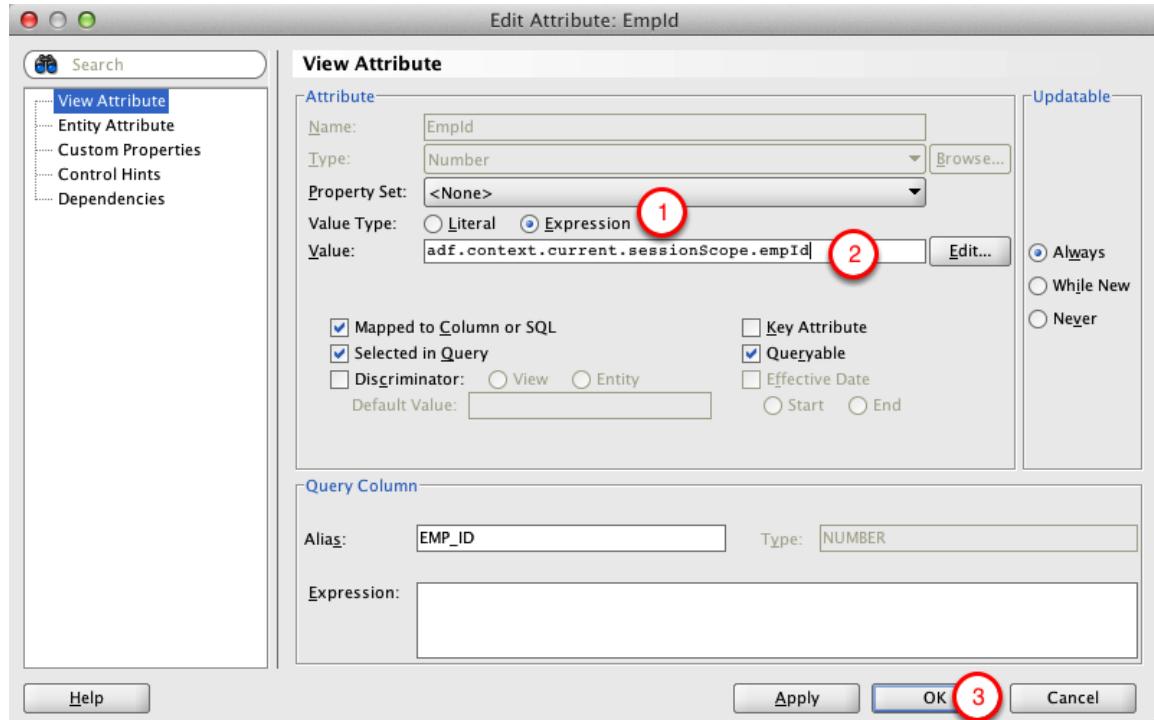
```

Open Empld attribute



1. Open OpportunitiesView
2. Open Attributes tab
3. Double-click Empld attribute

Edit Attribute: Empld

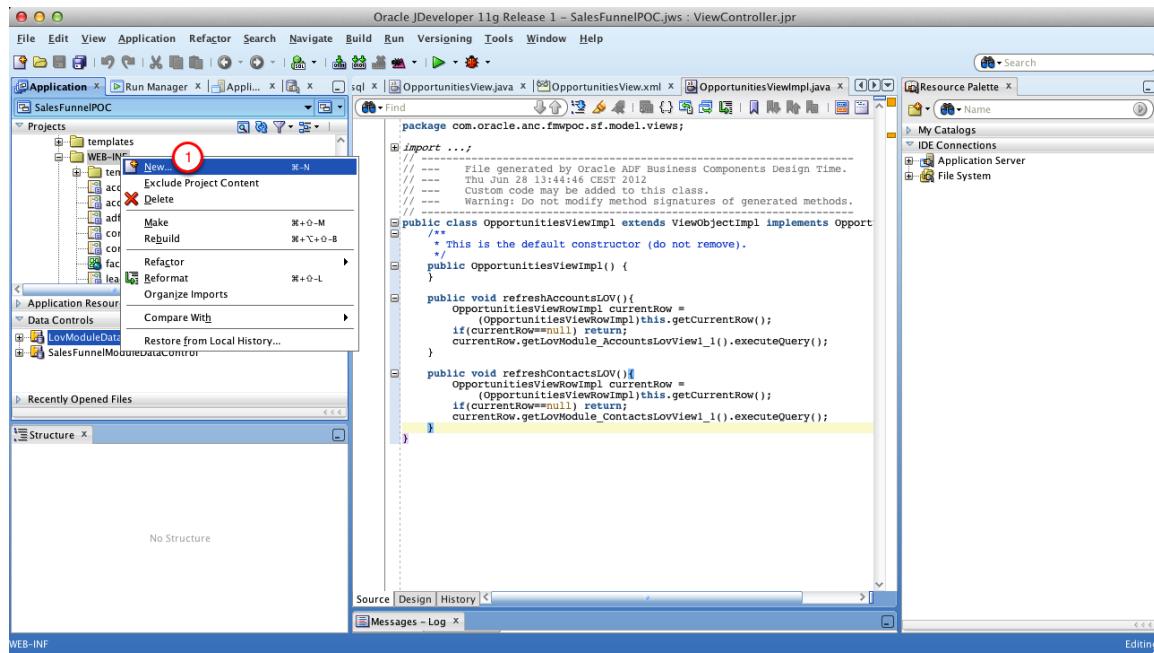


1. Select Expression
2. Enter adf.context.current.sessionScope.empld
3. Click OK

IV. Implementing Dashboard

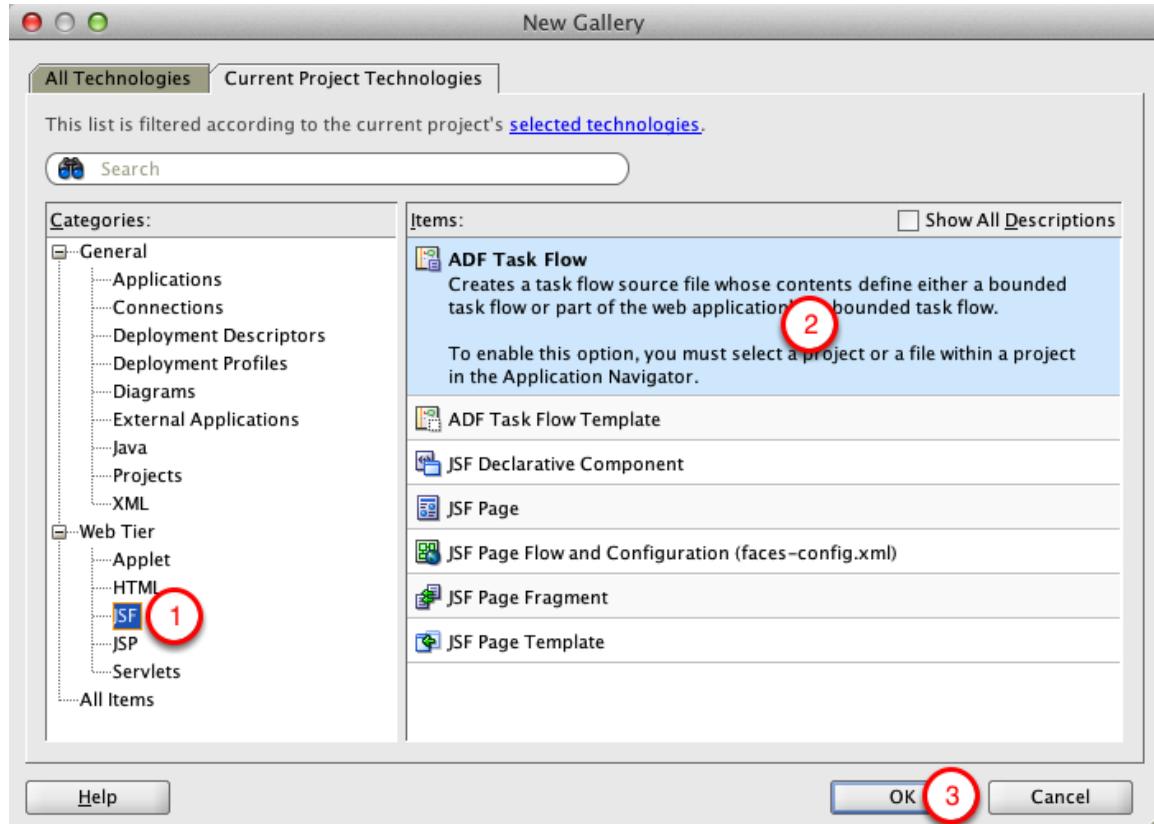
33. Dashboard

Create New Task Flow



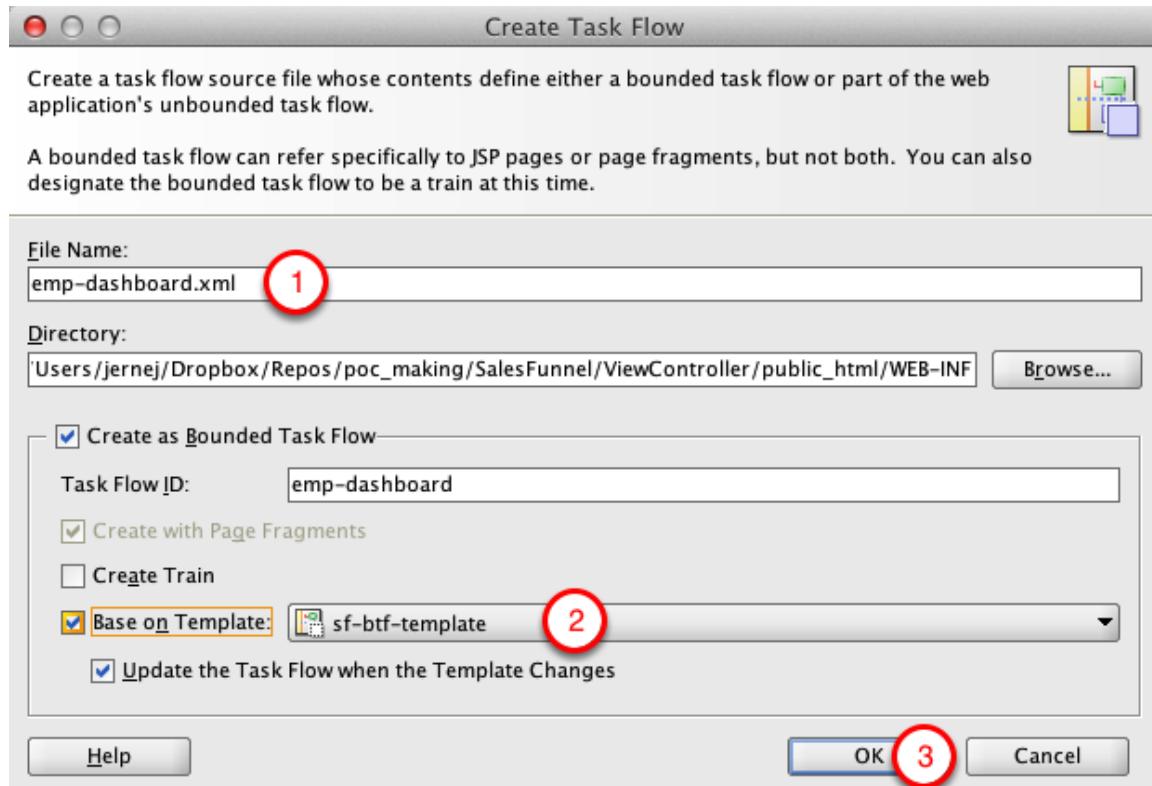
1. Right-click WEB-INF, select New

New Gallery



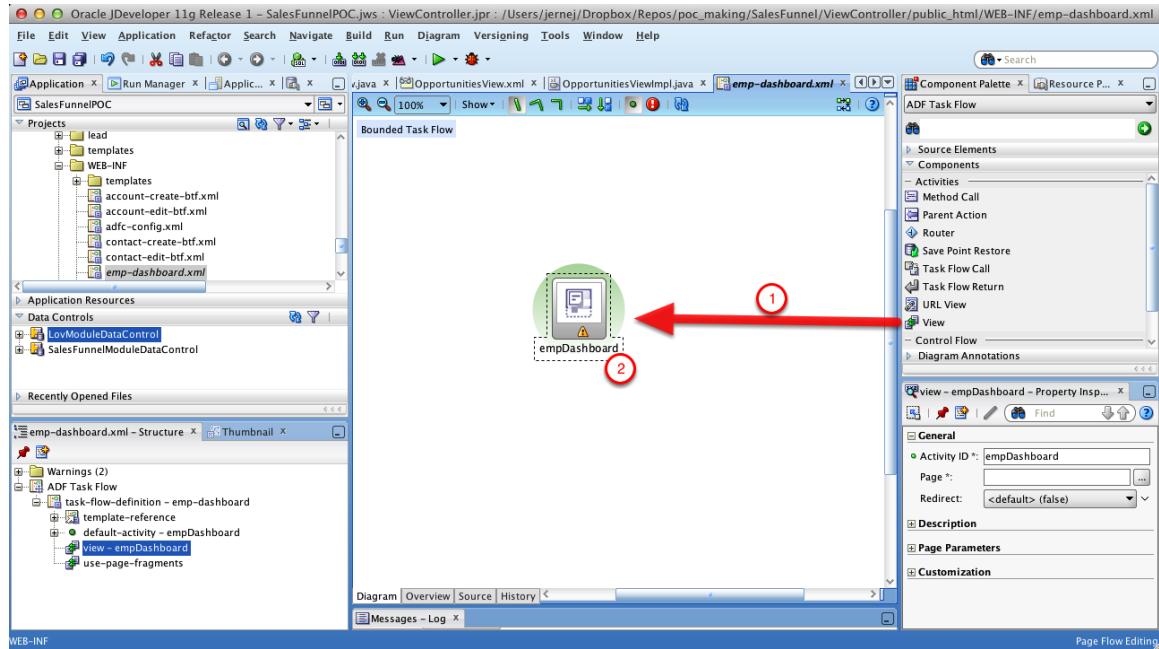
1. Select Web Tier > JSF Category
2. Select ADF Task Flow
3. Click OK

Create Task Flow



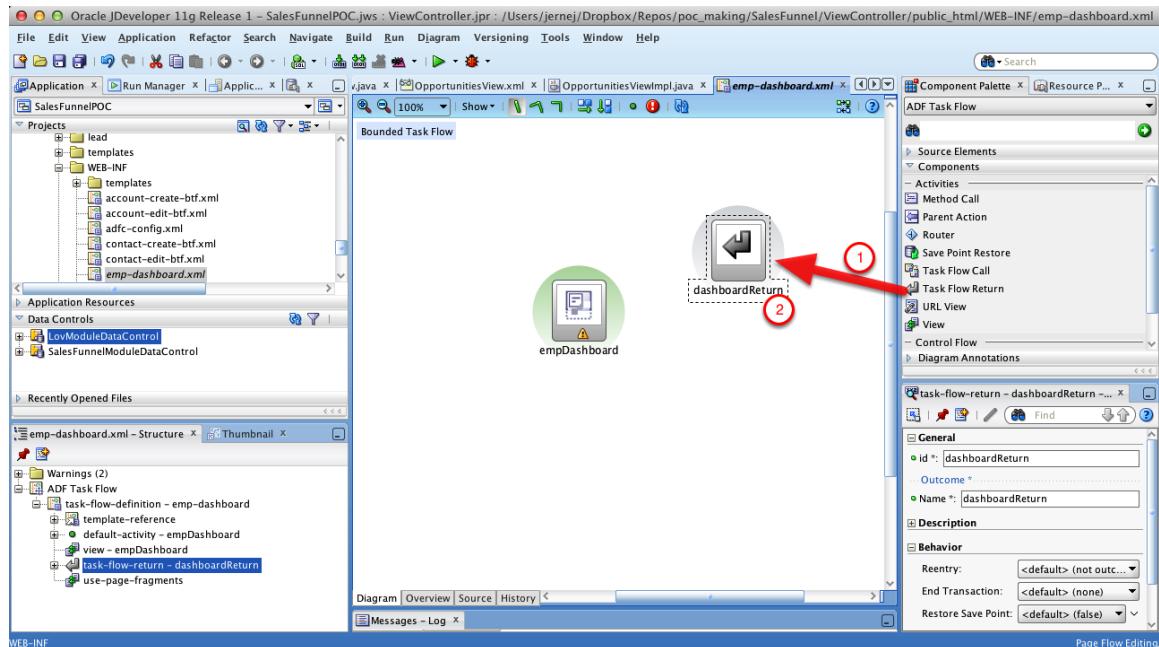
1. Name it emp-dashboard-btf.xml
2. Base it on sf-btf-template
3. Click OK

Add empDashboard View



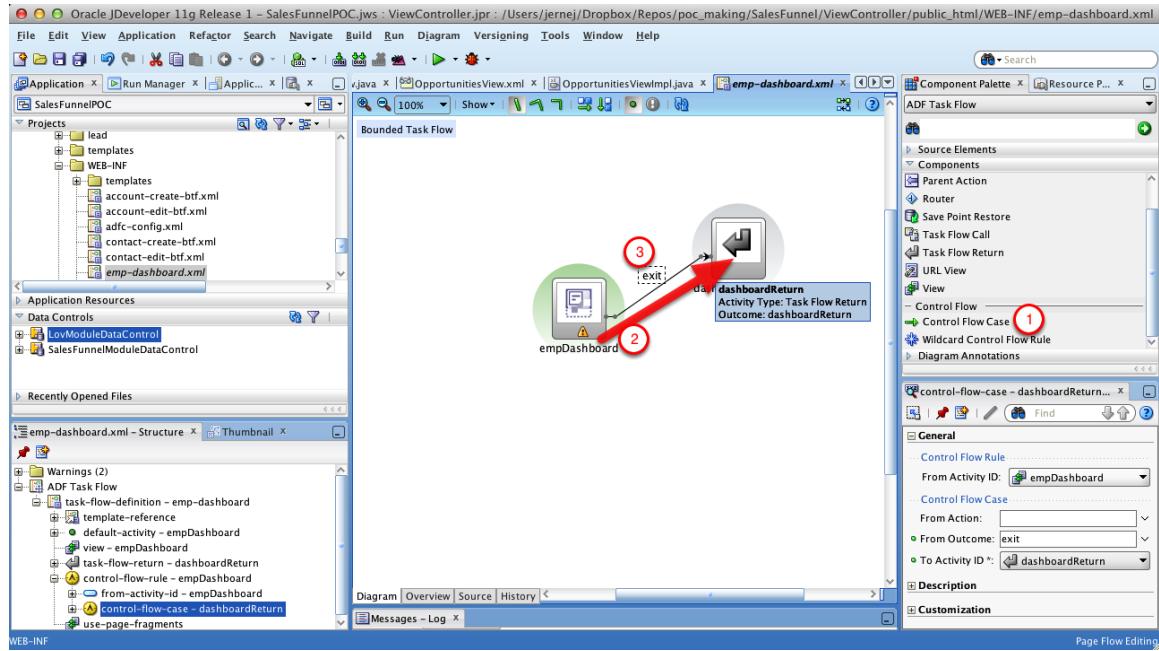
1. Drag and drop View on the page
2. Name it empDashboard

Add dashboardReturn Task Flow Return



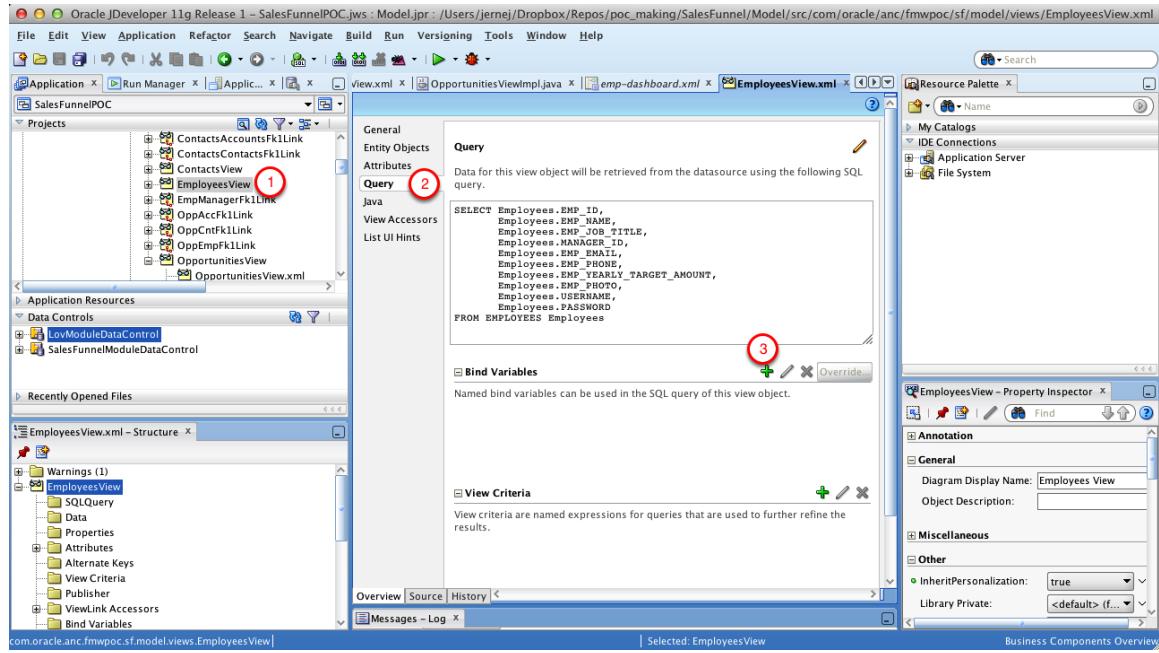
1. Drag and drop Task Flow Return on the page
2. Name it dashboardReturn

Connect empDashboard to dashboardReturn



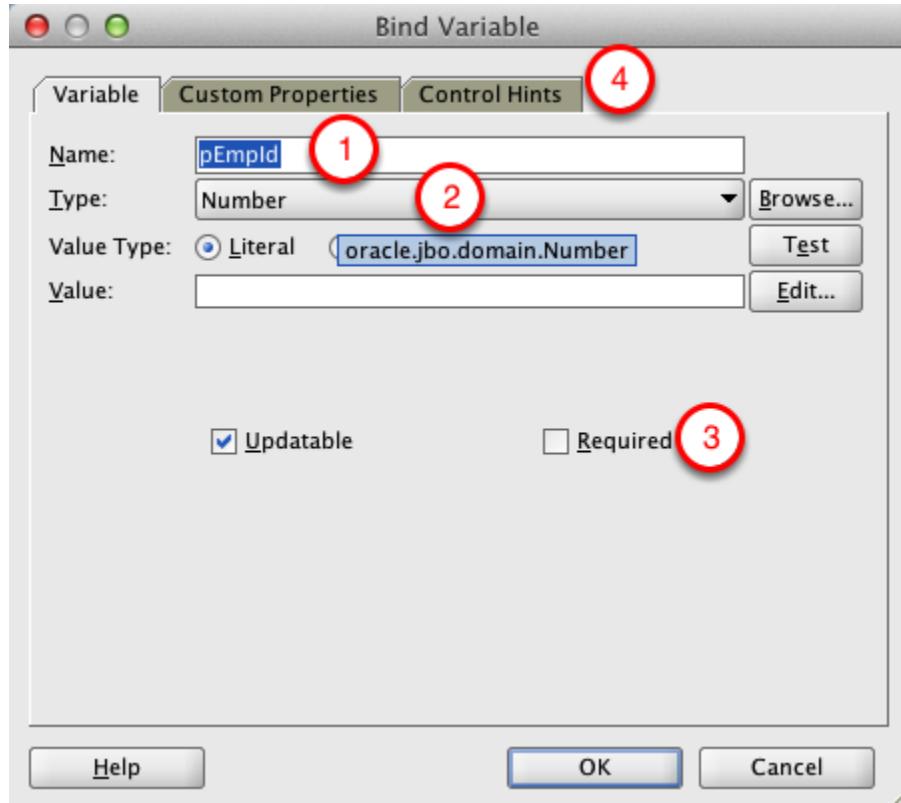
1. Select Control Flow Case in the Component Palette
2. Connect empDashboard to dashboardReturn
3. Name the action "exit"

Add variable to EmployeesView



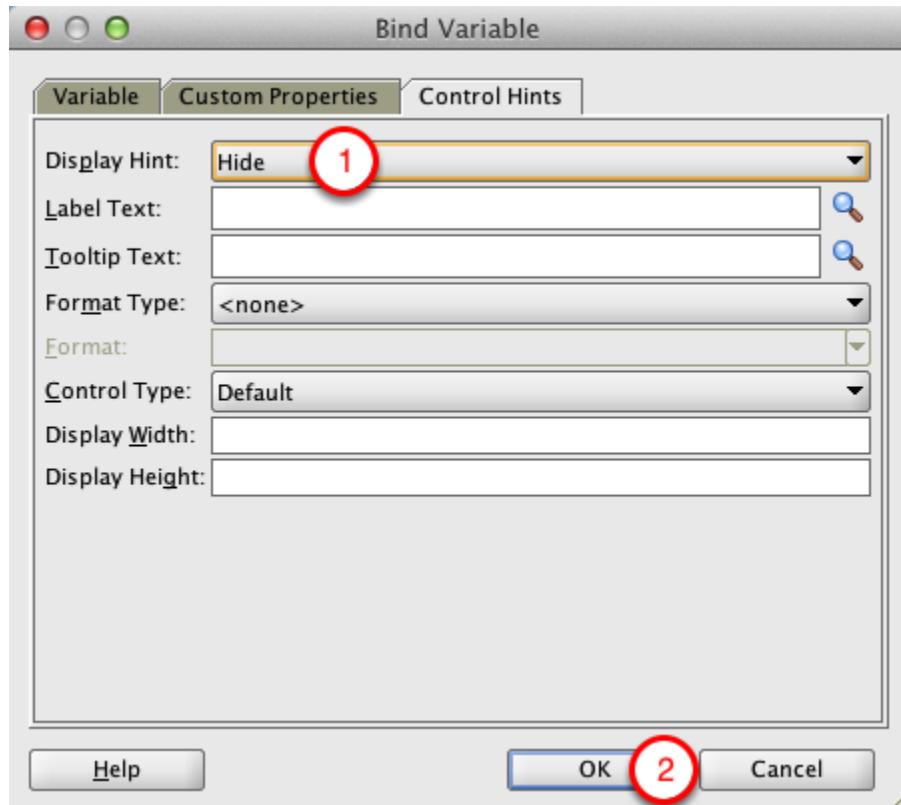
1. Locate and open EmployeesView
2. Open the Query tab
3. Click the plus next to Bind Variables

Bind Variable



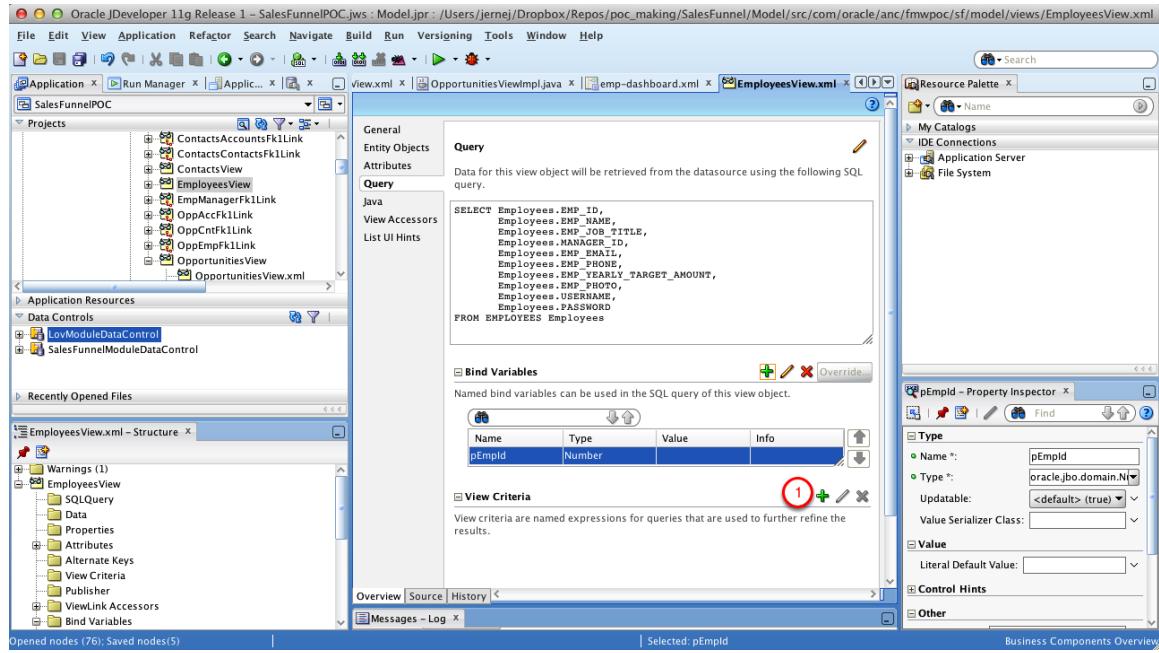
1. Name the variable pEmpId
2. Select Number as type
3. Uncheck Required
4. Open Control Hints tab

Bind Variable



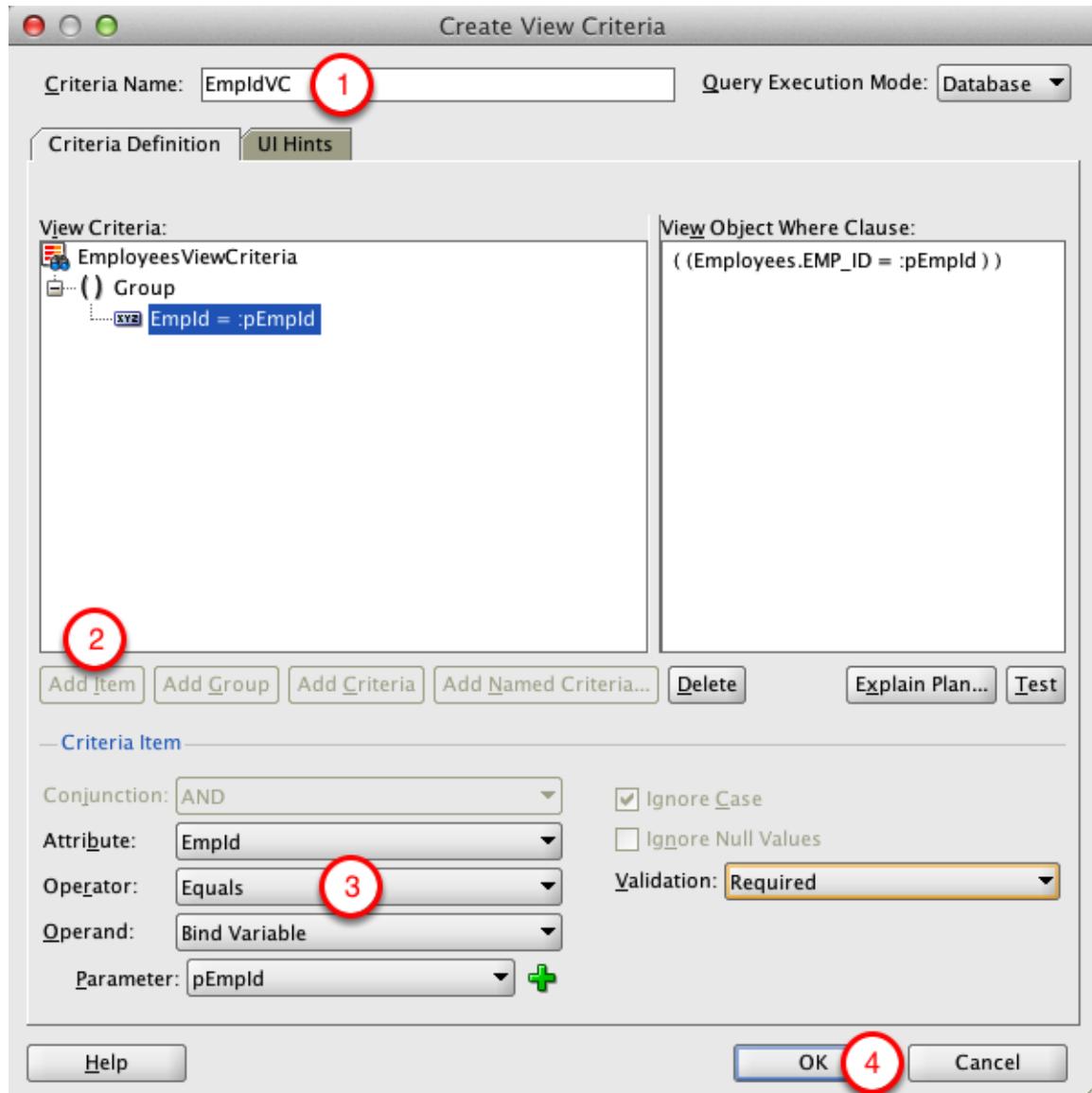
1. Set Display Hint to Hide
2. Click OK

Add View Criteria to EmployeesView



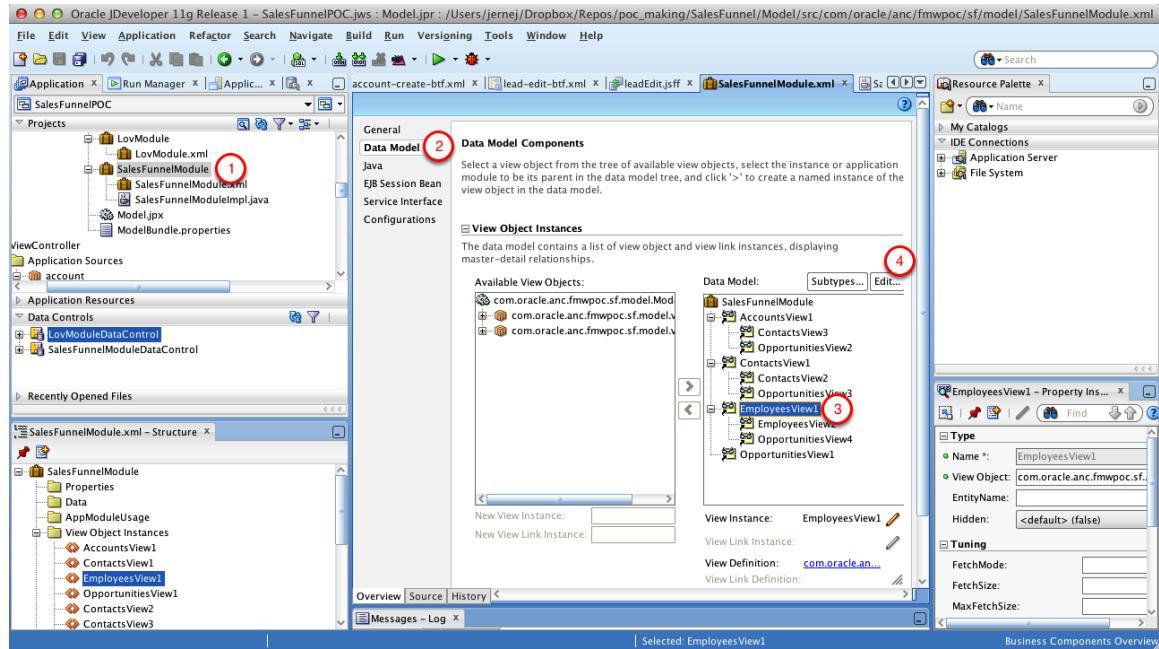
1. Click the plus next to View Criteria

Create View Criteria



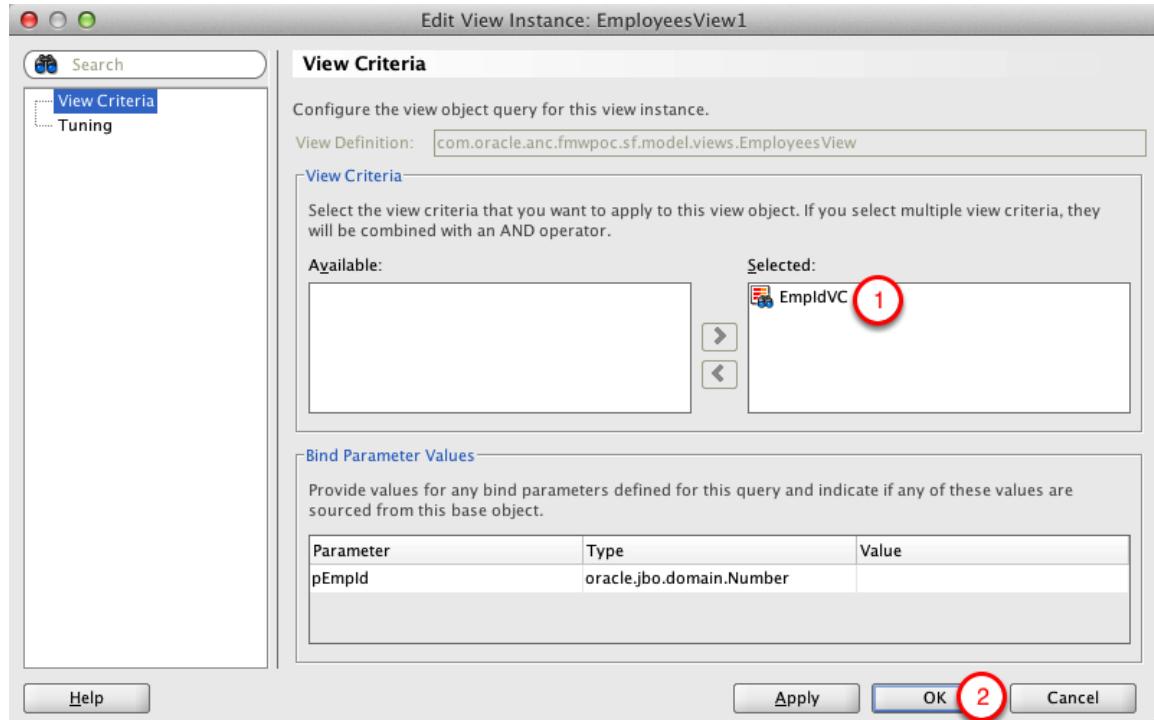
1. Name it EmplIdVC
2. Click Add Item
3. Set
 - Attribute: EmplId
 - Operator: Equals
 - Operand: Bind Variable
 - Parameter: pEmplId
 - Validation: Required
4. Click OK

Add View Criteria as default filter on EmployeesView1



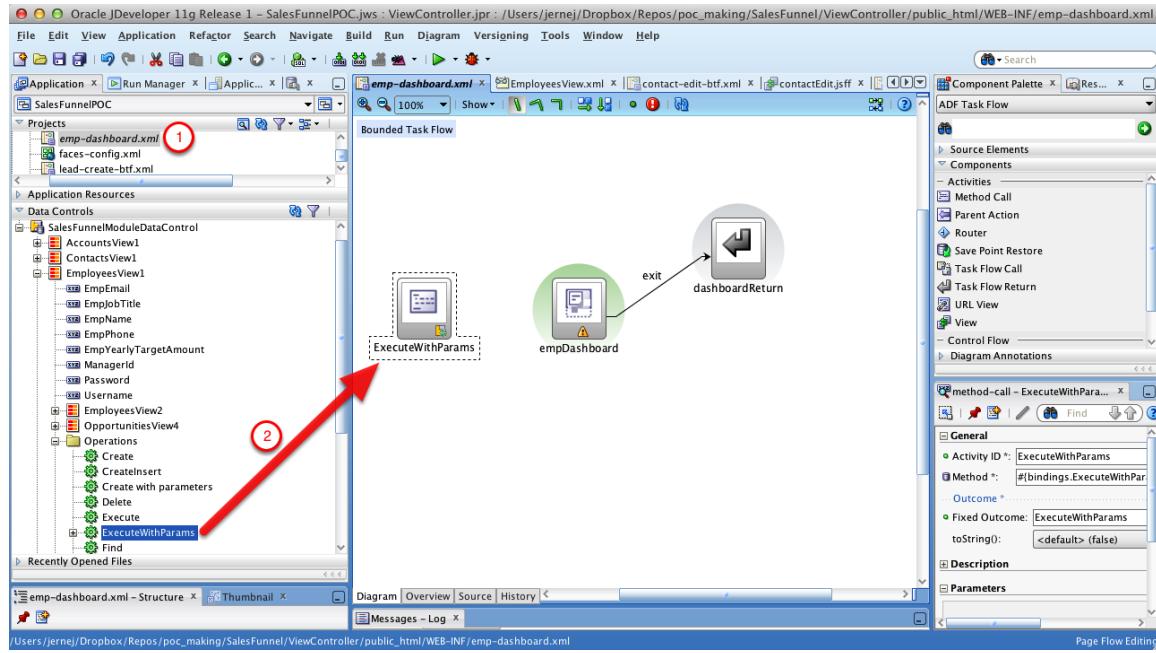
1. Click to open SalesFunnelModule
2. Open Data Model tab
3. Select EmployeesView1
4. Click Edit button

Edit View Instance: EmployeesView1



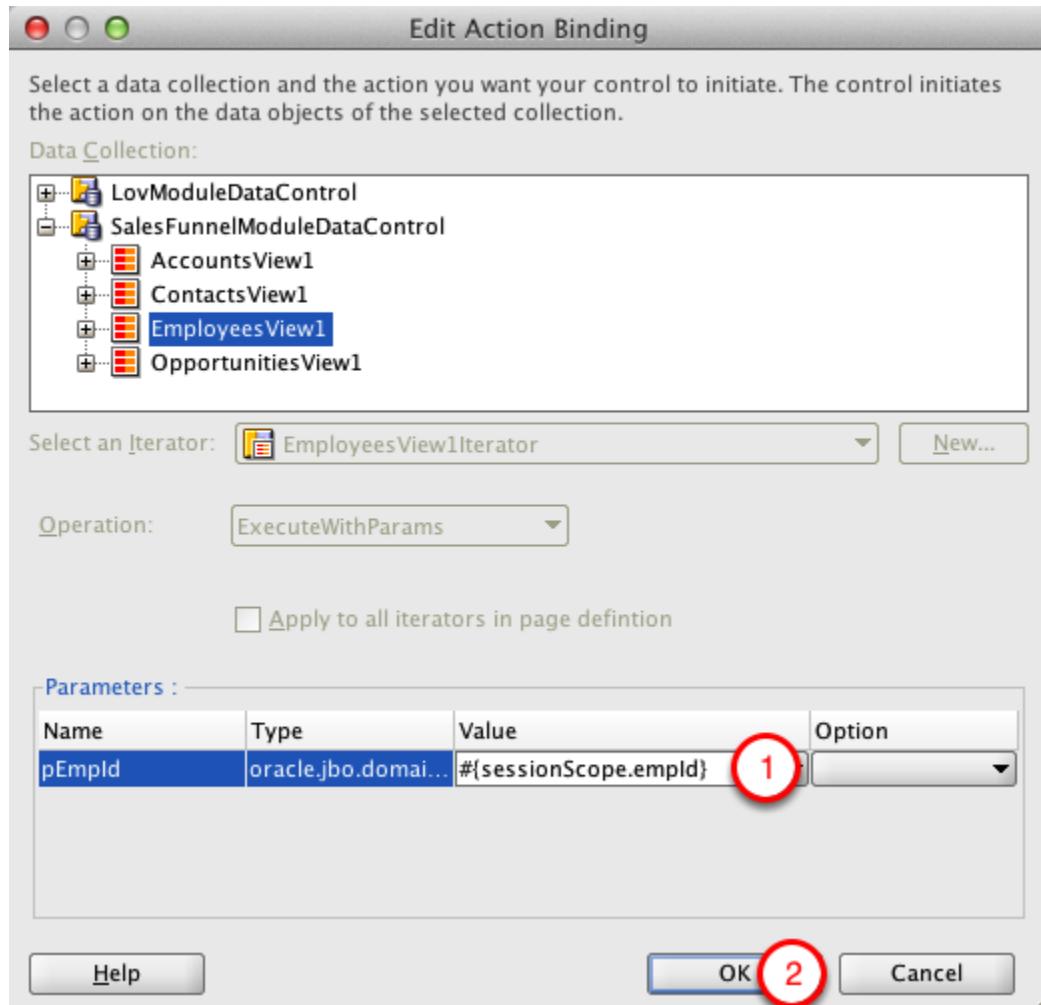
1. Slide EmplIdVC to the right
2. Click OK

Add EmployeesView1 ExecuteWithParams to emp-dashboard



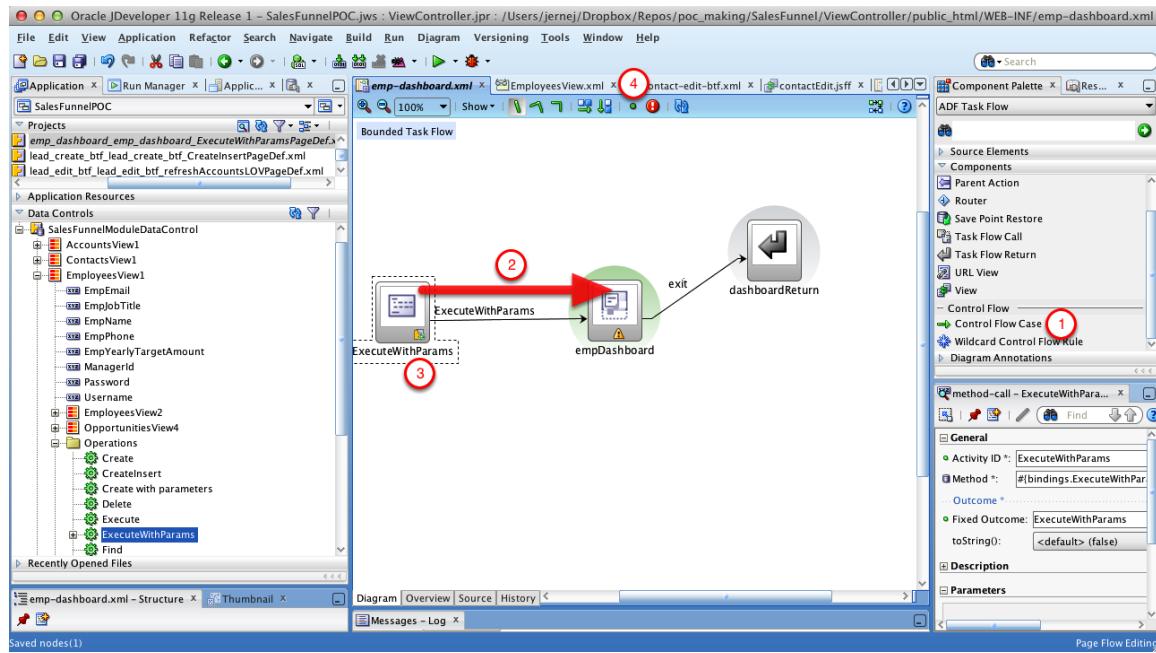
1. Open emp-dashboard
2. Expand EmployeesView1 in Data Controls and find ExecuteWithParams, drag and drop it to the diagram

Edit Action Binding



1. Enter #\${sessionScope.empld} as Value
2. Click OK

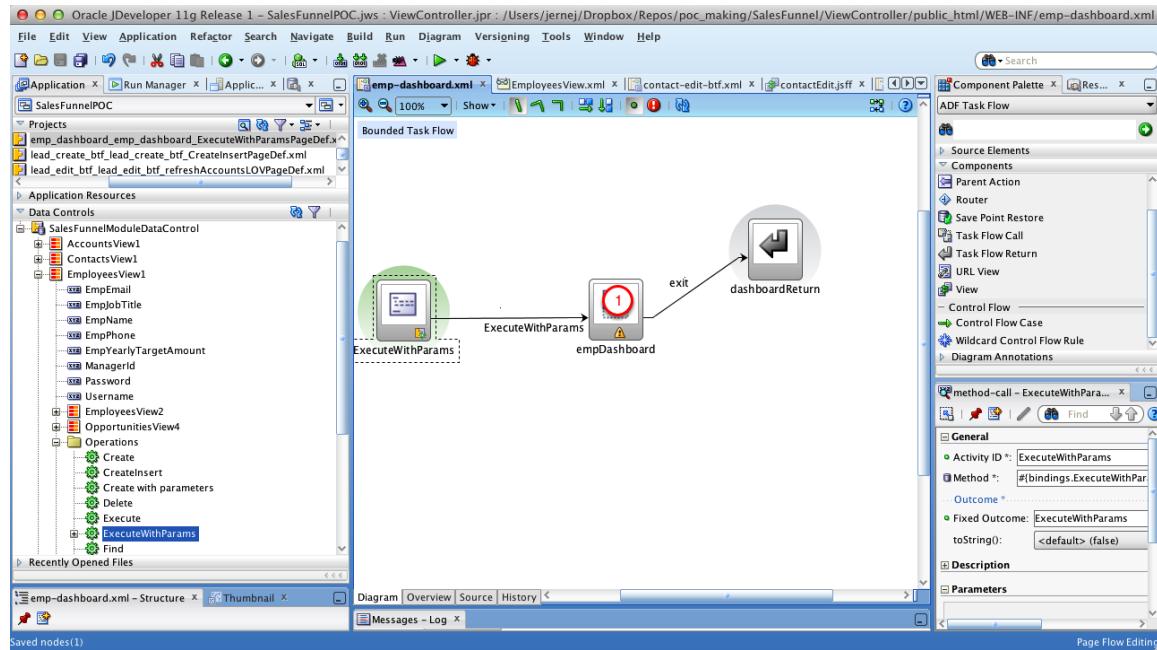
Connect ExecuteWithParams to empDashboard



1. Select Control Flow Case in the Palette
2. Connect ExecuteWithParams to the dashboard
3. Select ExecuteWithParams activity
4. Click on the green dot, to make ExecuteWithParams default activity

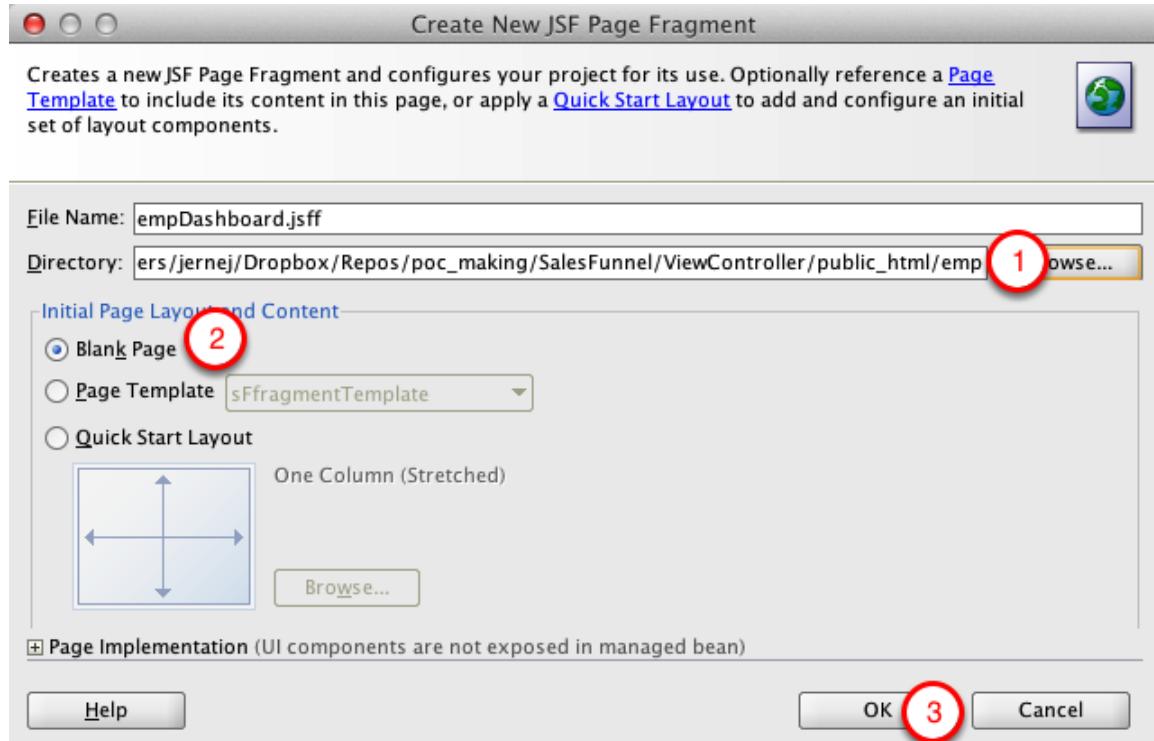
34. Implement Dashboard page

Implement Dashboard page



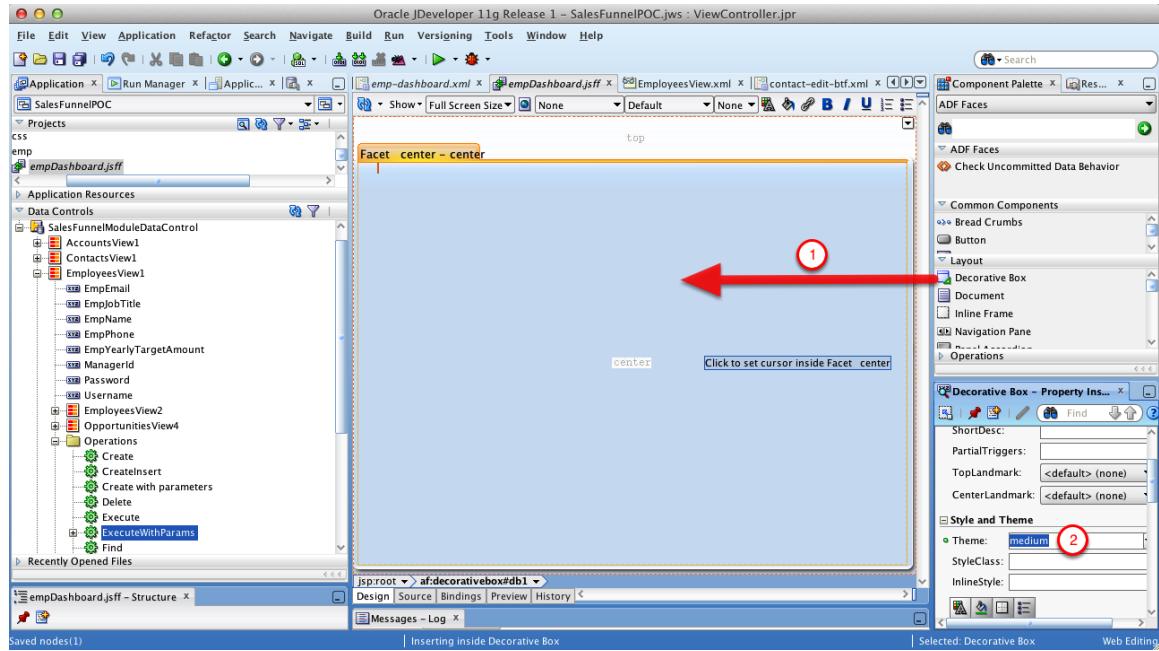
1. Double-click on empDashboard view

Create New JSF Page Fragment



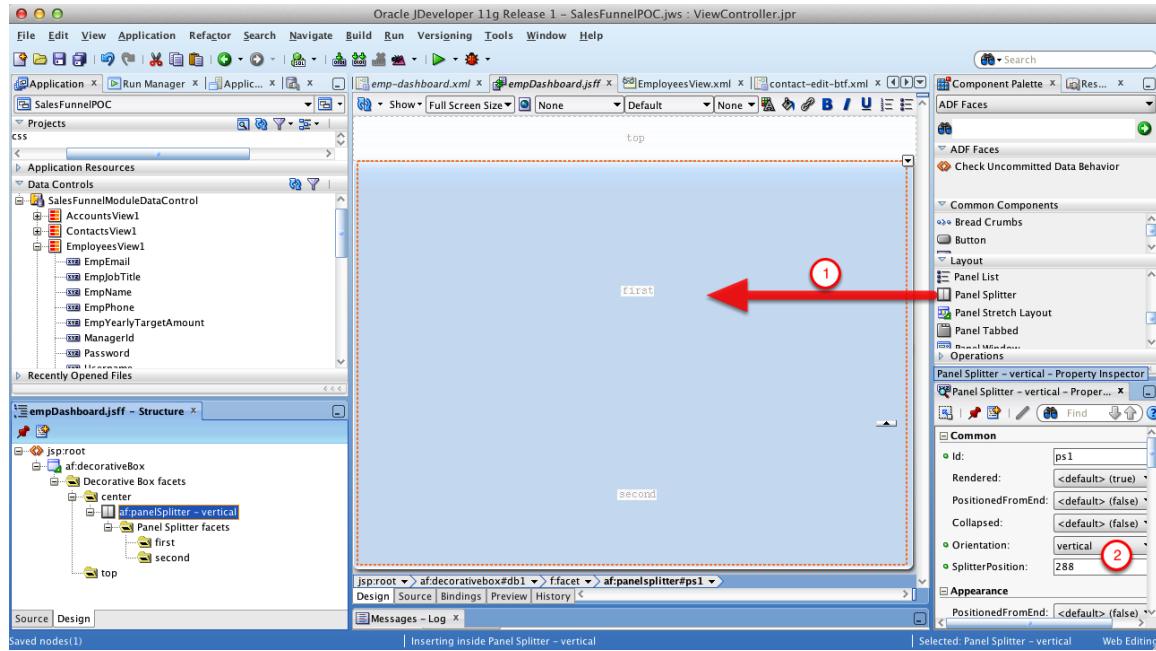
1. Append /emp to the directory
2. Select Blank Page, we will create a page from scratch
3. Click OK

Add DecorativeBox



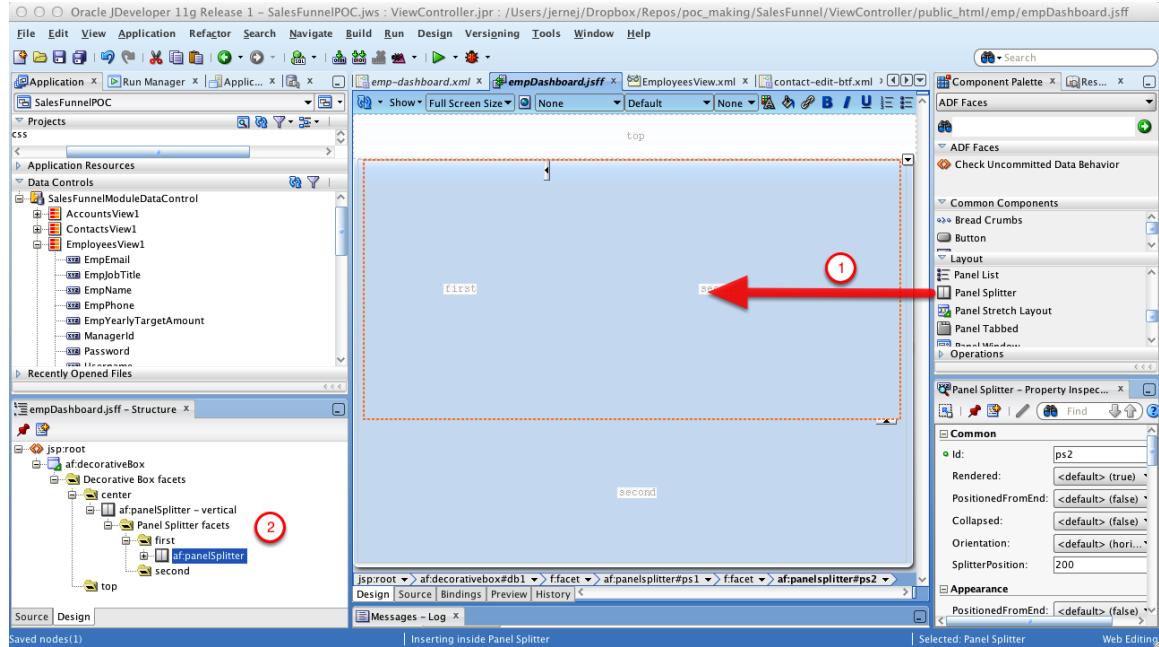
1. Drag and drop Decorative Box to the page
2. Change Theme to medium

Add Panel Splitter



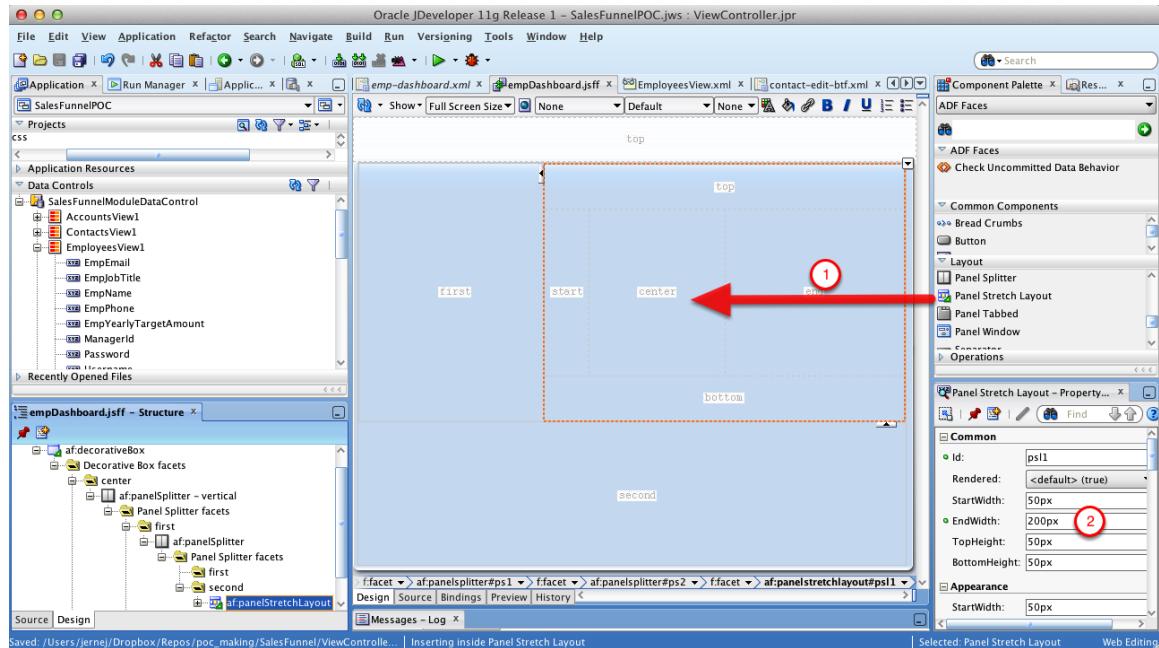
1. Drag and drop Panel Splitter to the Decorative Box center facet
2. Change Properties:
 - Orientation: Vertical
 - Splitter Position: 288

Add another Panel Splitter



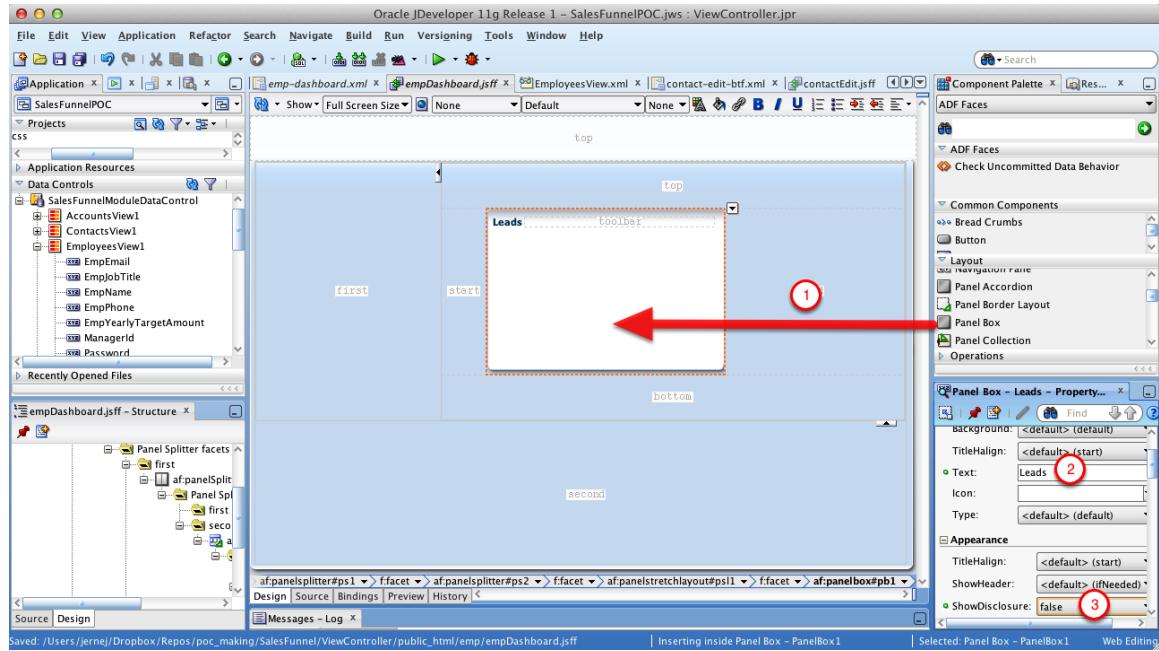
1. Drag and drop another Panel Splitter to the first facet of the first Splitter
2. Make sure your structure looks like the one on the screenshot

Add Panel Stretch Layout



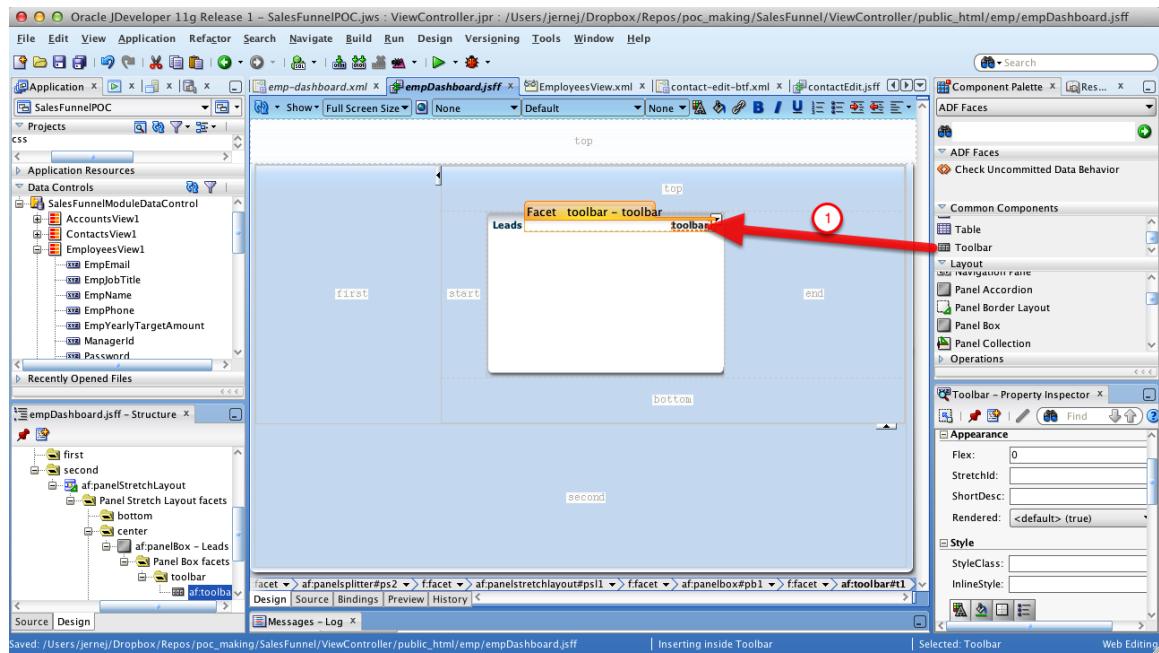
1. Drag and drop Panel Stretch Layout to the second facet of the second splitter
2. Change EndWidth property to 200px

Add Panel Box



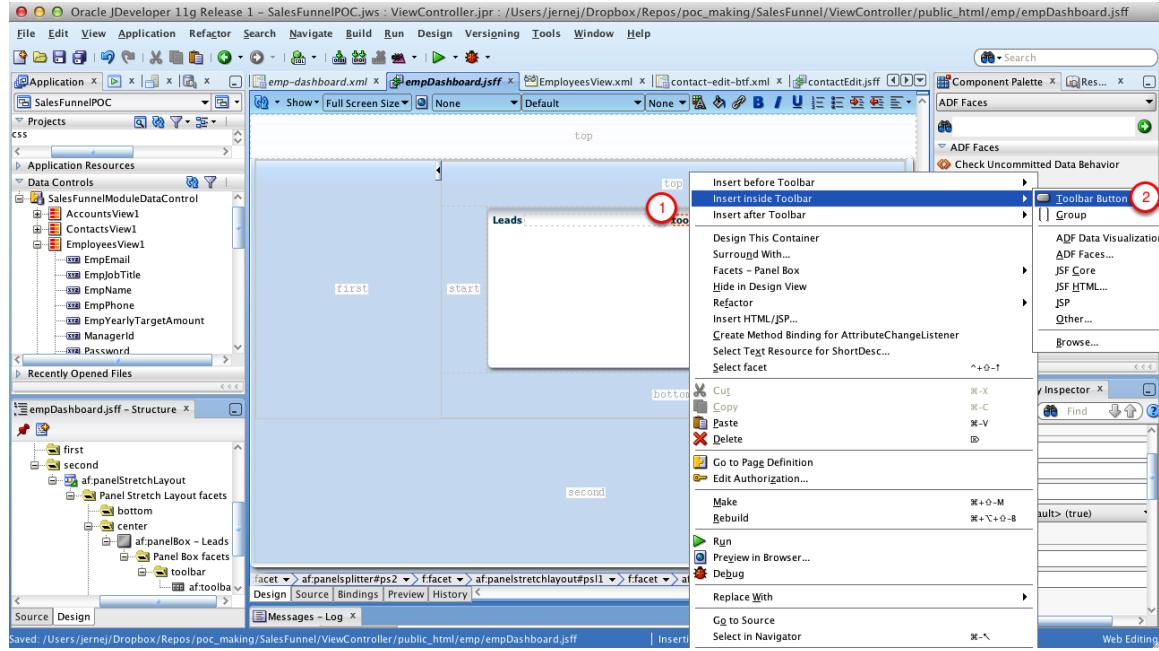
1. Drag and drop Panel Box to the center facet of Panel Stretch Layout
2. Set Text to Leads
3. Set ShowDisclosure to false

Add Toolbar



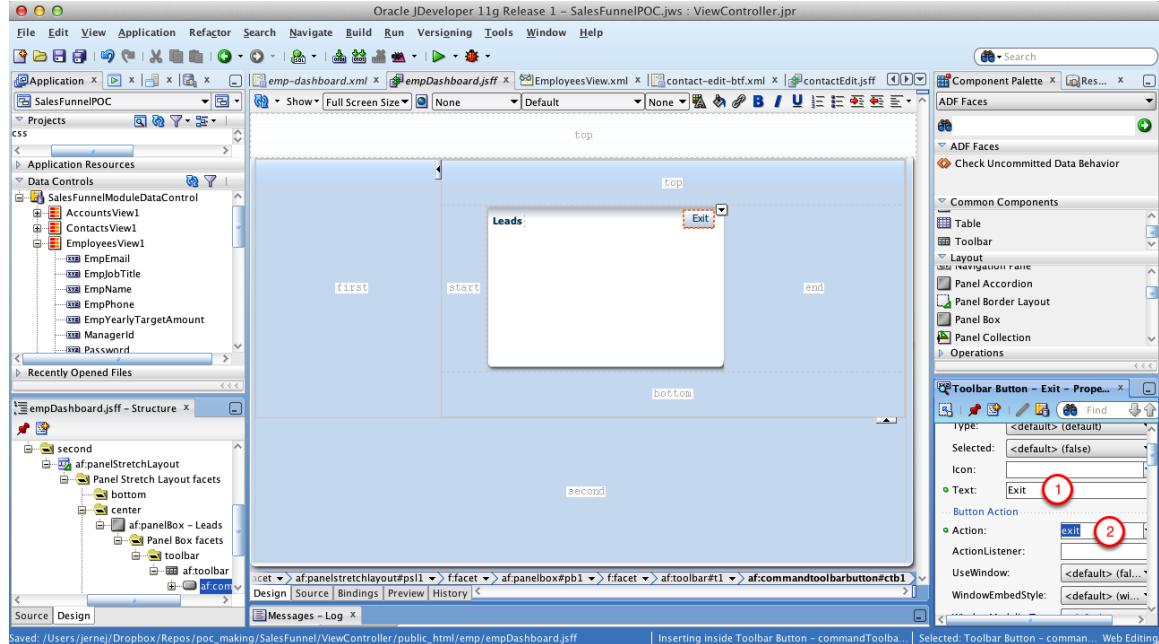
1. Drag and drop Toolbar to the toolbar facet of Panel Box

Add Button



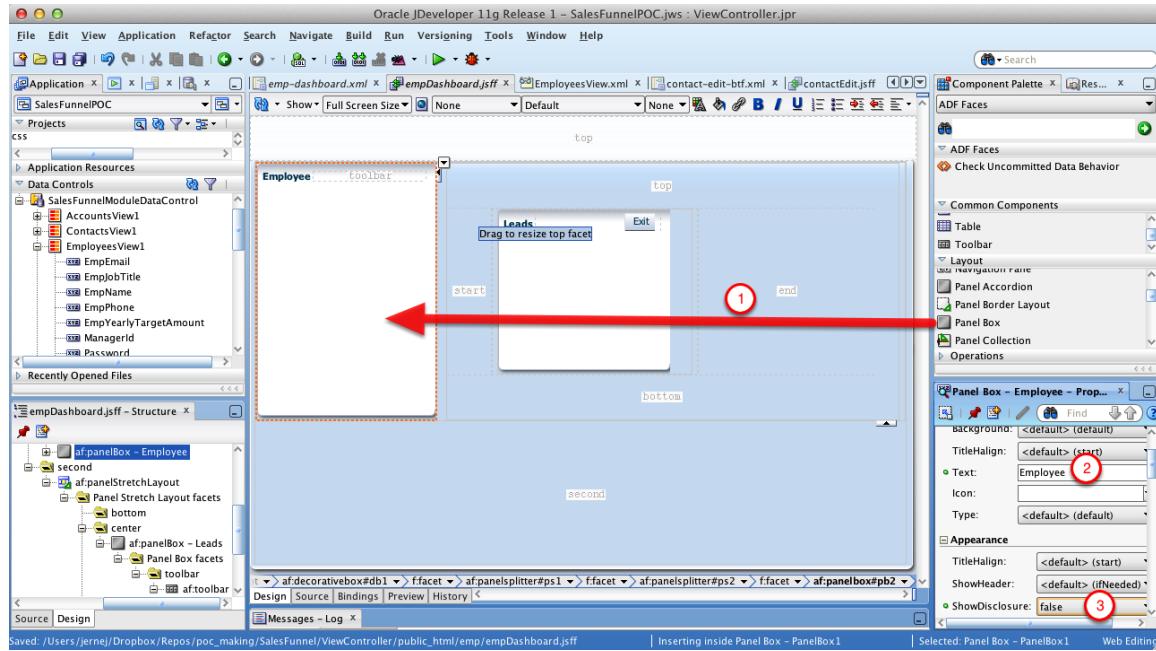
1. Right-click the toolbar
2. Insert inside > Toolbar Button

Set Button properties



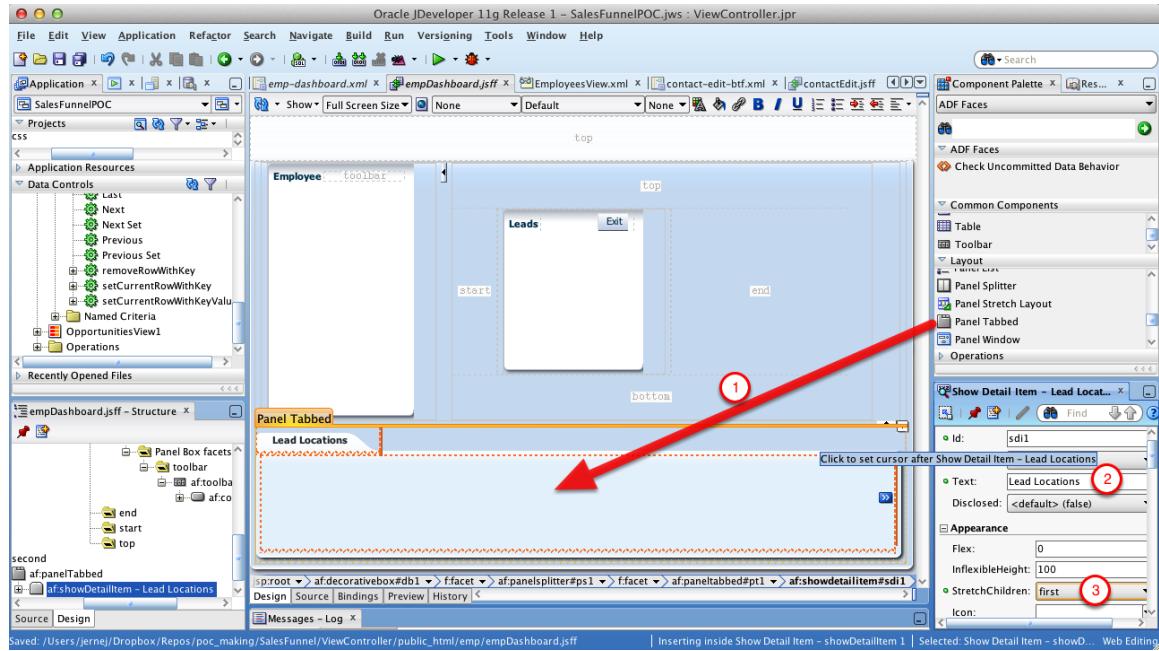
1. Set Text to Exit
2. Set Action to exit

Add Panel Box



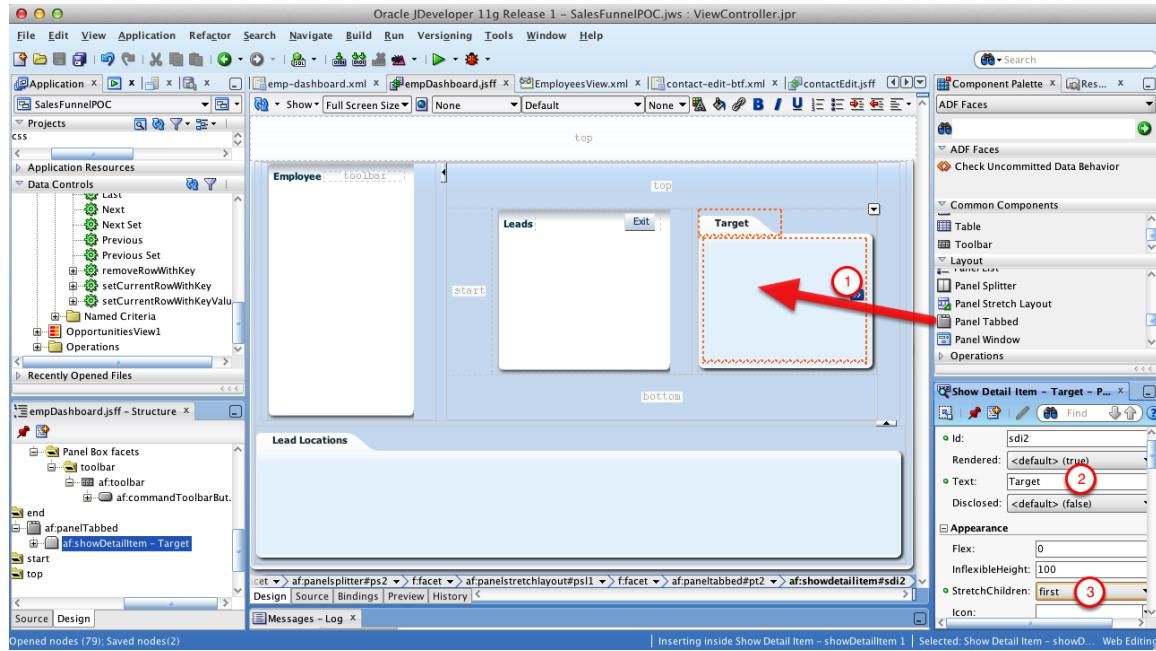
1. Drag and drop Panel Box to the first facet of the second (top) panel splitter
2. Set Text to Employee
3. Set ShowDisclosure to false

Add Panel Tabbed



1. Drag and drop Panel Tabbed to the second facet of the first splitter
2. Set showDetailItem Text to Lead Locations
3. Set StretchChildren to first

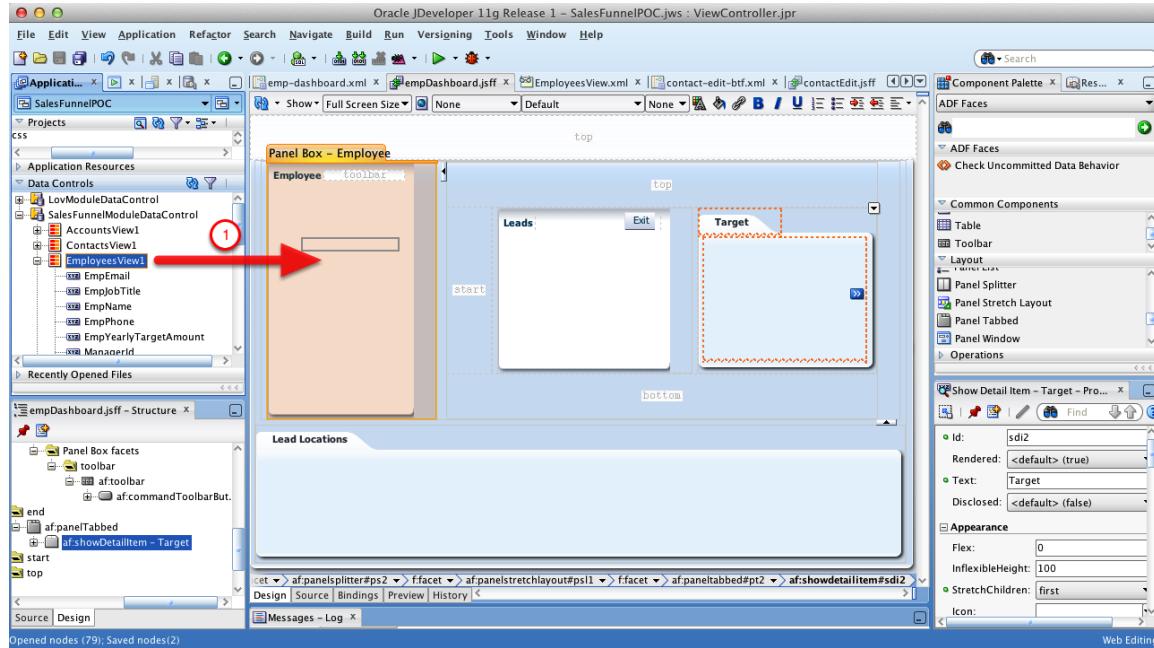
Add another Panel Tabbed



1. Drag and drop Panel Tabbed to the end facet of Panel Stretch Layout
2. Set Text of showDetailItem to Target
3. Set StretchChildren to first

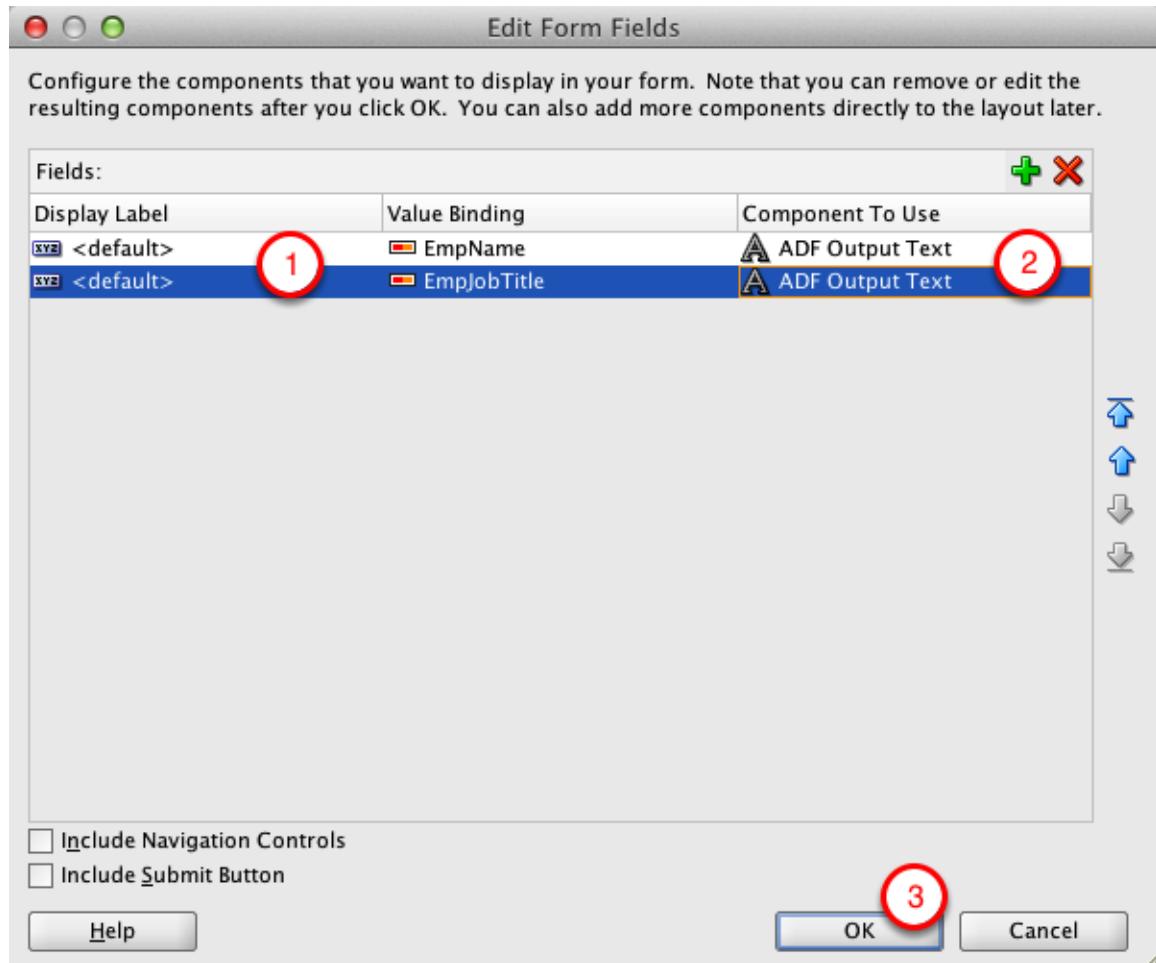
35 Add data to Dashboard

Add Employee data to dashboard



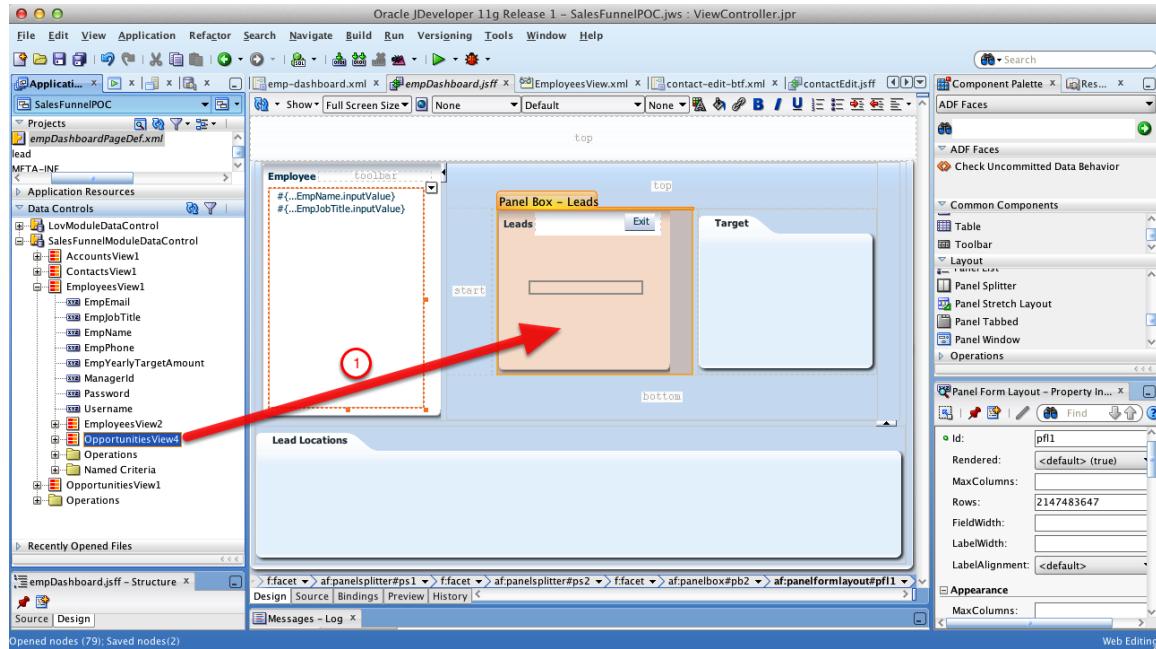
1. Drag and drop EmployeesView1 from Data Controls on Panel Box - Employee
2. Choose Form > ADF Read Only Form from the menu

Edit Form Fields



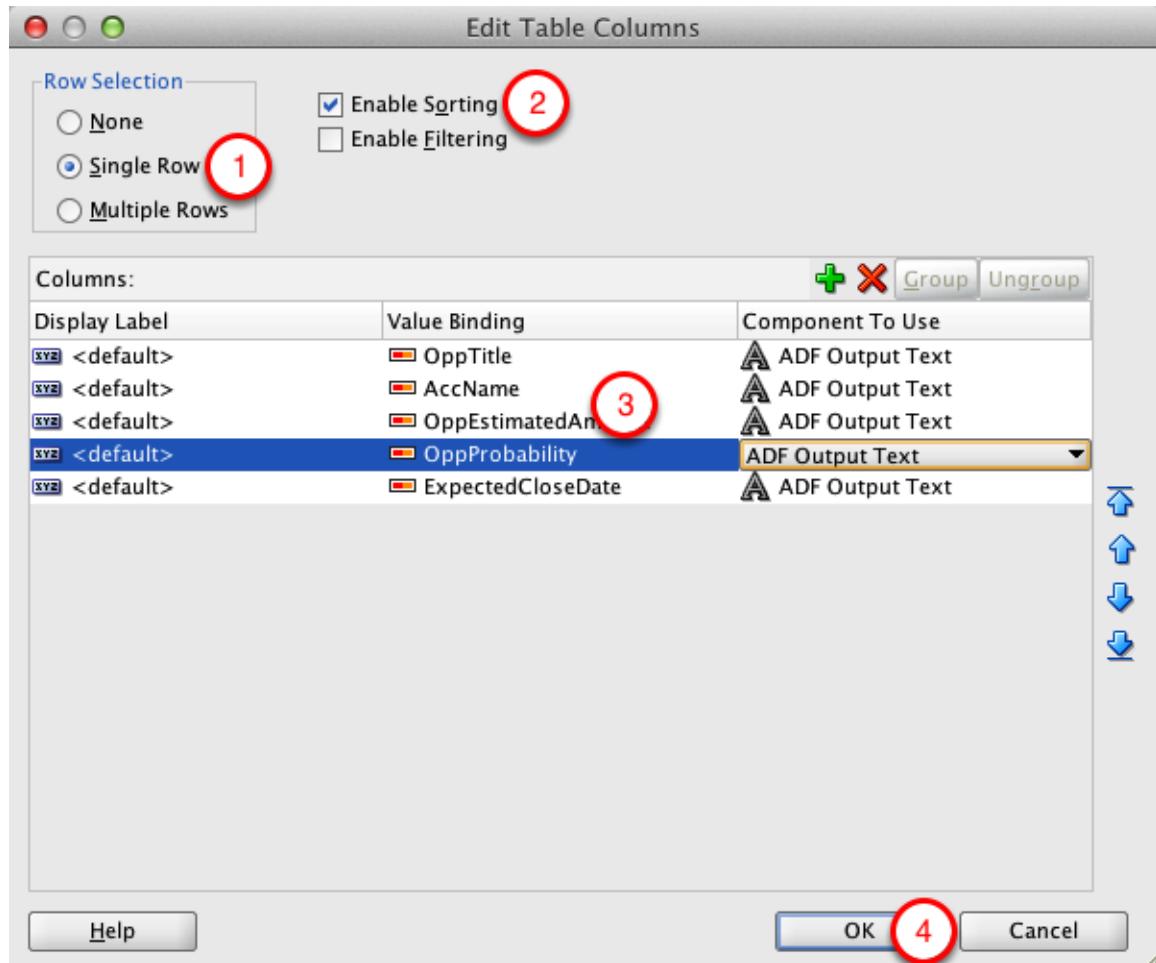
1. Delete all attributes except EmpName and EmpJobTitle
2. Change Component To Use to ADF Output Text
3. Click OK

Add Opportunities Table



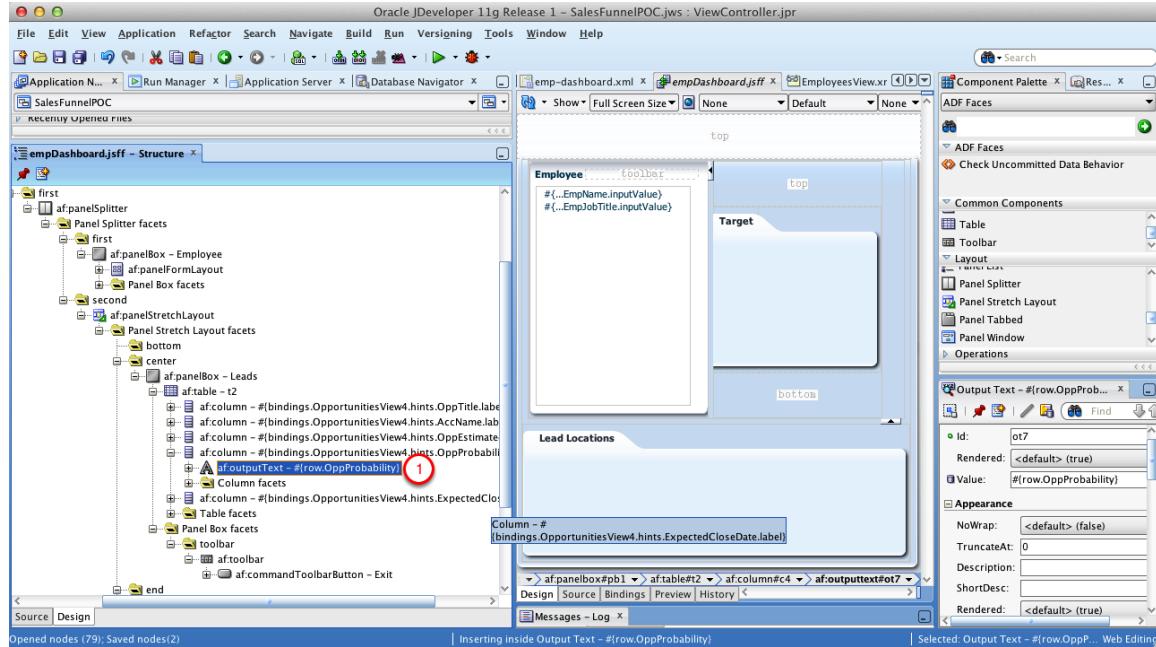
1. Drag and drop OpportunitiesView4 nested under EmployeesView1 to the Panel Box - Leads
2. Select Table > ADF Read Only Table from the context menu

Edit Table Columns



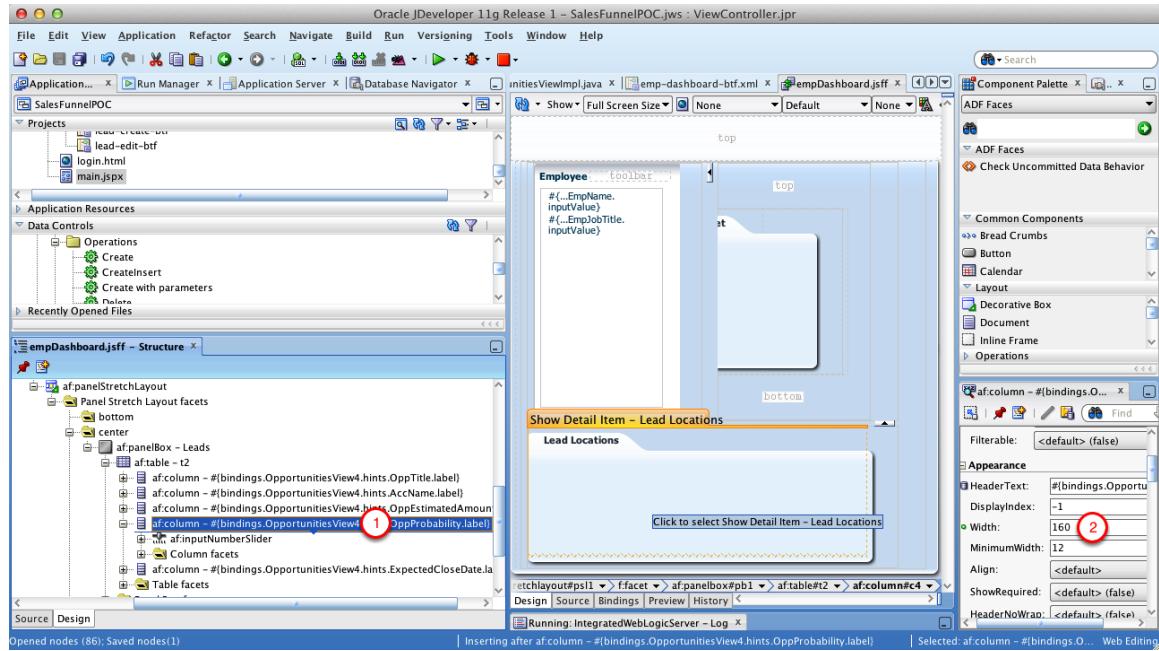
1. Select Single Row Selection
2. Select Enable Sorting
3. Rearrange the attributes to have the same order and attributes as the screenshot (use delete button, up / down arrows, or change attributes by clicking on them)
4. Click OK

Convert outputText to Number Slider



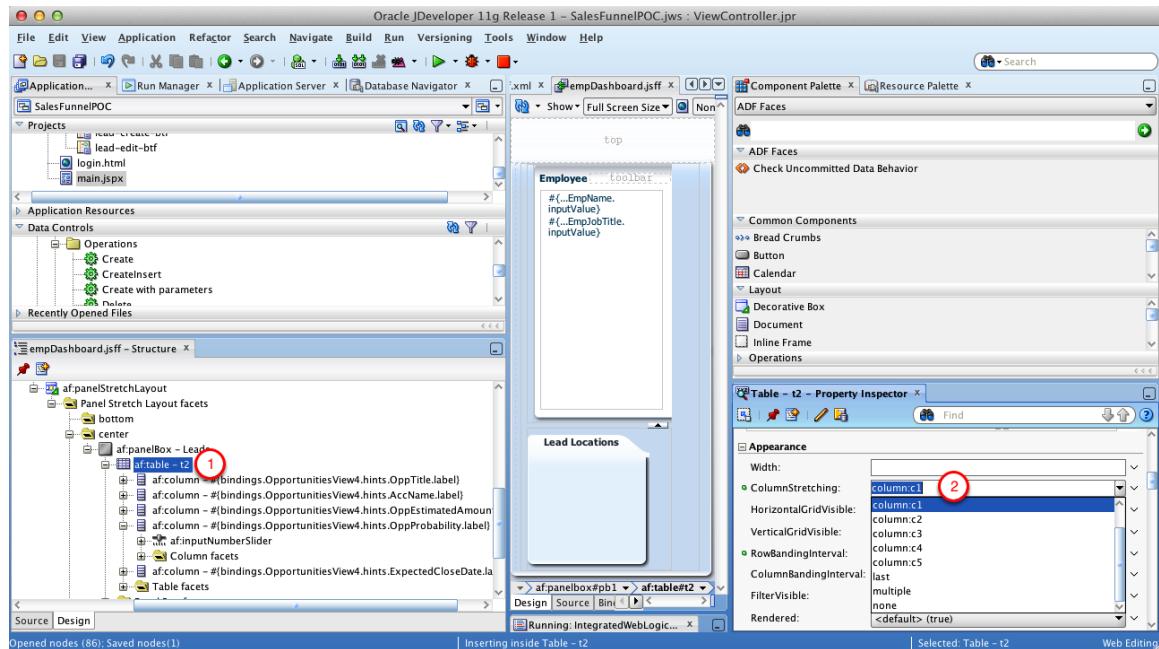
1. In the structure window, find the generated table, expand it and find OppProbability Output Text
2. Right click it and select "Convert to" from the menu
3. Select Input Number Slider

Set Column Properties



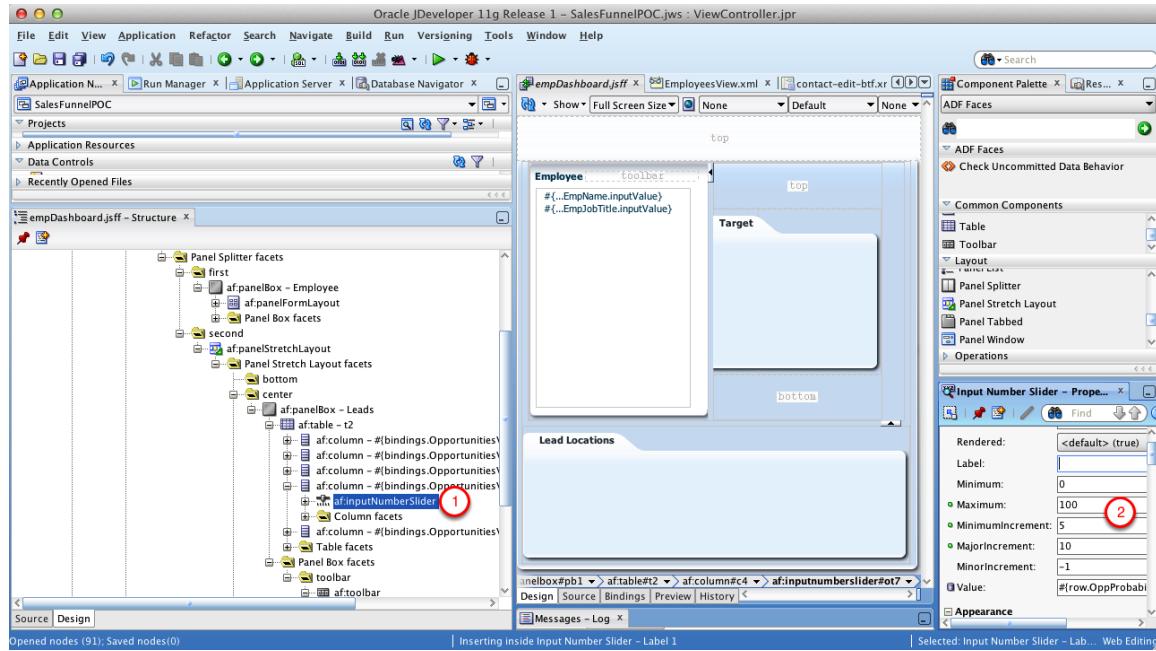
1. Select the column above the slider component
2. Change width Property to 160

Set Table Properties



1. Select leads table
2. Set ColumnStretching property to column:c1

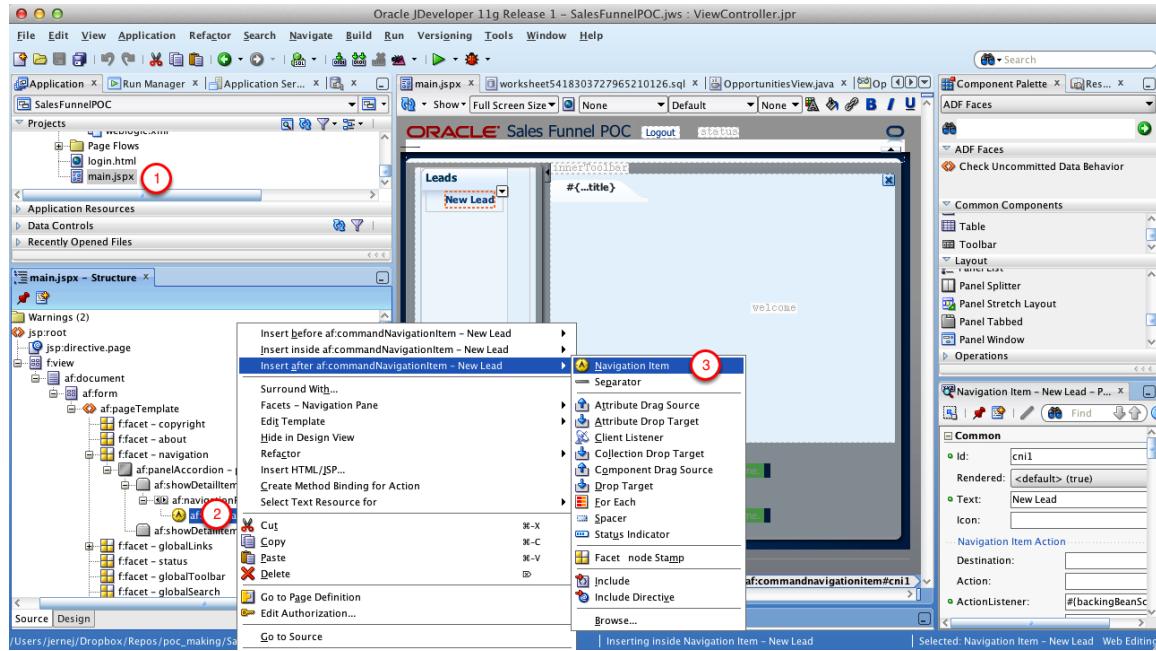
Set Number Slider properties



1. Select the slider
2. Set properties:

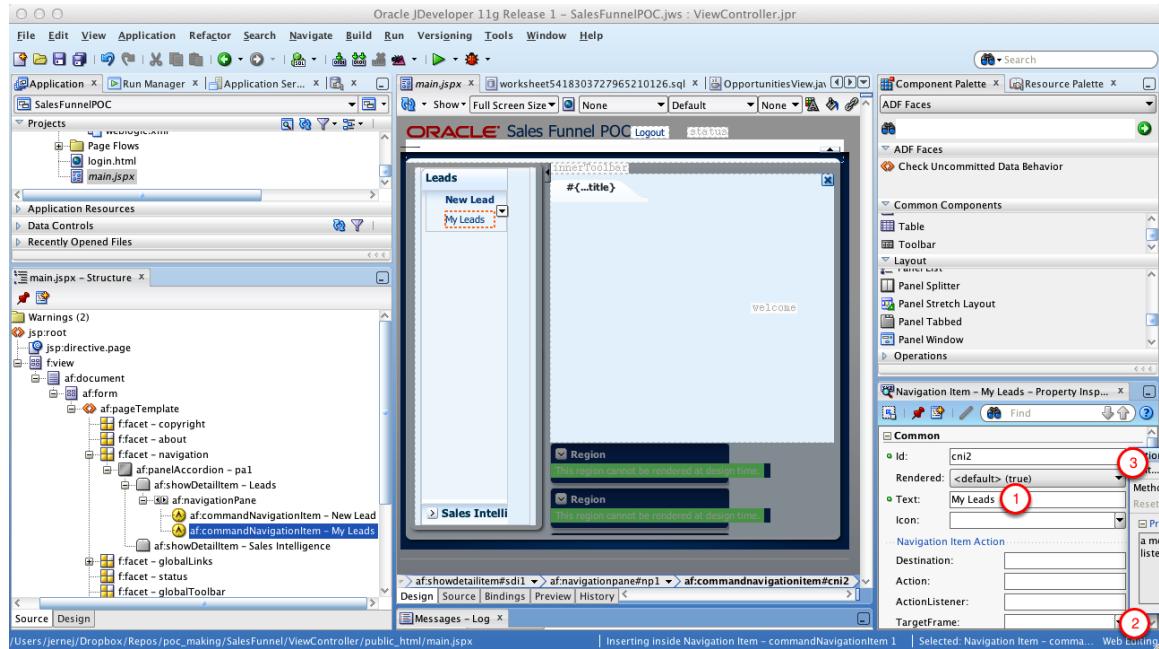
- Label to empty string
- Maximum: 100
- MinimumIncrement:5
- MajorIncrement:10

Add Navigation Item to the main form



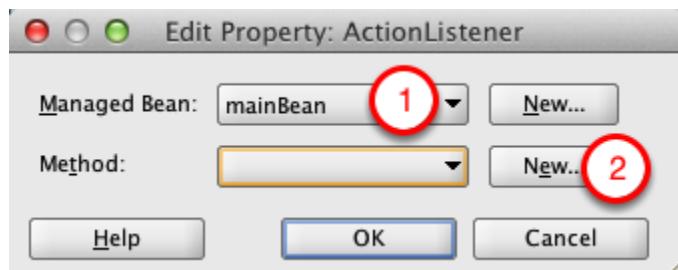
1. Open main.jspx
2. Find the navigation item in the Structure Window and right-click it
3. Insert after > Navigation Item

Set Navigation item properties



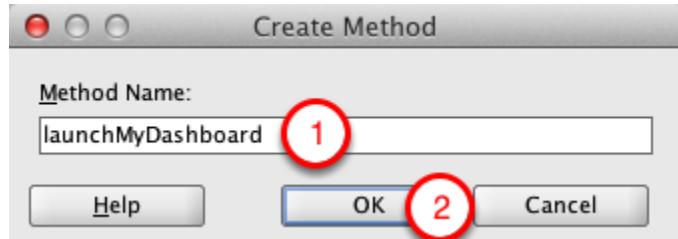
1. Change Text to "My Leads"
2. Click the down arrow next to ActionListener
3. Select Edit from the menu

Edit Property: ActionListener



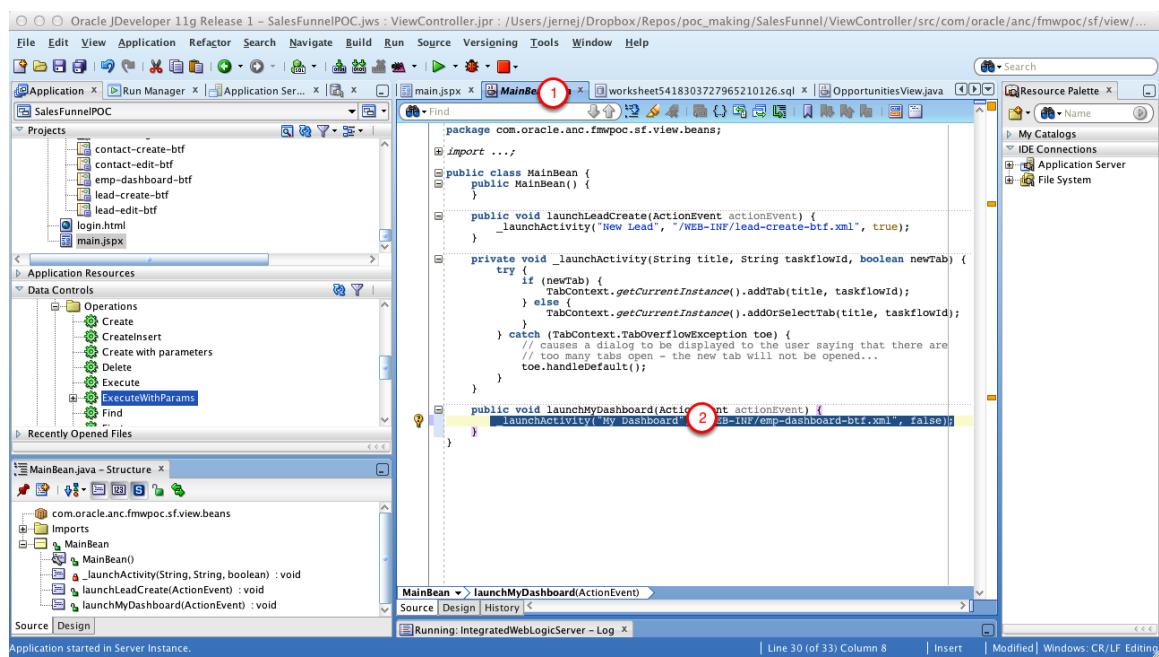
1. Select mainBean
2. Click New next to Method

Create Method



1. Name it launchMyDashboard
2. Click OK

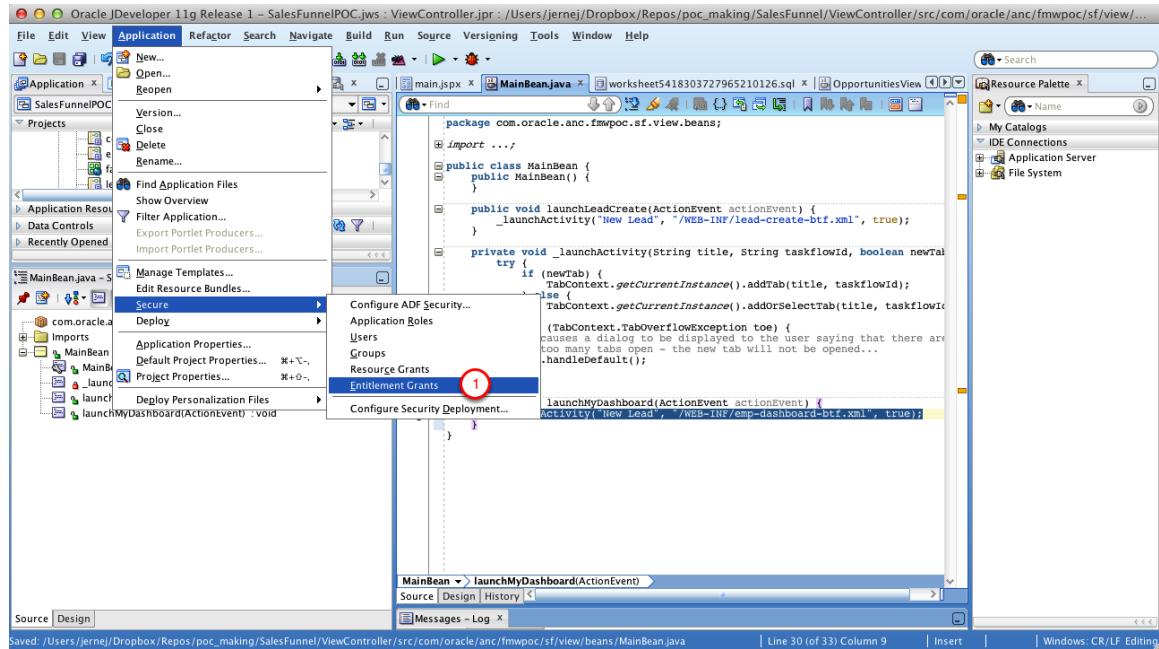
Implement launchMyDashboard



1. Open MainBean.java
2. Paste the code inside launchMyDashboard method:

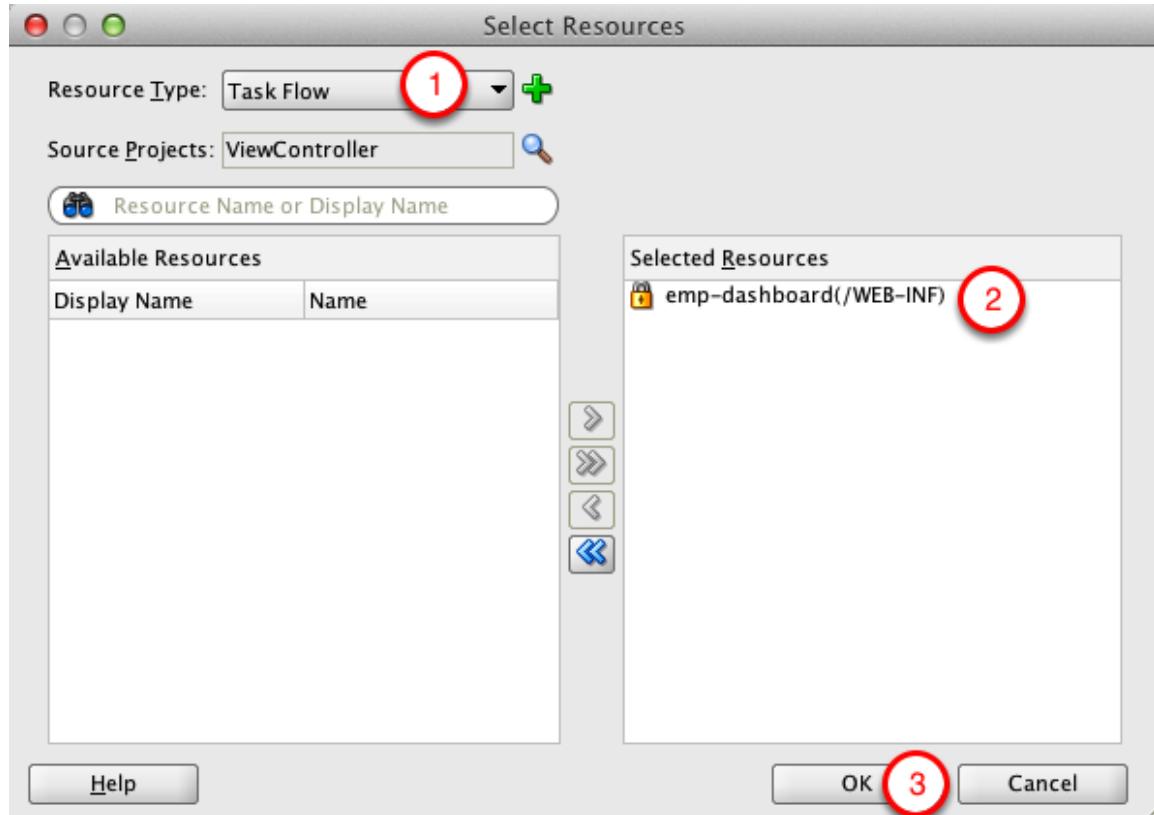
```
_launchActivity("My Dashboard", "/WEB-INF/emp-dashboard-btf.xml", false);
```

Configure Security Grants



1. Select Application > Secure > Entitlement Grants from the menu

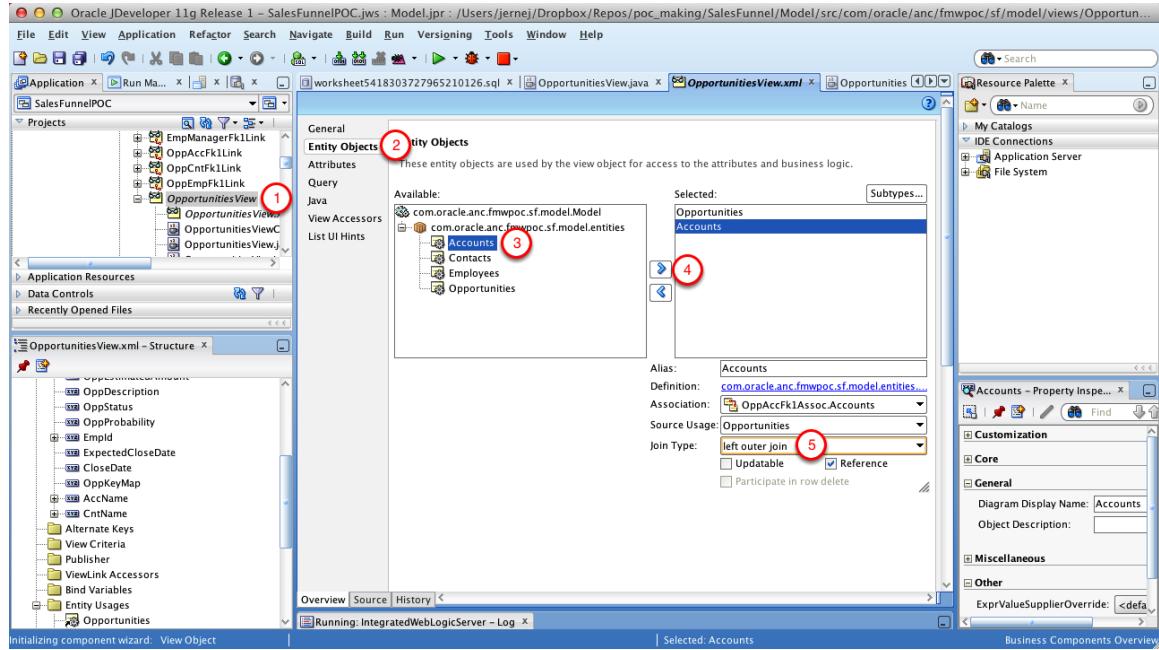
Select Resources



1. Change Resource Type to Task Flow
2. Slide emp-dashboard to the right
3. Click OK

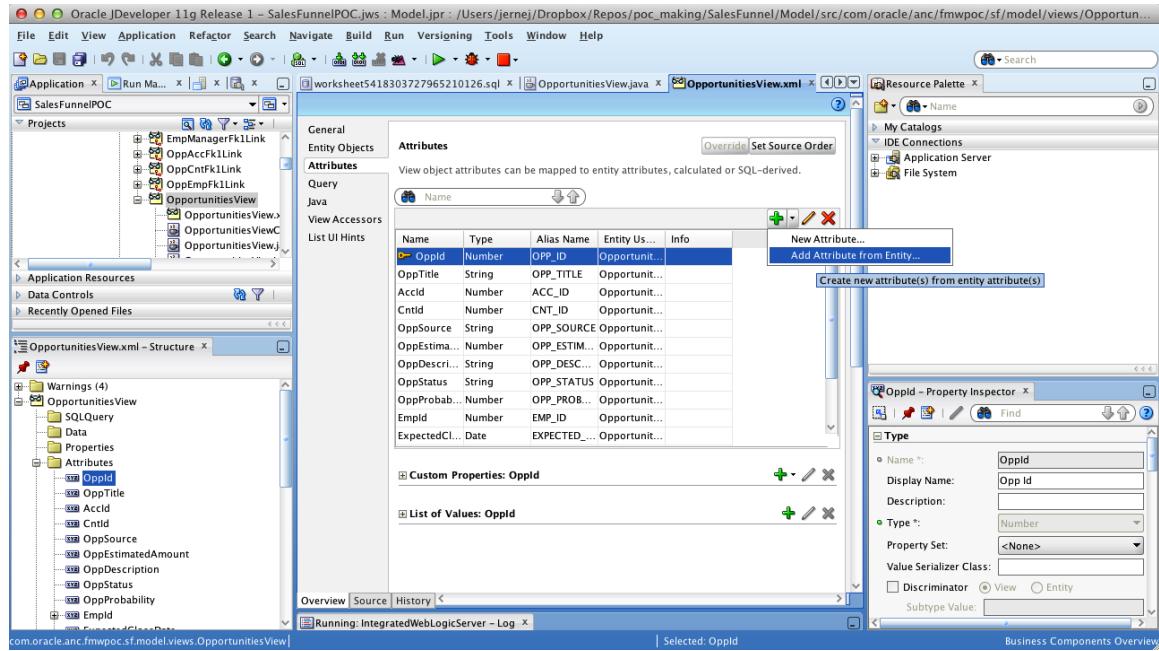
36. Geographic map

Add Accounts Entity to OpportunitiesView



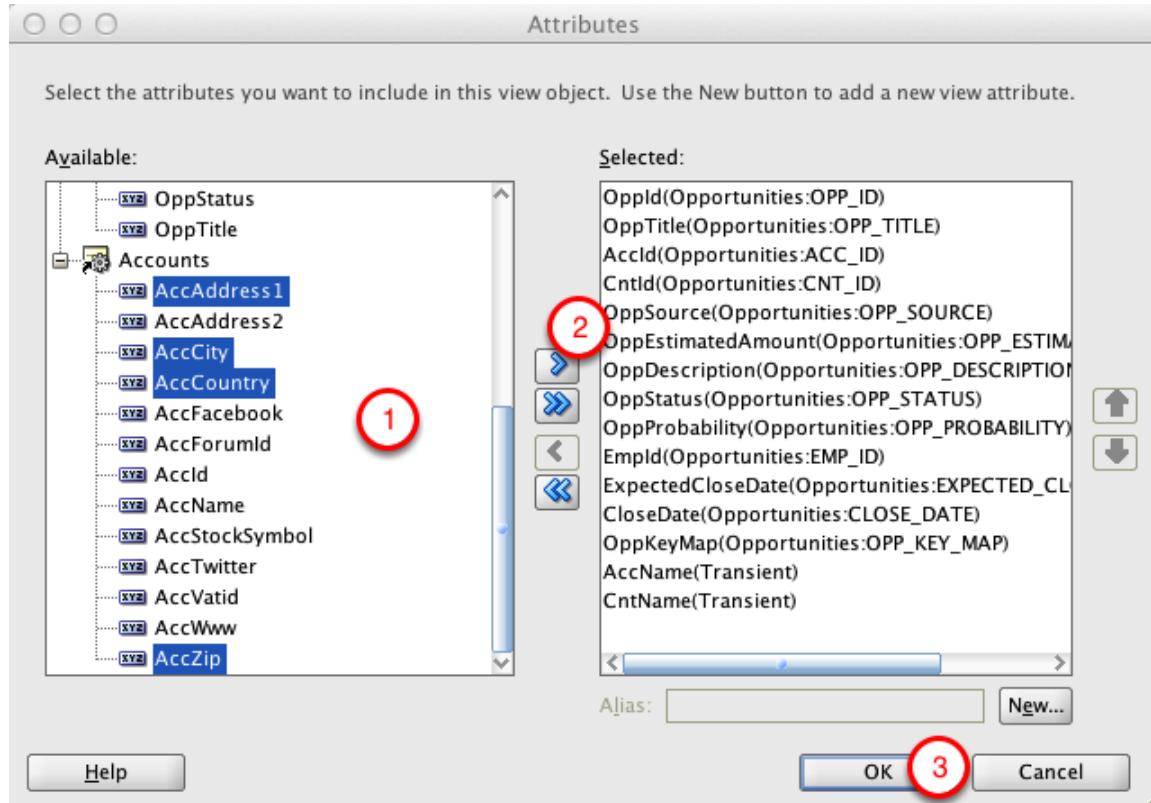
1. Open OpportunitiesView
2. Open Entity Objects tab
3. Select Accounts entity
4. Slide it to the right
5. Set Join Type to Left Outer Join

Add Attributes to OpportunitiesView



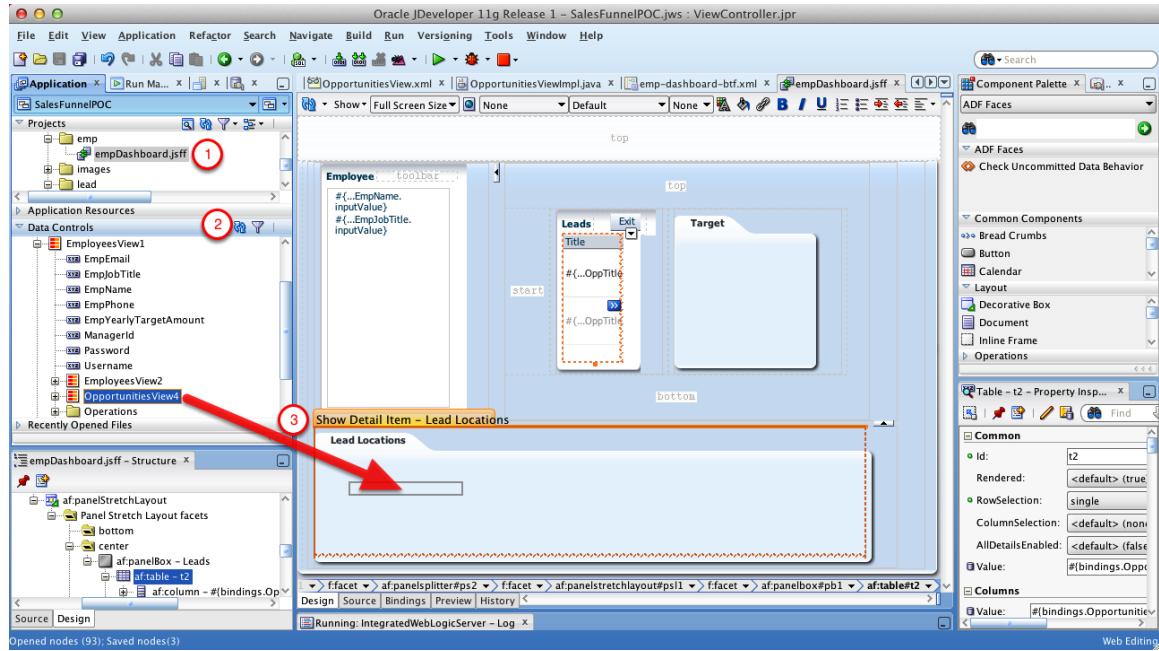
1. Open Attributes tab
2. Click the down arrow next to plus and select "Add Attribute from Entity"

Attributes



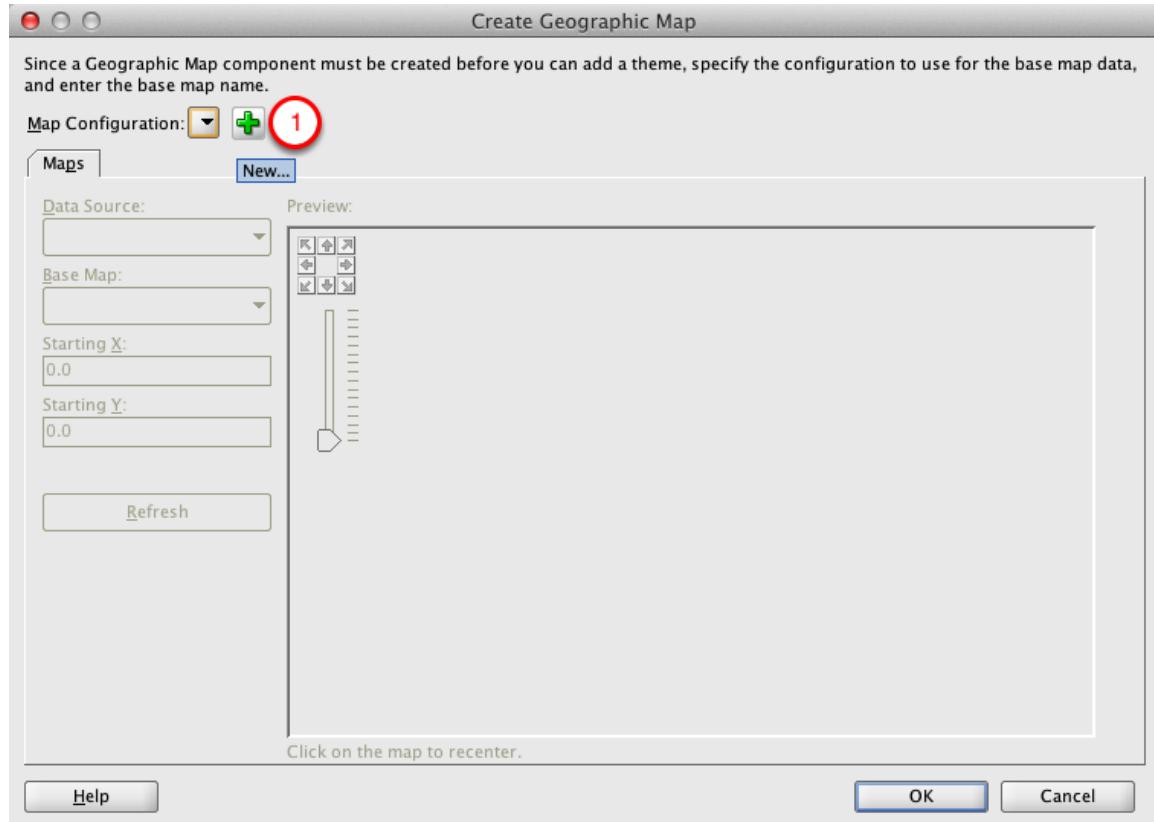
1. Select AccAddress1, AccCity, AccCountry, AccZip
2. Slide them to the right (note: AccId will automatically be added)
3. Click OK

Drag and drop OpportunitiesView4 on Panel Tabbed



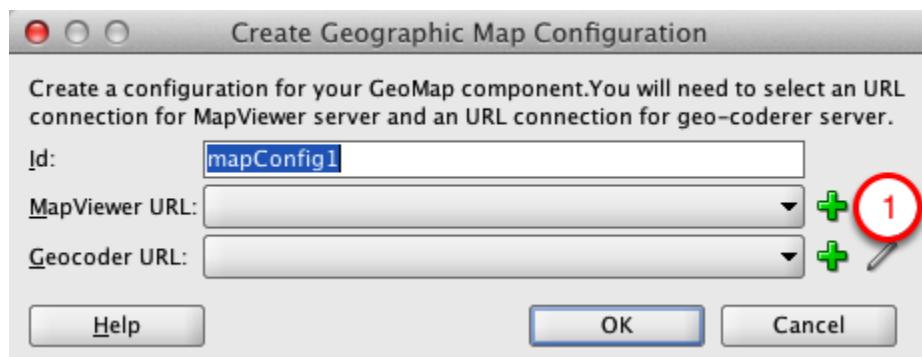
1. Open empDashboard.jsff
2. Click refresh button in the Data Controls
3. Expand EmployeesView1 and drag and drop the nested OpportunitiesView4 to the bottom panel tabbed
4. Select Geographic map > Map And Point Theme from the context menu

Create Geographic Map



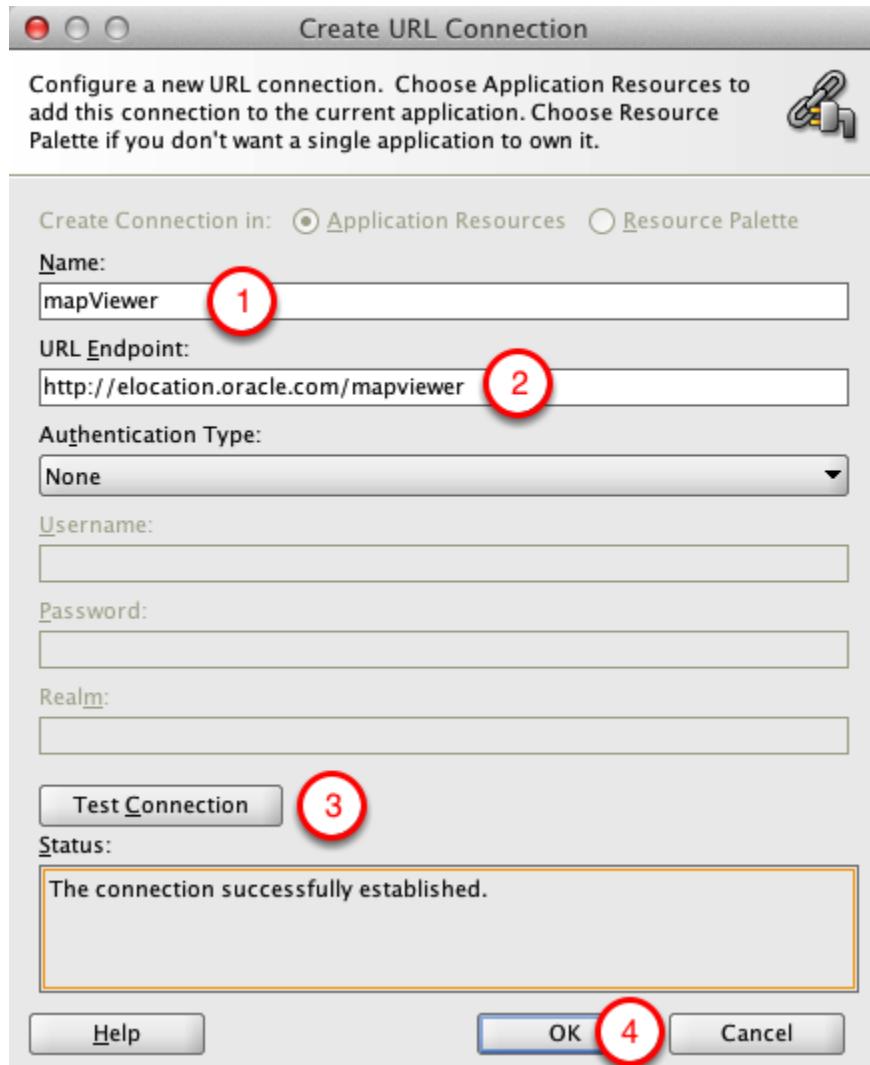
1. Click plus next to map configuration

Create Geographic Map Configuration



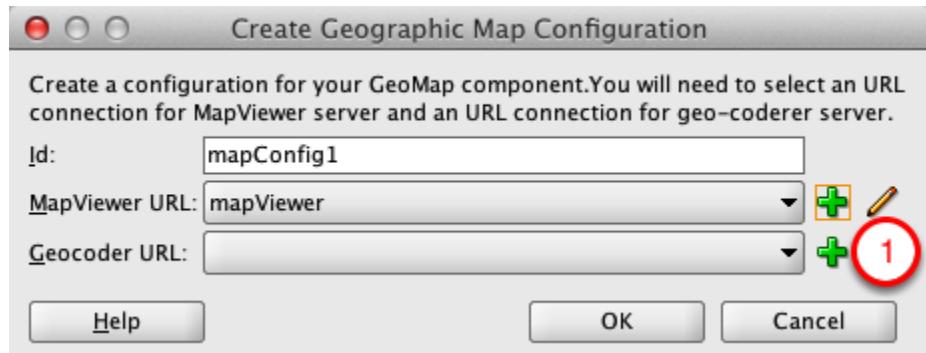
1. Click plus next to MapViewer URL

Create URL Connection



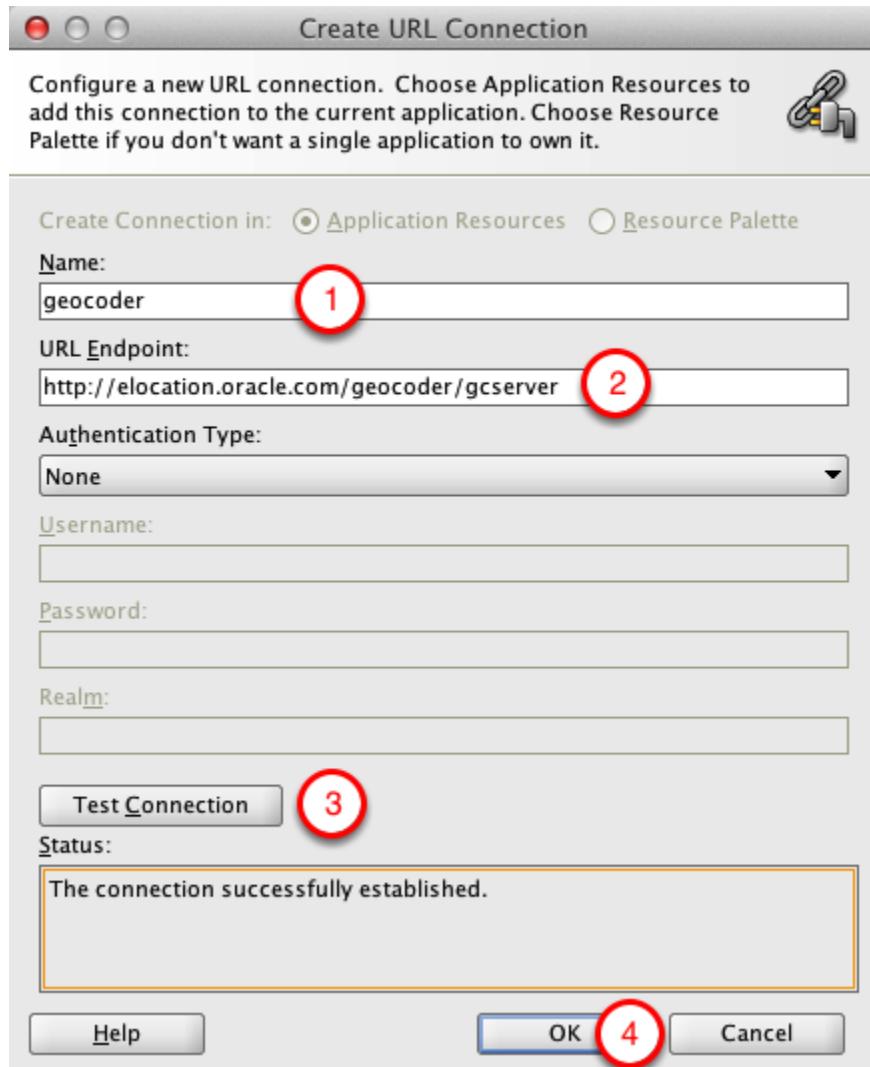
1. Set name to mapViewer
2. Set endpoint URL to <http://elocation.oracle.com/mapviewer>
3. Test connection and make sure it is successful
4. Click OK

Create Geographic Map Configuration



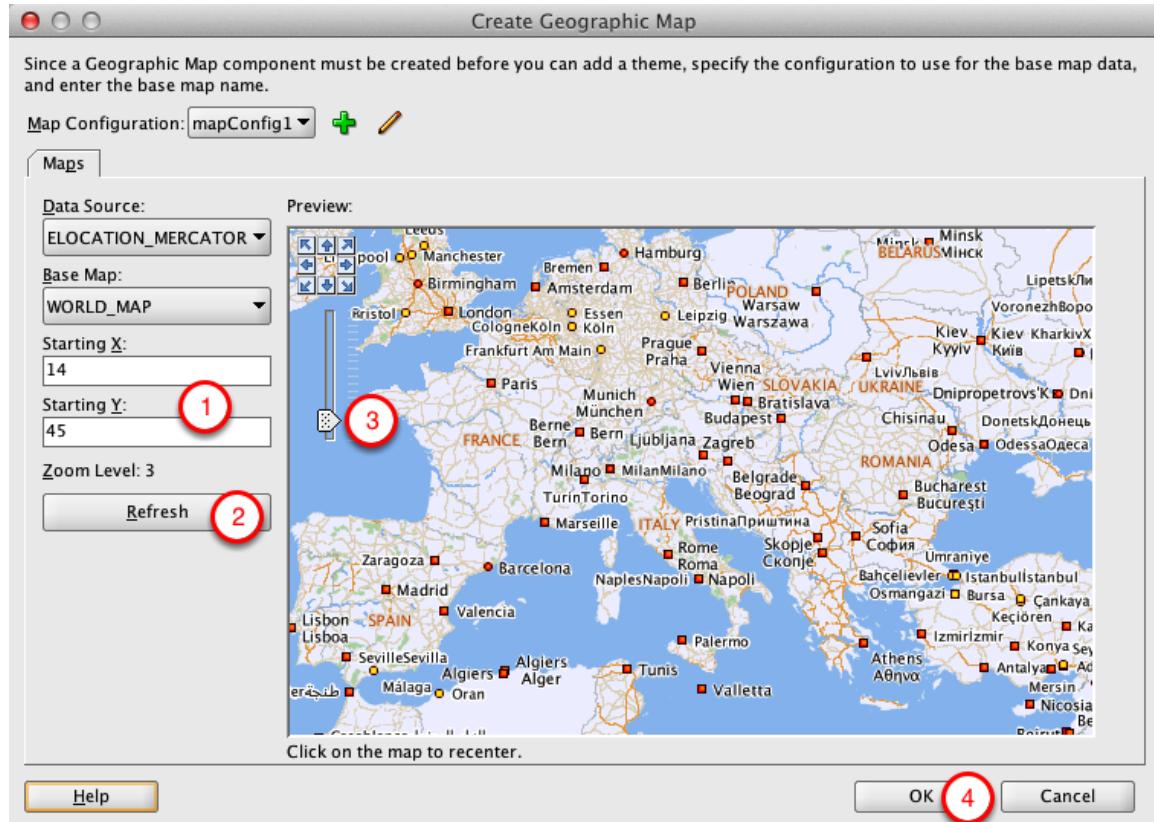
1. Click plus next to Geocoder URL

Create URL Connection



1. Set name to geocoder
2. Set endpoint URL to http://elocation.oracle.com/geocoder/gcserver
3. Test connection and make sure it is successful
4. Click OK

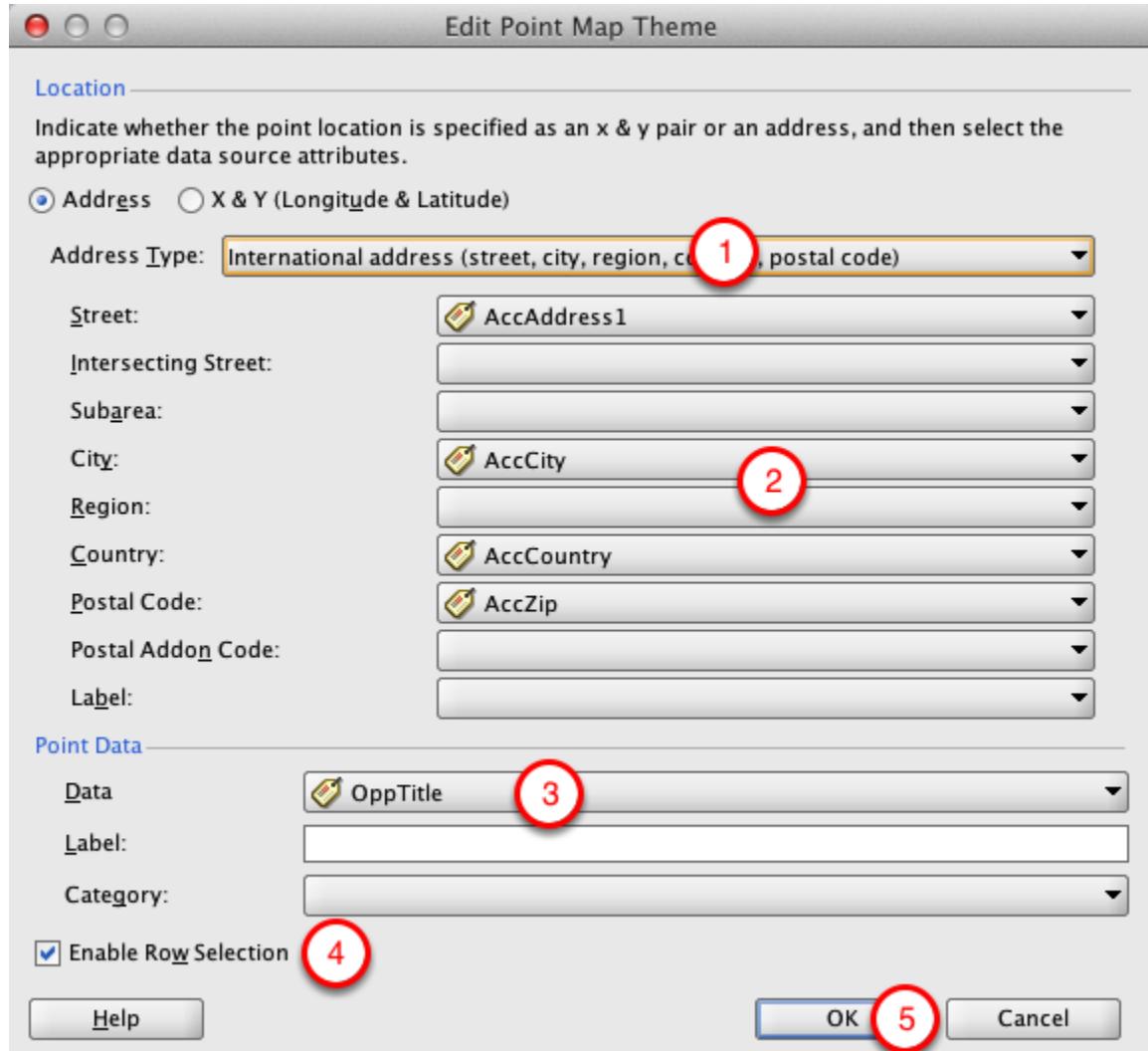
Create Geographic Map



1. Set x to 14, y to 45
2. Click Refresh
3. Use slider to set zoom level to 3
4. Click OK

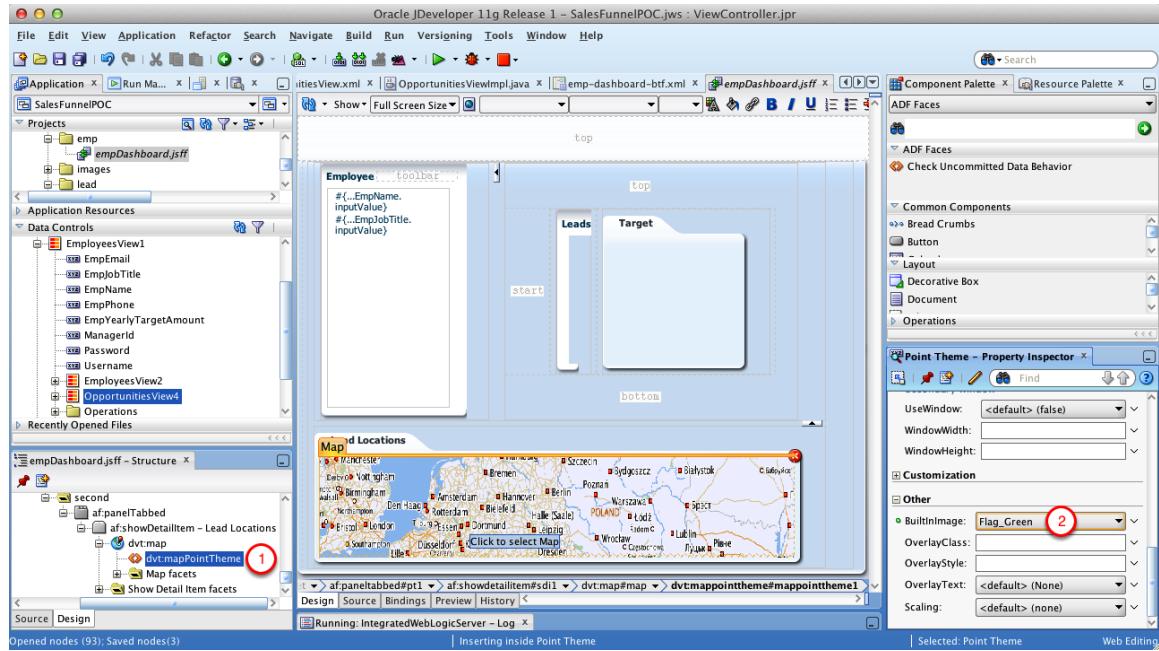
Note: you can select any starting position you want, but the default data is provided for Eastern Europe

Create Point Map Theme



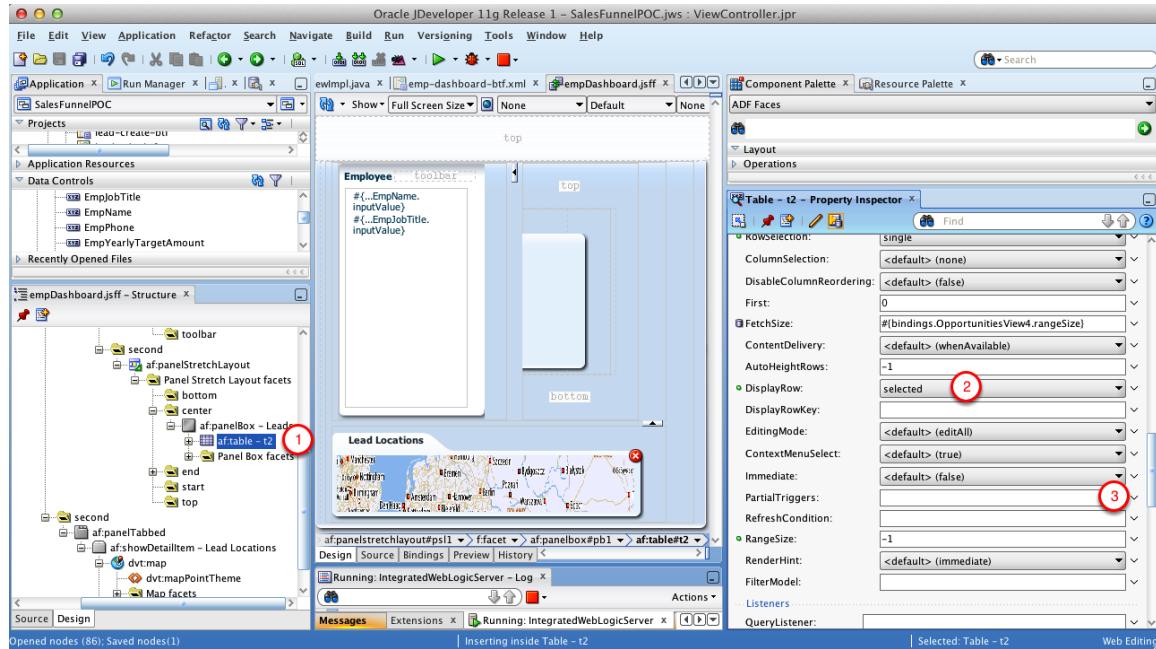
1. Change address type to International Address
2. Map the address to data attributes
3. Set OppTitle as Data
4. Enable row selection checkbox
5. Click OK

Change Map Image



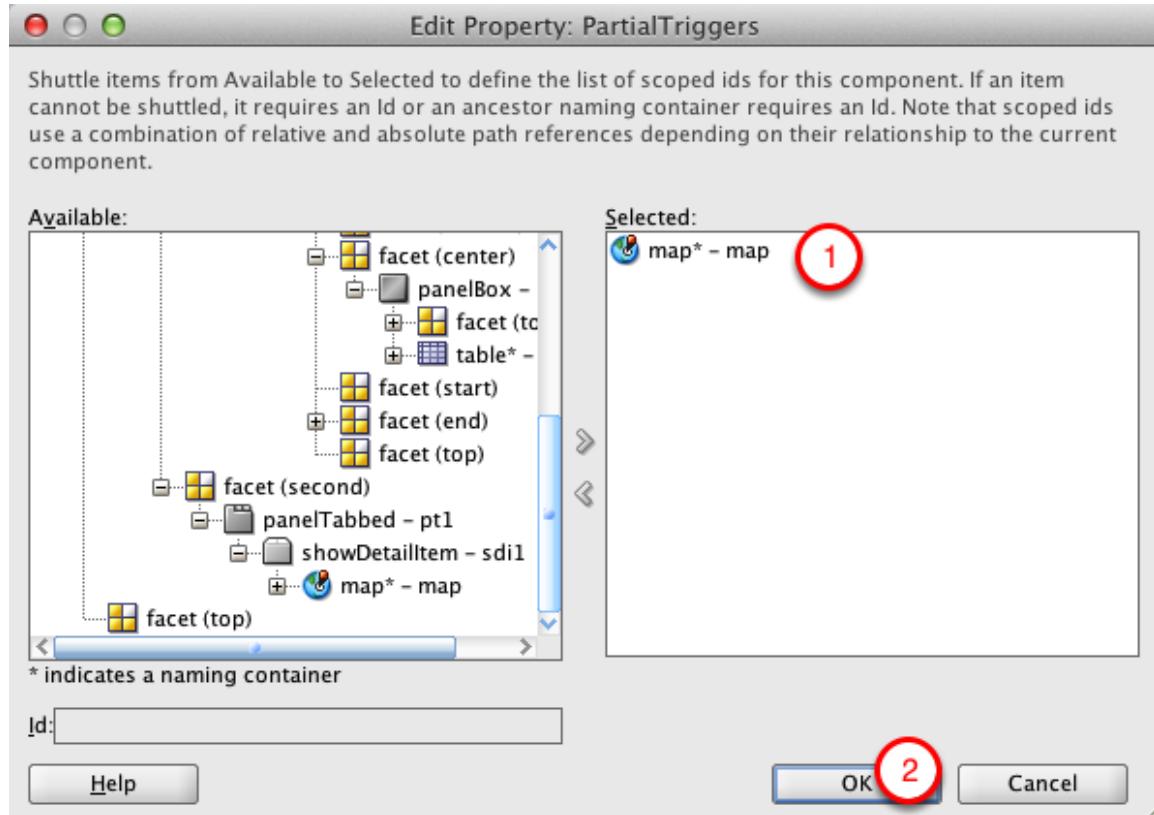
1. In the Structure window, expand map and select mapPointTheme
2. Change BuiltinImage to Flag_Green

Synchronize map selection with table



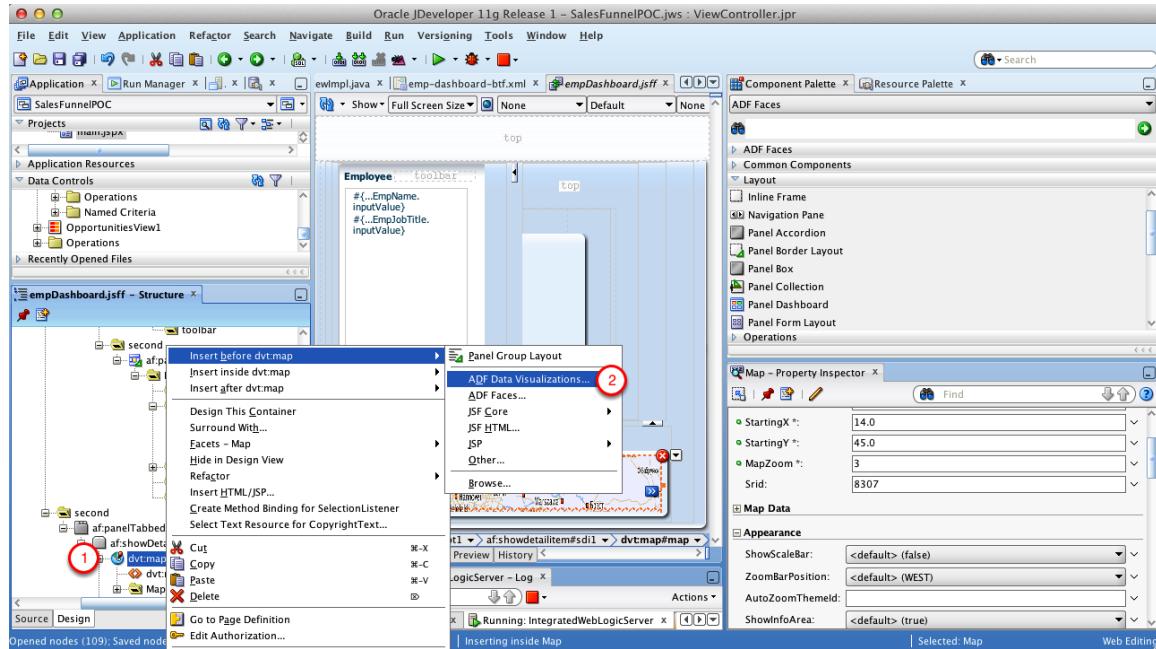
1. In the structure window, select the leads table
2. Change DisplayRow property to selected
3. Click the down arrow next to PartialTrigger edit box and select Edit from the menu

Edit Property: PartialTriggers



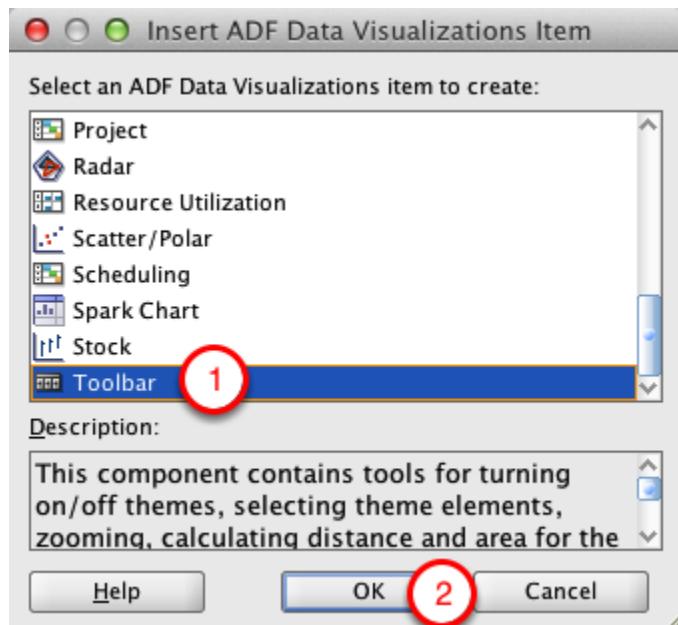
1. Find map in the structure and slide it to the right
2. Click OK

Add Map Toolbar



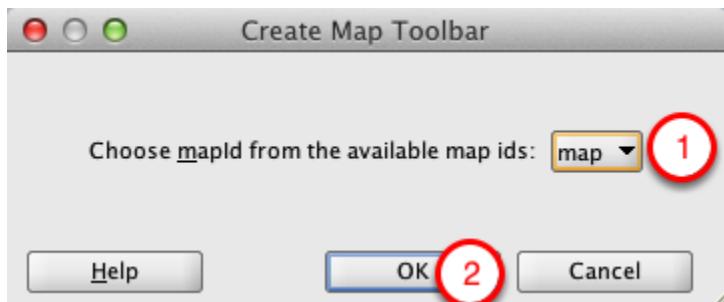
1. Right-click the map in the structure window
2. Select Insert before dvt:map > ADF Data Visualizations

Insert ADF Data Visualizations Item



1. Select toolbar from the menu
2. Click ok

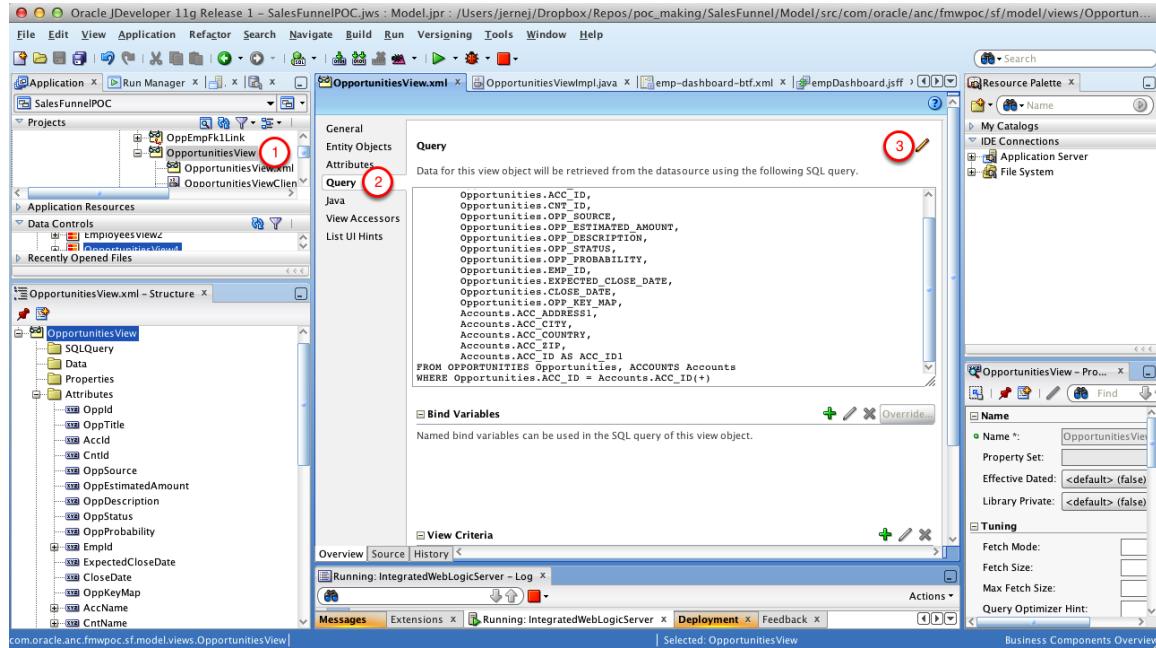
Create Map Toolbar



1. Select the map in the combo
2. Click OK

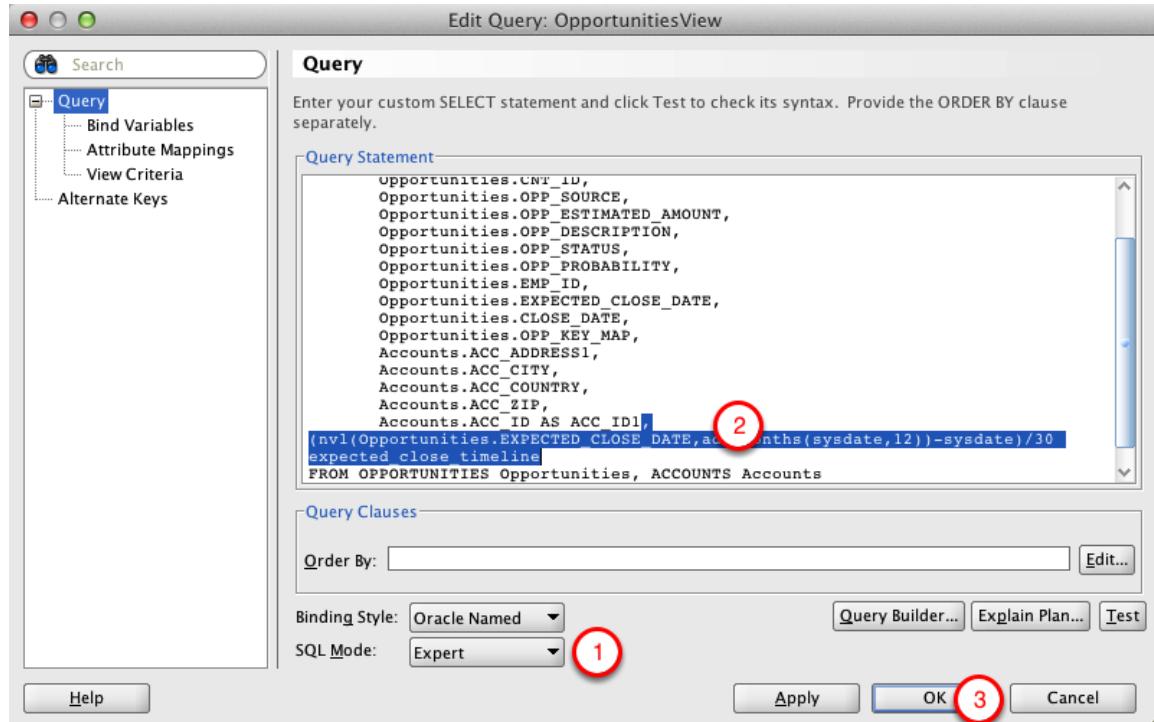
37. Bubble chart

Edit OpportunitiesView Query



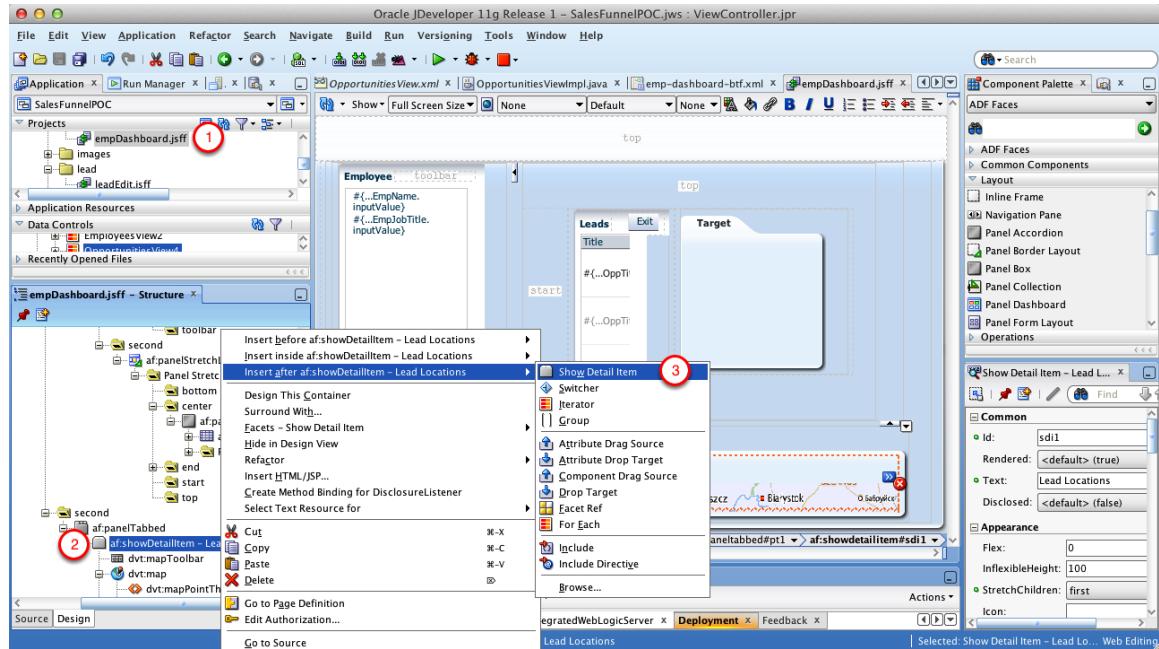
1. Open OpportunitiesView
2. Open Query tab
3. Click Edit icon

Edit Query: OpportunitiesView



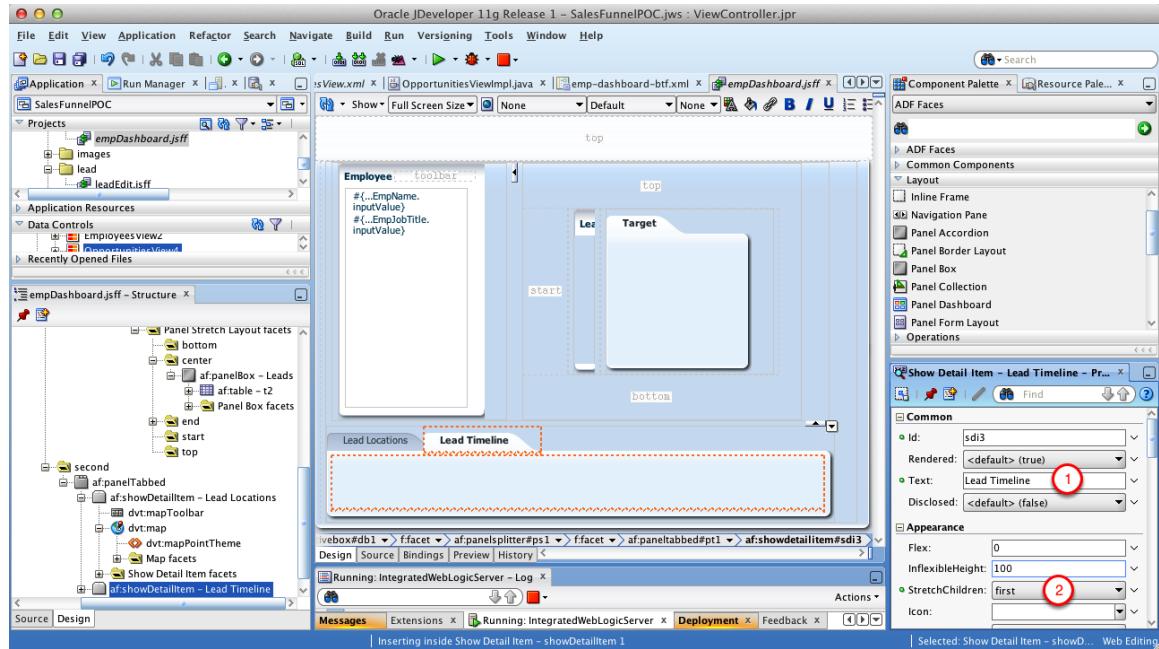
1. Change SQL Mode to expert
2. Paste sql after the last column:
",(nvl(Opportunities.EXPECTED_CLOSE_DATE,add_months(sysdate,12))-sysdate)/30
expected_close_timeline"
3. Click OK

Add tab to Panel Tabbed



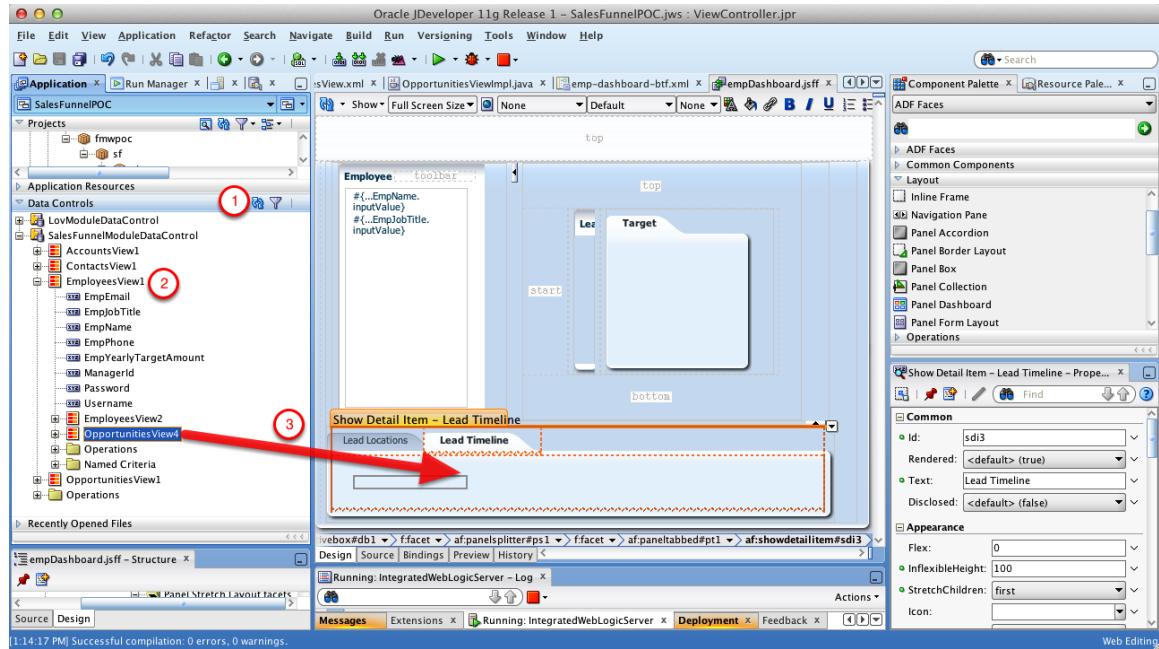
1. Open `empDashboard.jsff`
2. Right-click `showDetailItem` hosting the map component
3. Select `insert after > Show Detail Item`

Set tab properties

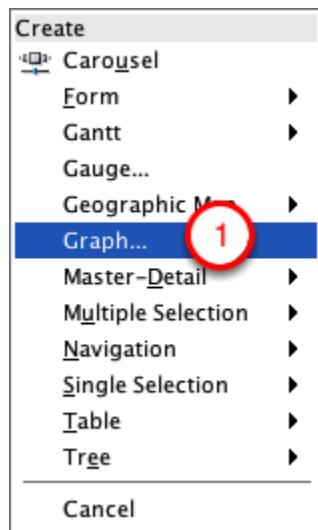


1. Set Text property to Lead Timeline
2. Set StretchChildren property to first

Add Bubble Chart

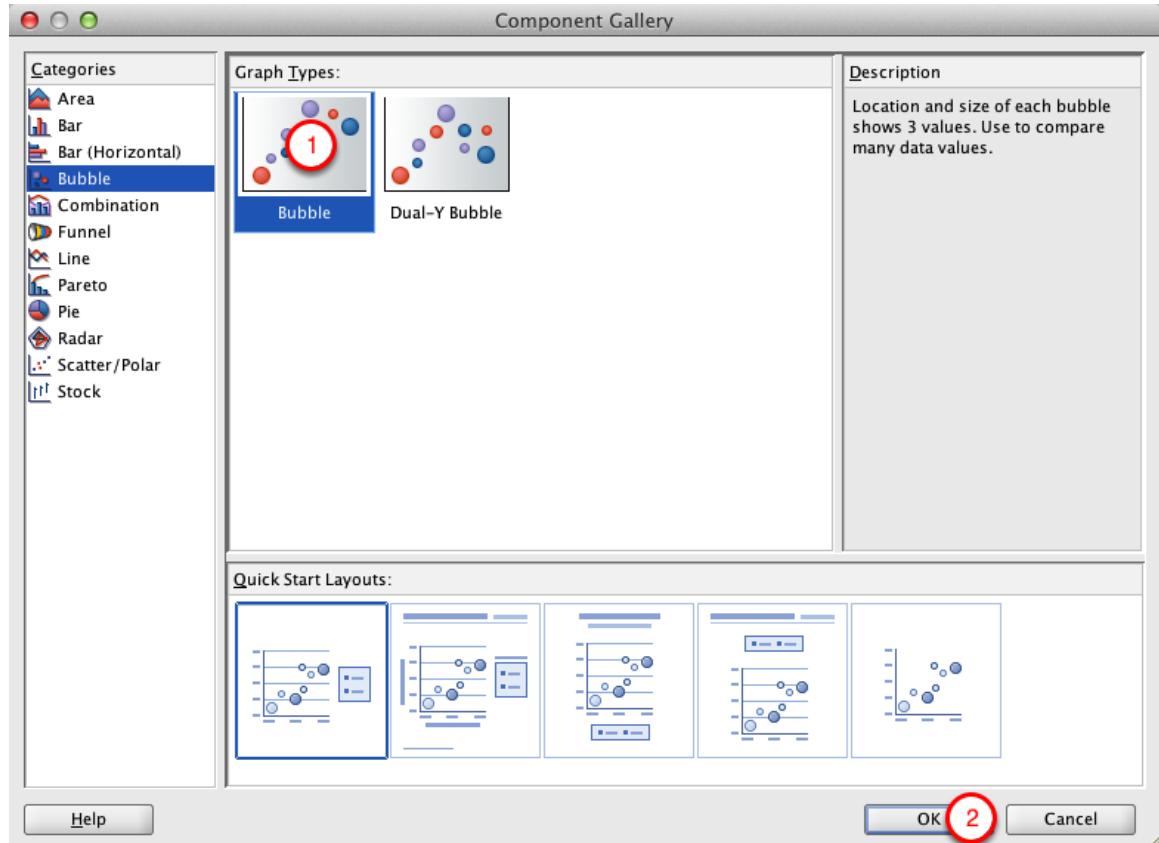


1. Click refresh icon in Data Controls
2. Expand EmployeesView1 and select the nested OpportunitiesView4
3. Drag and drop OpportunitiesView4 to the Lead Timeline tab



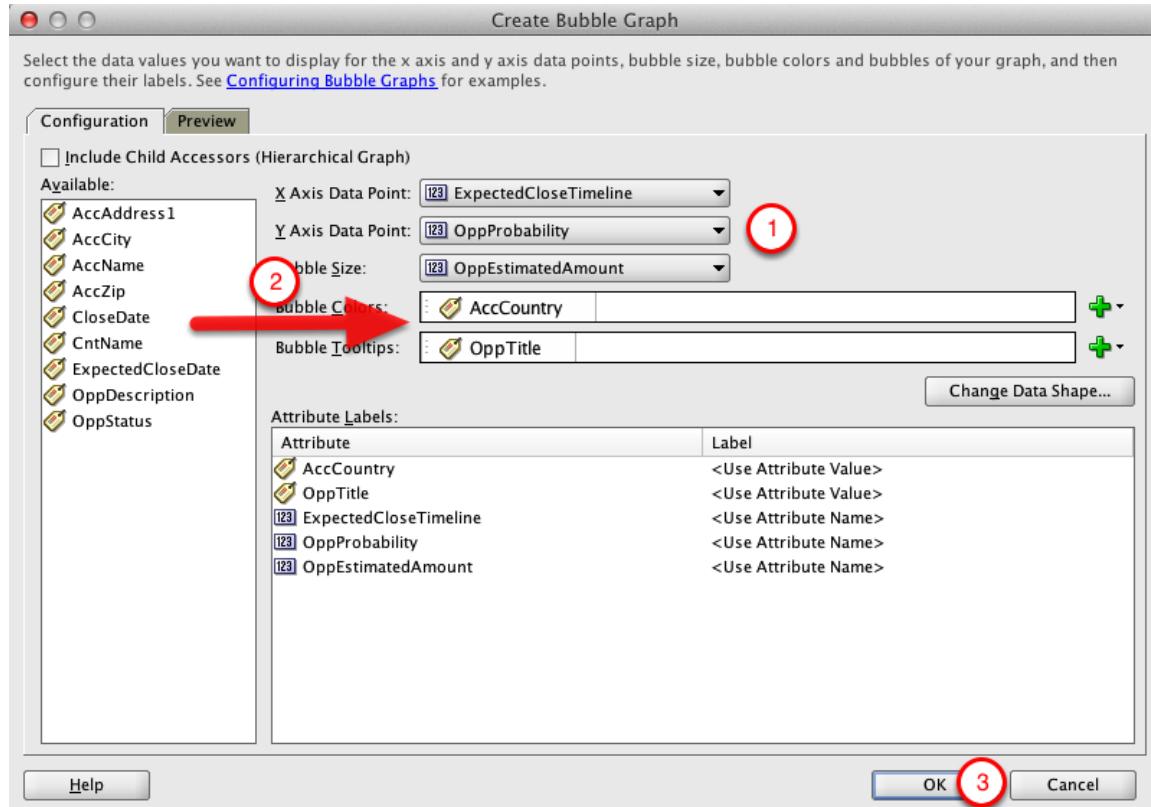
1. Select Graph from the popup menu

Component Gallery



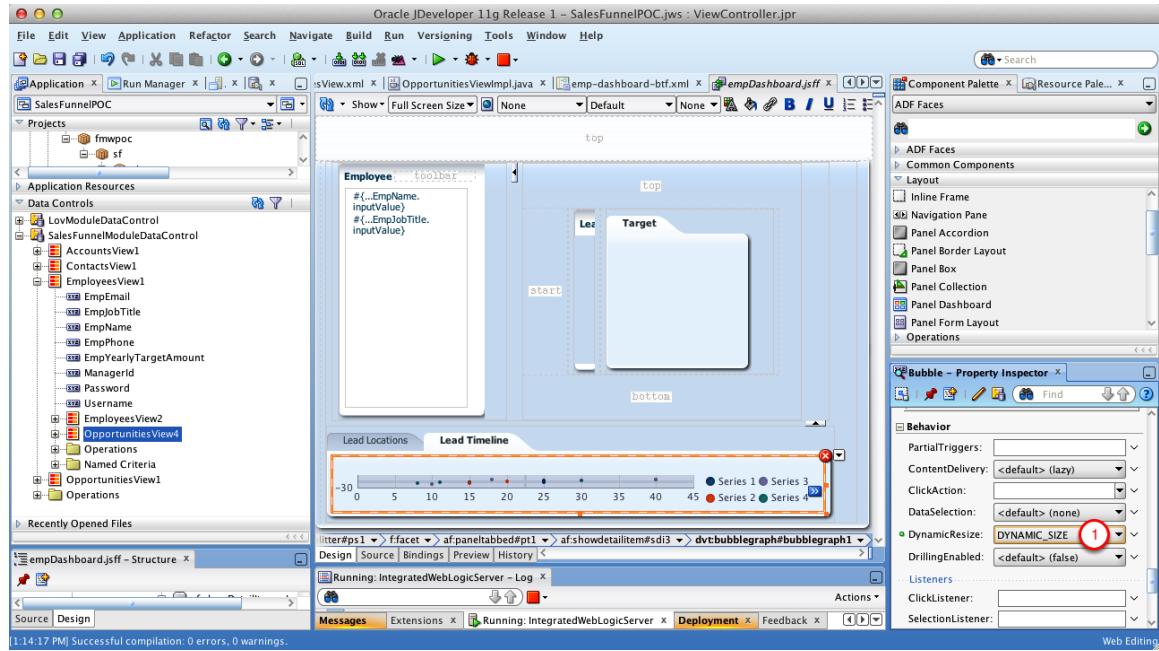
1. Select the first Bubble graph type
2. Click OK

Create Bubble Graph



1. Set properties
 - X: ExpectedCloseTimeline
 - Y: OppProbability
 - Size: OppEstimatedAmount
2. Drag and drop AccCountry to Bubble Colors and OppTitle to Bubble Tooltips
3. Click OK

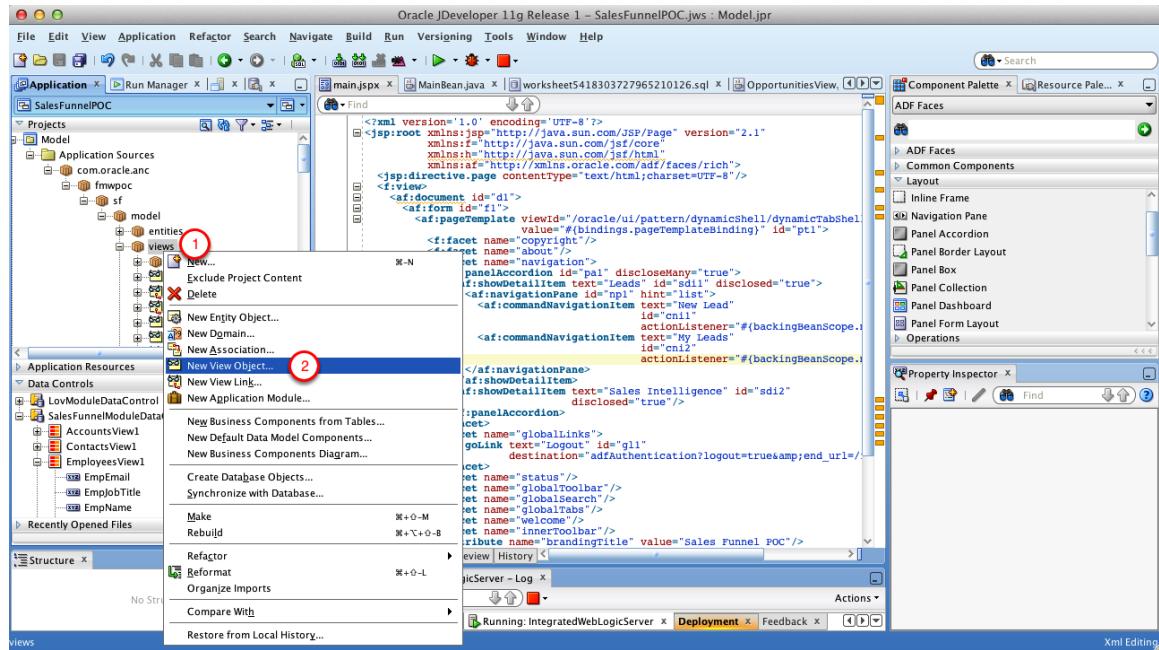
Configure Chart



1. Set DynamicResize property to DYNAMIC_SIZE

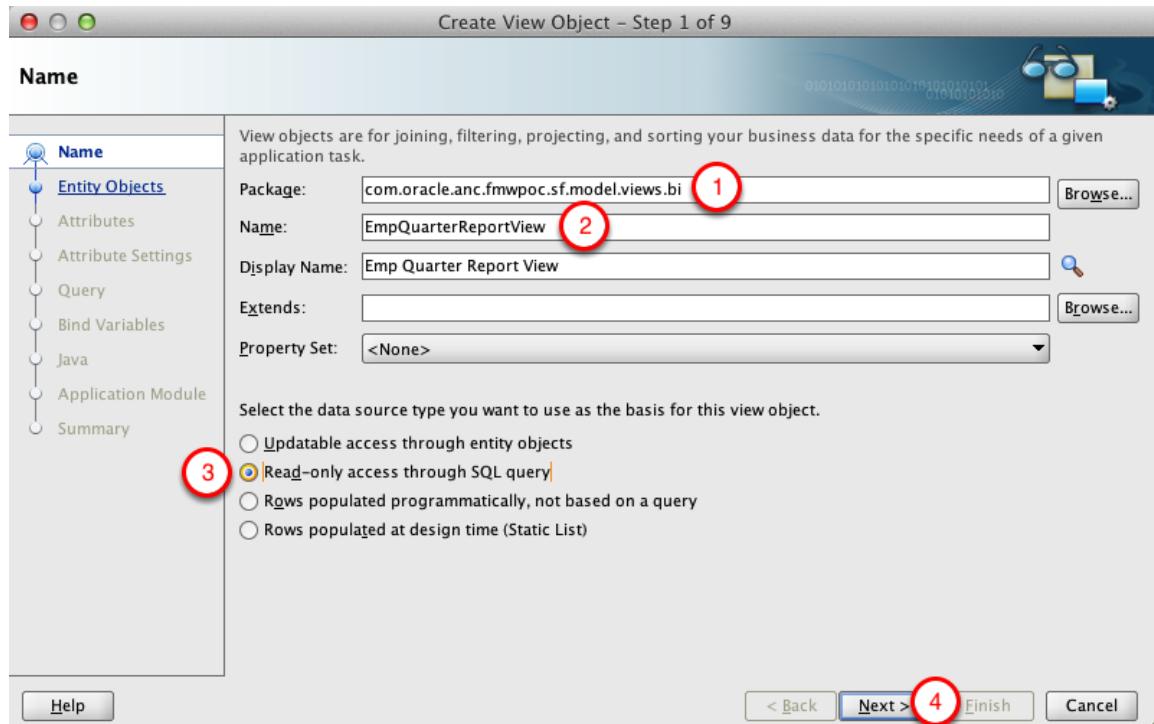
38. Gauges

Create New View Object



1. Locate and right-click views package in the Model project
2. Select New View Object from the menu

Create View Object - Step 1 of 9



1. Append .bi to the Package name
2. Set name to EmpQuarterReportView
3. Select Read-only access through SQL query radio button
4. Click Next

Create View Object - Step 2 of 9

Create View Object – Step 2 of 9

Query

Enter your custom SELECT statement and click Test to check its syntax. Provide the ORDER BY clause separately.

Query Statement

```
SELECT
    VW_EMP_QUARTER_REPORT.EMP_ID EMP_ID,
    VW_EMP_QUARTER_REPORT.QUARTER QUARTER,
    VW_EMP_QUARTER_REPORT.PROBABLE_AMOUNT PROBABLE_AMOUNT,
    VW_EMP_QUARTER_REPORT.CLOSED_AMOUNT CLOSED_AMOUNT,
    VW_EMP_QUARTER_REPORT.EMP_QUARTERLY_TARGET_AMOUNT EMP_QUARTERLY_TARGET_AMOUNT
FROM
    VW_EMP_QUARTER_REPORT
WHERE
    VW_EMP_QUARTER_REPORT.EMP_ID = :pEmpId
```

Query Clauses

Order By:

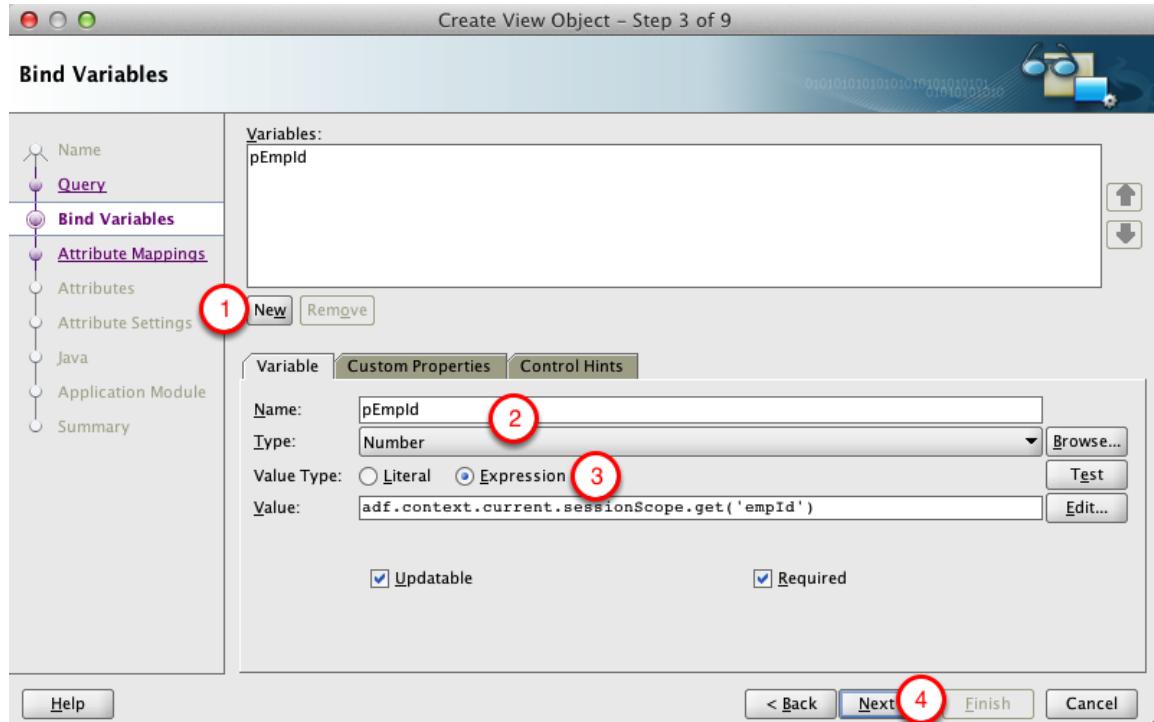
Binding Style:

SQL Mode:

1. Paste the query
2. Click Next

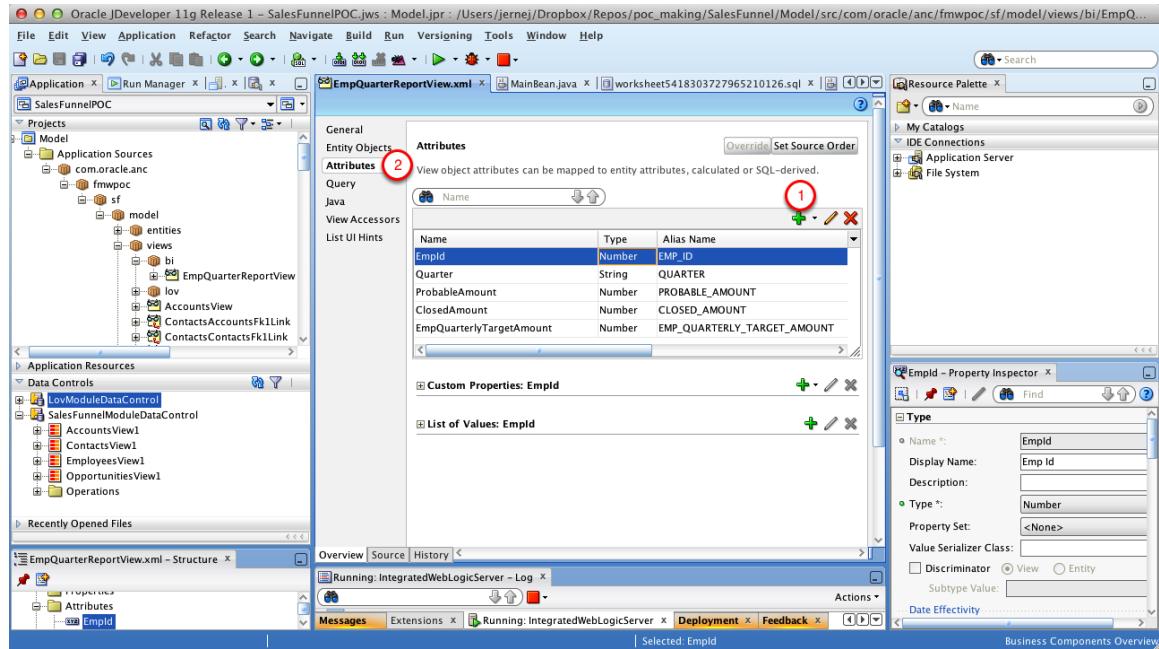
```
SELECT
    VW_EMP_QUARTER_REPORT.EMP_ID EMP_ID,
    VW_EMP_QUARTER_REPORT.QUARTER QUARTER,
    VW_EMP_QUARTER_REPORT.PROBABLE_AMOUNT PROBABLE_AMOUNT,
    VW_EMP_QUARTER_REPORT.CLOSED_AMOUNT CLOSED_AMOUNT,
    VW_EMP_QUARTER_REPORT.EMP_QUARTERLY_TARGET_AMOUNT EMP_QUARTERLY_TARGET_AMOUNT
FROM
    VW_EMP_QUARTER_REPORT
WHERE
    VW_EMP_QUARTER_REPORT.EMP_ID = :pEmpId
```

Create View Object - Step 3 of 9



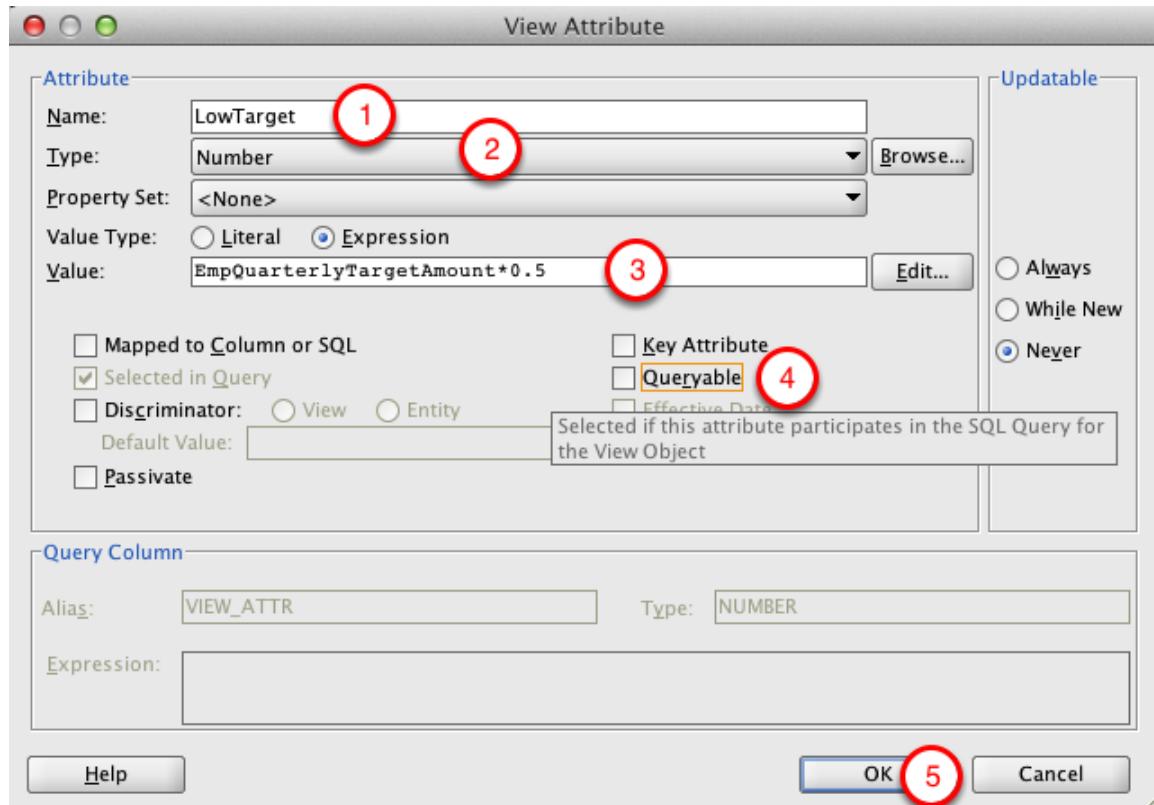
1. Click New button
2. Set Name to pEmpId and Type to Number
3. set Value Type to expression and set Value to adf.context.current.sessionScope.get('empld')
4. Click Next
5. Click Finish on the next screen

Add calculated attributes



1. Open Attributes tab
2. Add new Attribute

View Attribute



1. Set name to LowTarget
2. Set Type to Number
3. Set Value Type to expression and Value to

$(\text{EmpQuarterlyTargetAmount} \neq \text{null} ? \text{EmpQuarterlyTargetAmount} : 0) * 0.5$

4. Uncheck Queryable attribute
5. Click OK

View Attribute

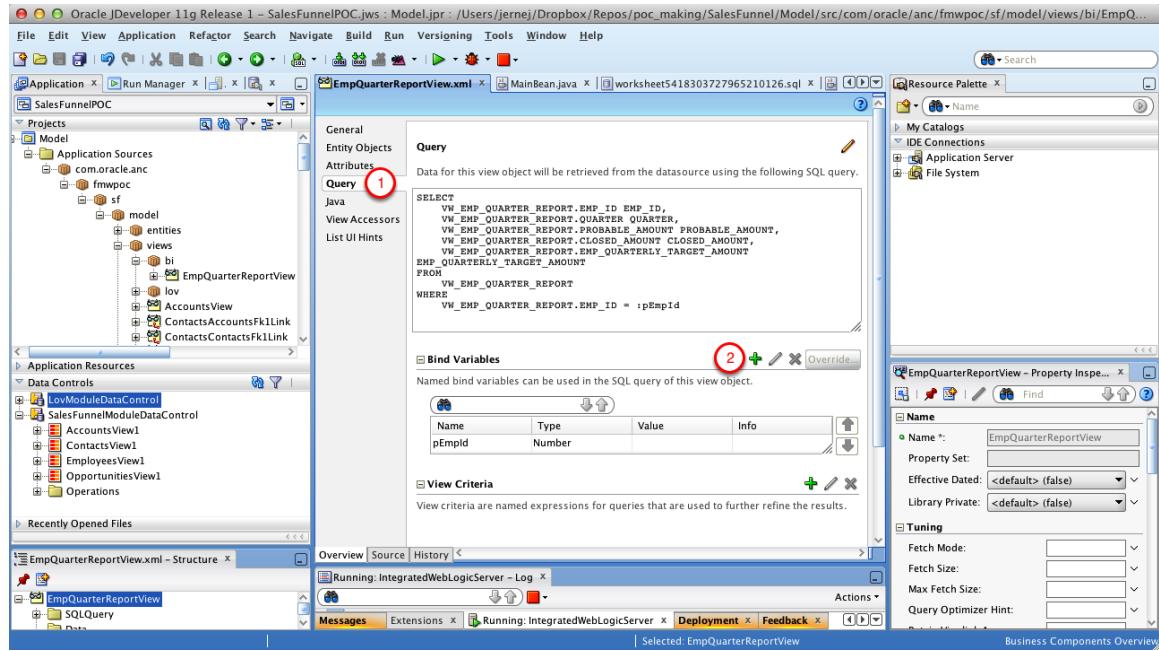
Repeat the previous step to define additional attributes:

1. MediumTarget, expression:

$(\text{EmpQuarterlyTargetAmount} \neq \text{null} ? \text{EmpQuarterlyTargetAmount} : 0) * 0.75$

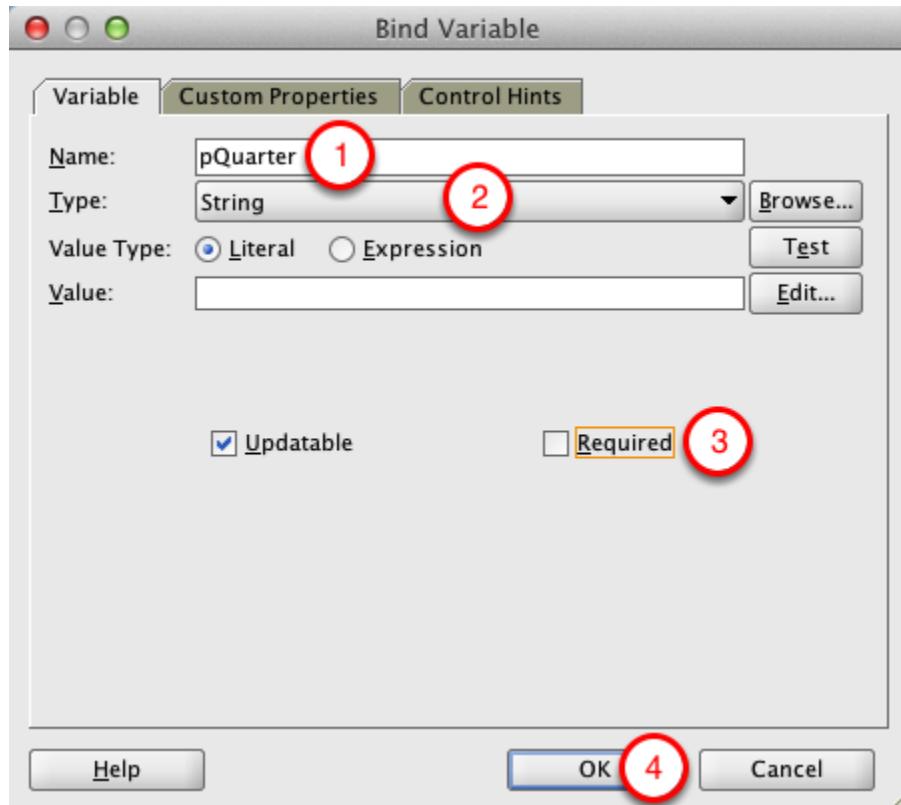
2. MaxTarget, expression: $\text{ClosedAmount} > \text{EmpQuarterlyTargetAmount} ? \text{ClosedAmount} : \text{EmpQuarterlyTargetAmount}$

View Attribute



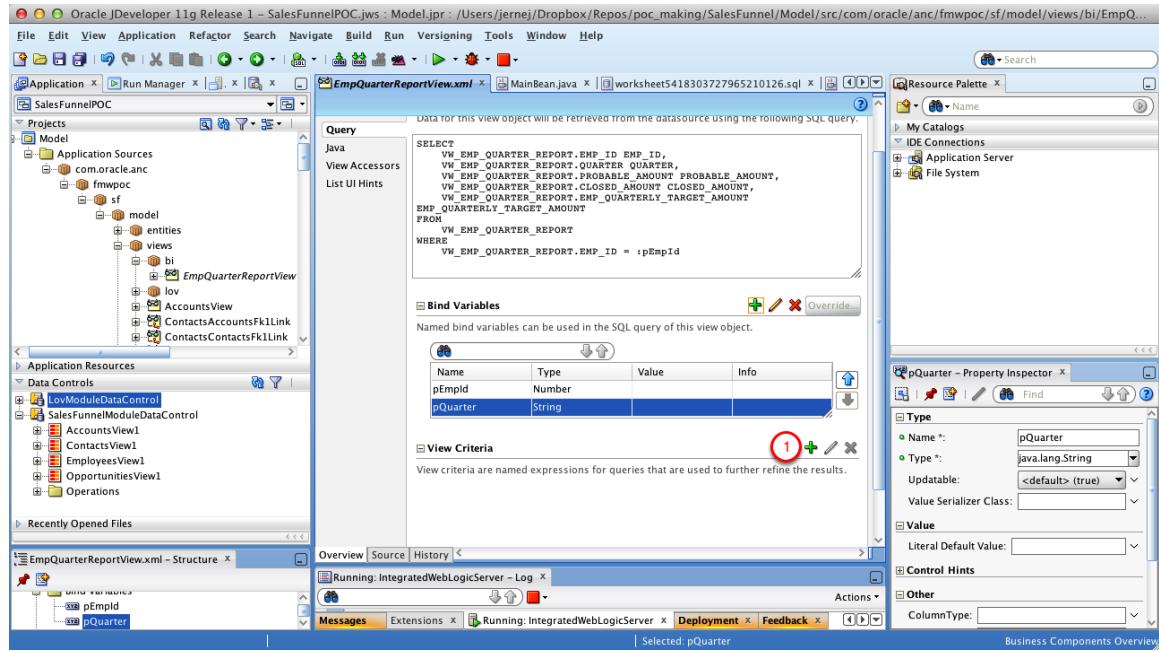
1. Open Query tab
2. Add new Bind Variable

Bind Variable



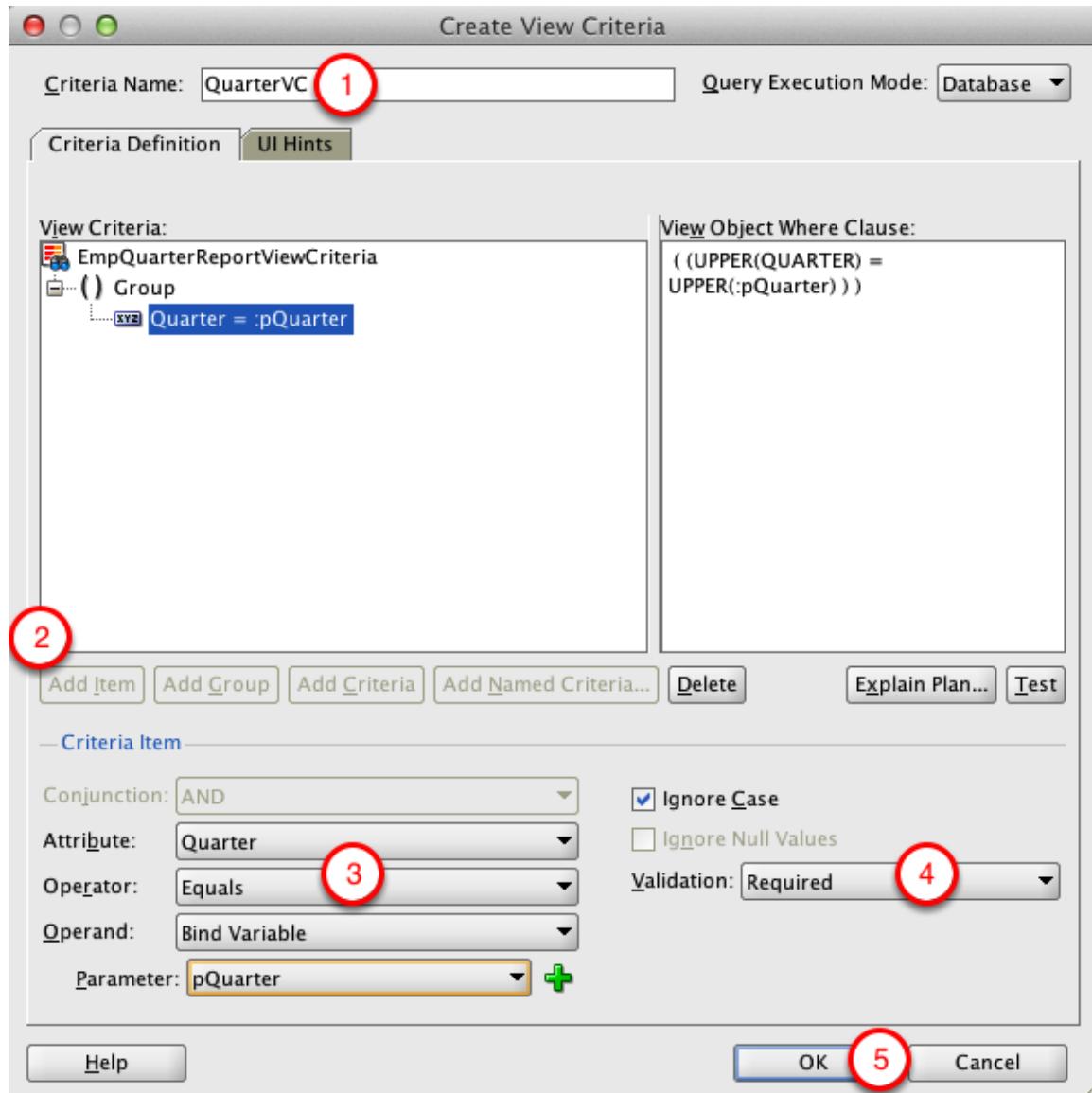
1. Set Name to pQuarter
2. Set Type to String
3. Uncheck Required
4. Click OK

Add View Criteria



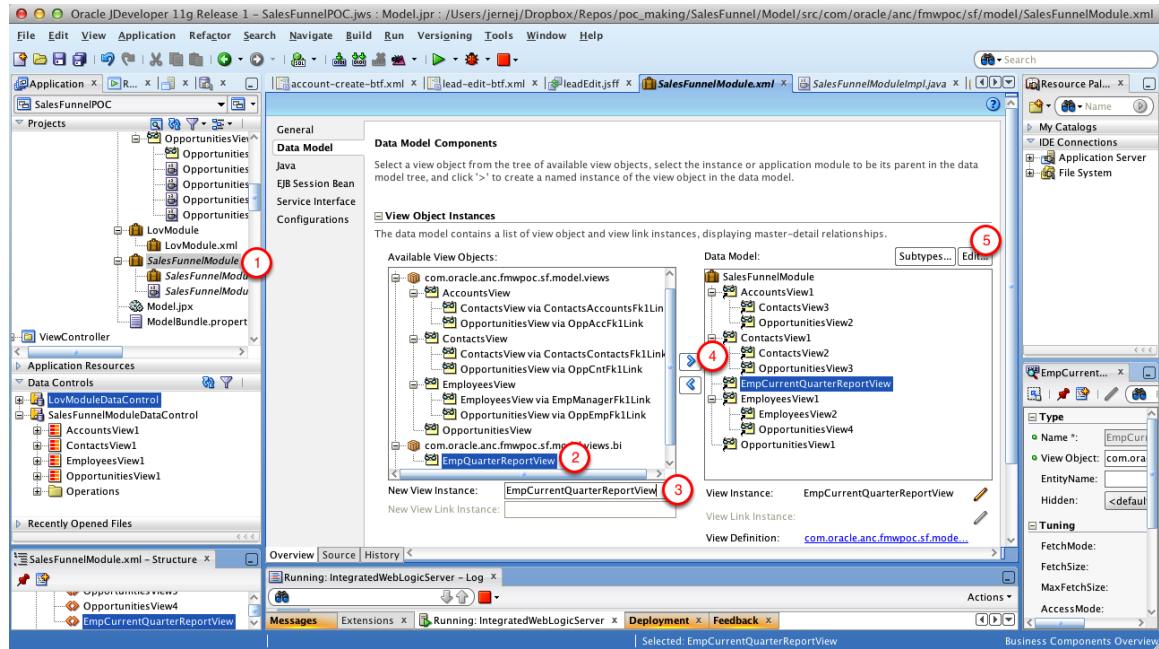
1. Add View Criteria

Create View Criteria



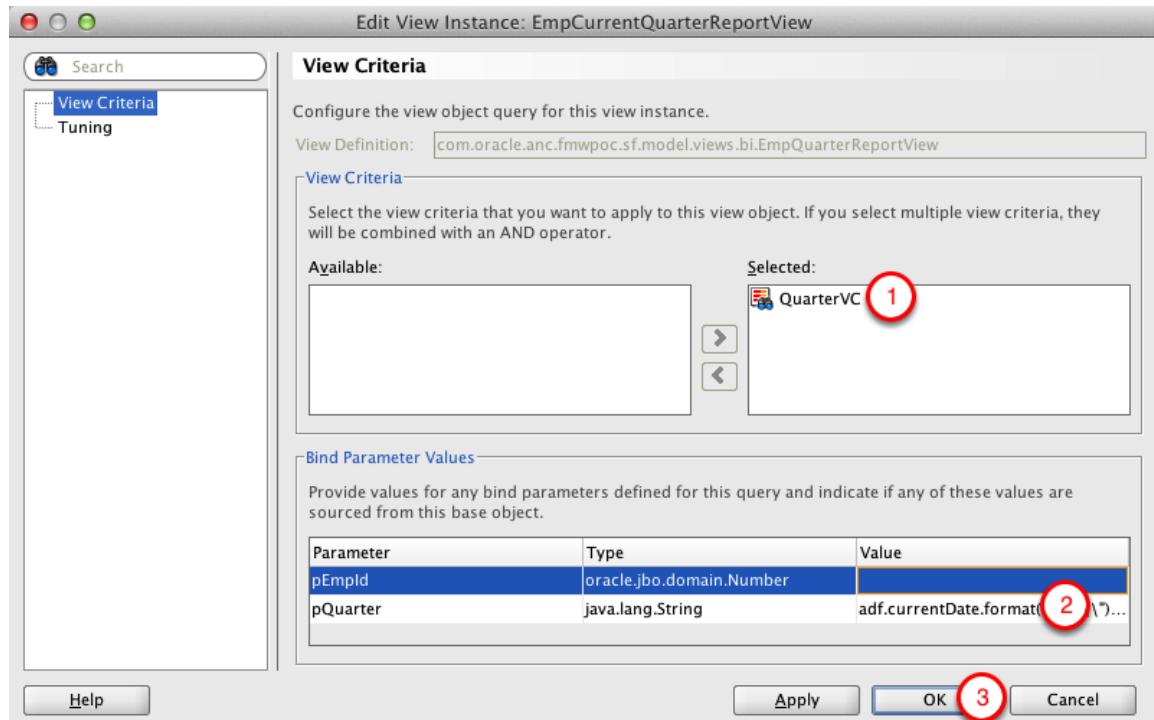
1. Set Name to QuarterVC
2. Click Add Item
3. Set Attribute=Quarter, Operator=Equals, Operand= Bind Variable, Parameter=pQuarter
4. Set Validation to Required
5. Click OK

Add View Object to SalesFunnelModule



1. Open SalesFunnelModule
2. SelectEmpQuarterReportView
3. Set New View Instance to EmpCurrentQuarterReportView
4. Slide it to the right
5. Click Edit button

Edit View Instance: EmpCurrentQuarterReportView

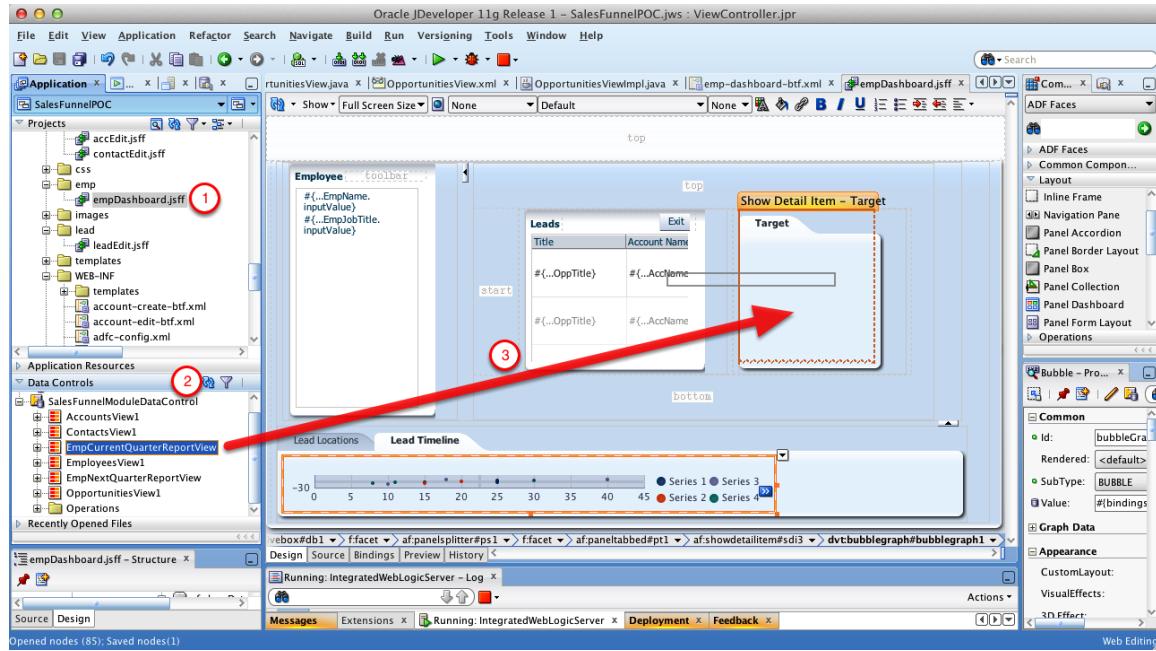


1. Slide QuarterVC to the right
2. Enter value (without quotes) "adf.currentDate.format("yyyy\\\"\\\""+((adf.currentDate.getMonth()/3).setScale(0,1)+1)+"")..." for pQuarter Value
3. Click OK

Edit View Instance: EmpCurrentQuarterReportView

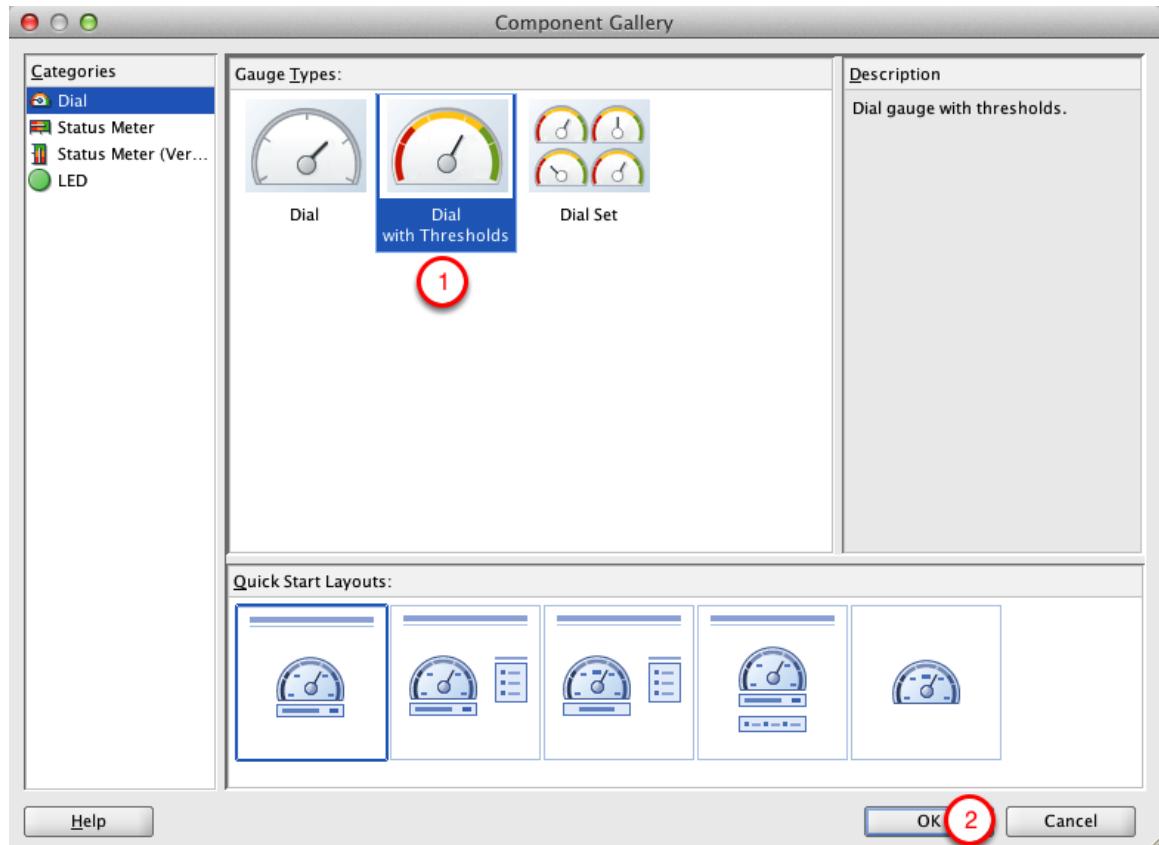
1. Repeat previous two steps to define EmployeeNextQuarterReportView with "1" as the value for pQuarter

Add target gauge



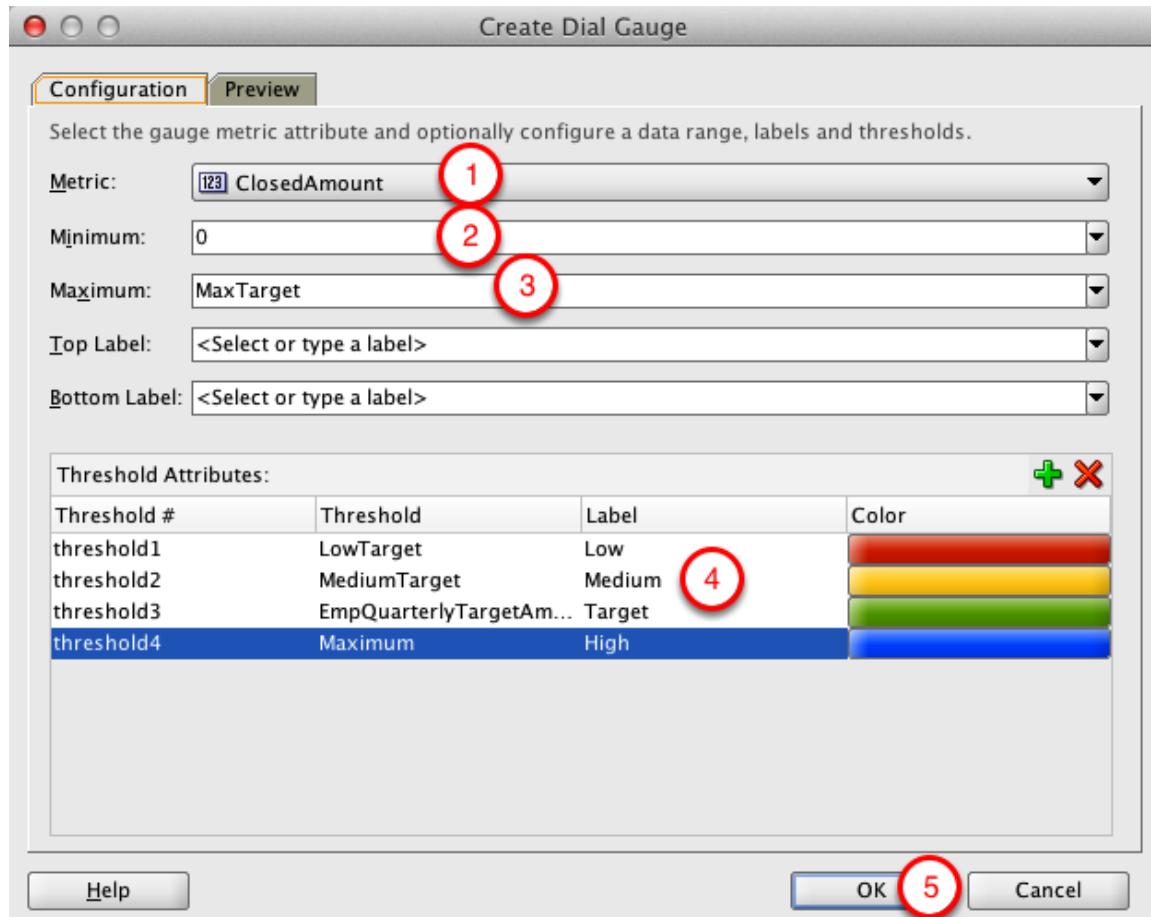
1. Select `empDashboard.jsff`
2. Click refresh in Data Controls
3. Drag and drop `EmpCurrentQuarterReportView` to the target tab
4. Select Gauge from the Menu

Component Gallery



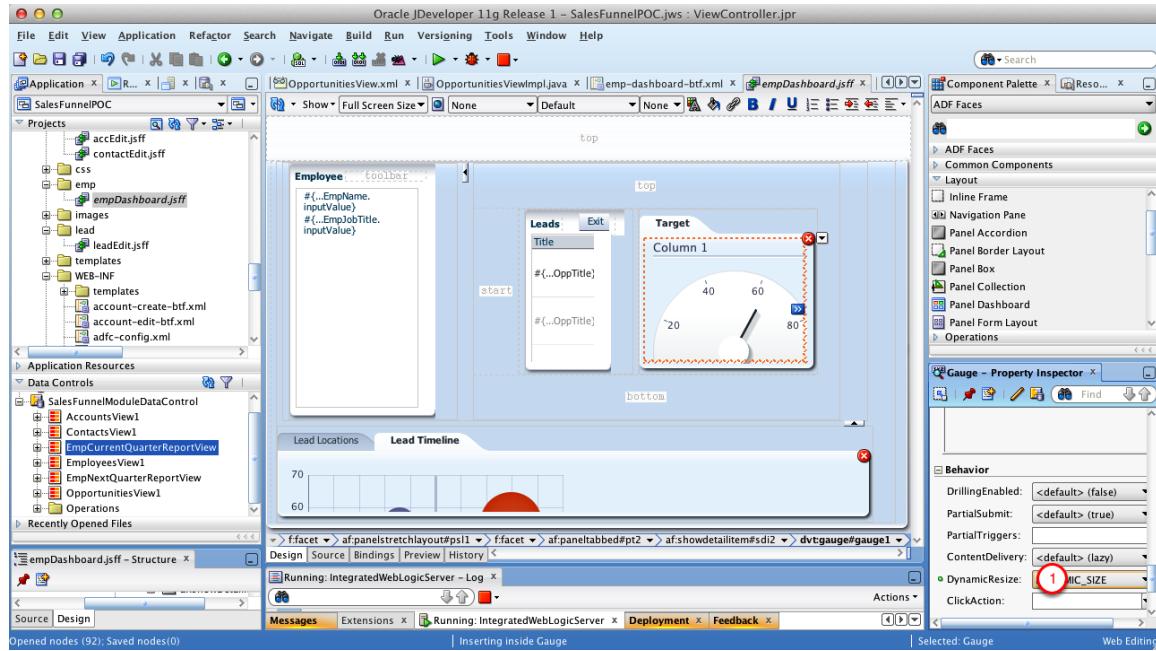
1. Select Dial with Thresholds
2. Click OK

Create Dial Gauge



1. Set Metric to ClosedAmount
2. Set Minimum to 0
3. Set Maximum to MaxTarget
4. Set: threshold1 to LowTarget, threshold2 to MediumTarget, add new threshold3, EmpQuarterlyTargetAmount and change color to green, change color of Maximum to Blue
5. Click OK

Set Gauge Properties



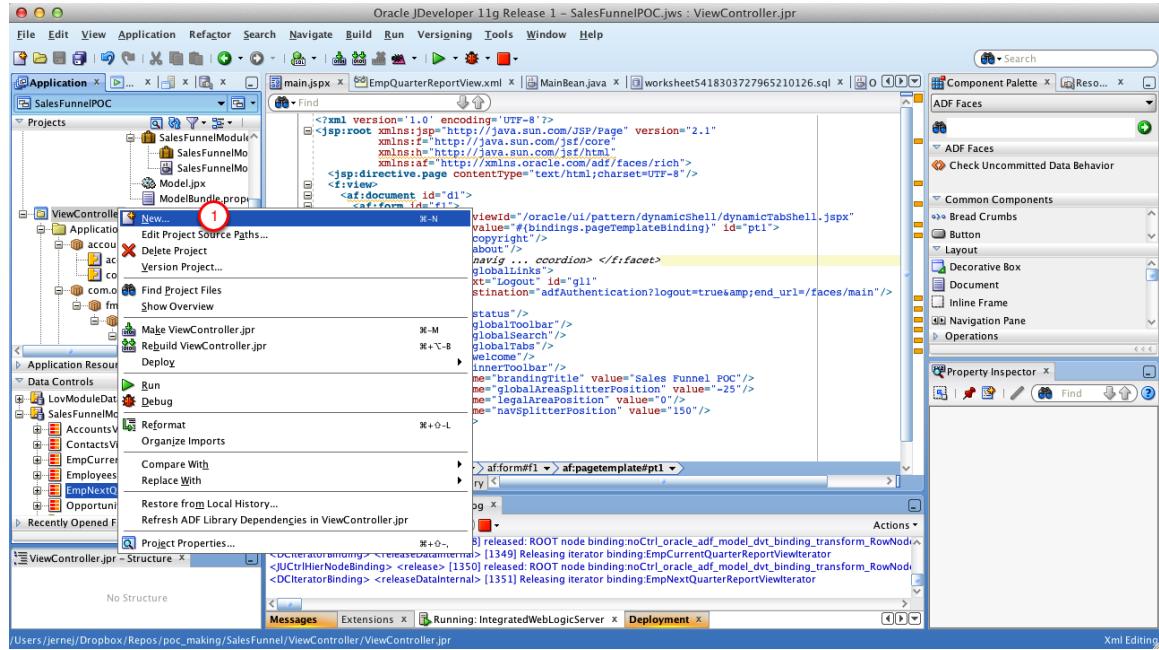
1. Set DynamicResize property of the gauge to DYNAMIC_SIZE

Exercise

Repeat previous steps to define a gauge for the next quarter in a new showDetailItem component.

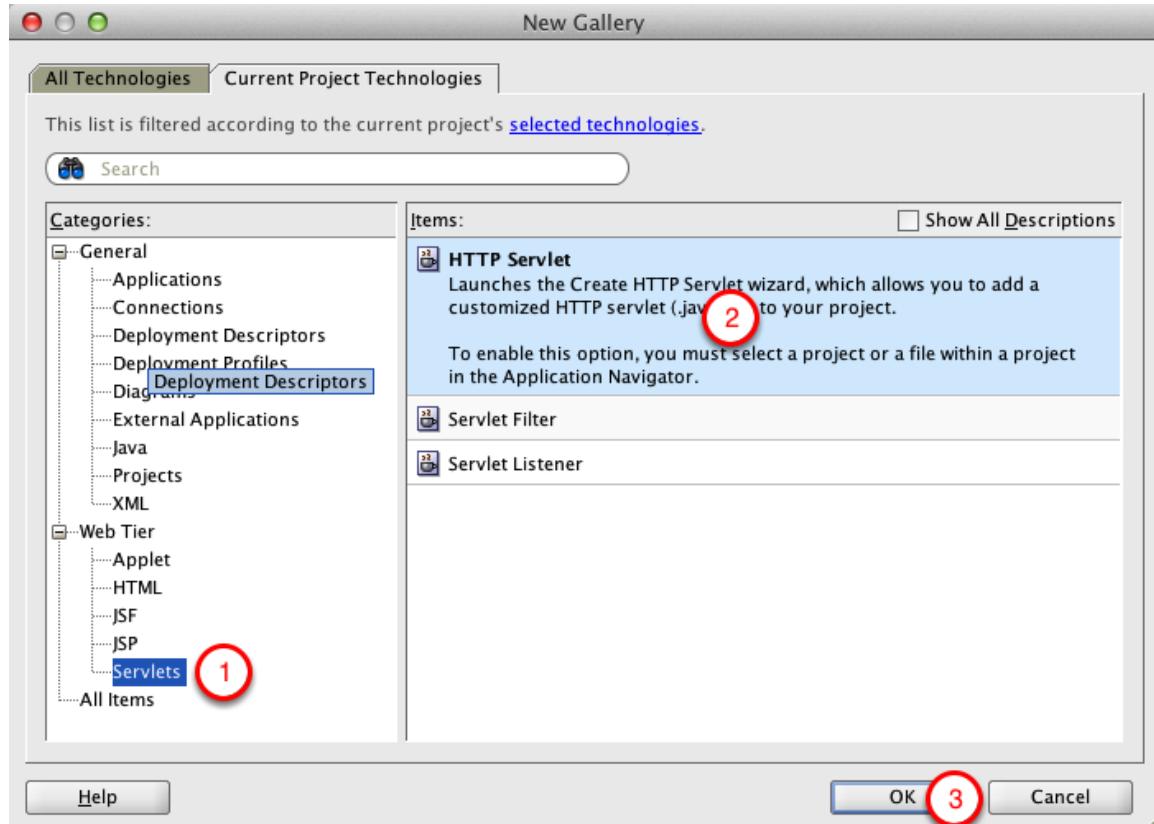
39. Image Servlet

Create HTTP ImageServlet



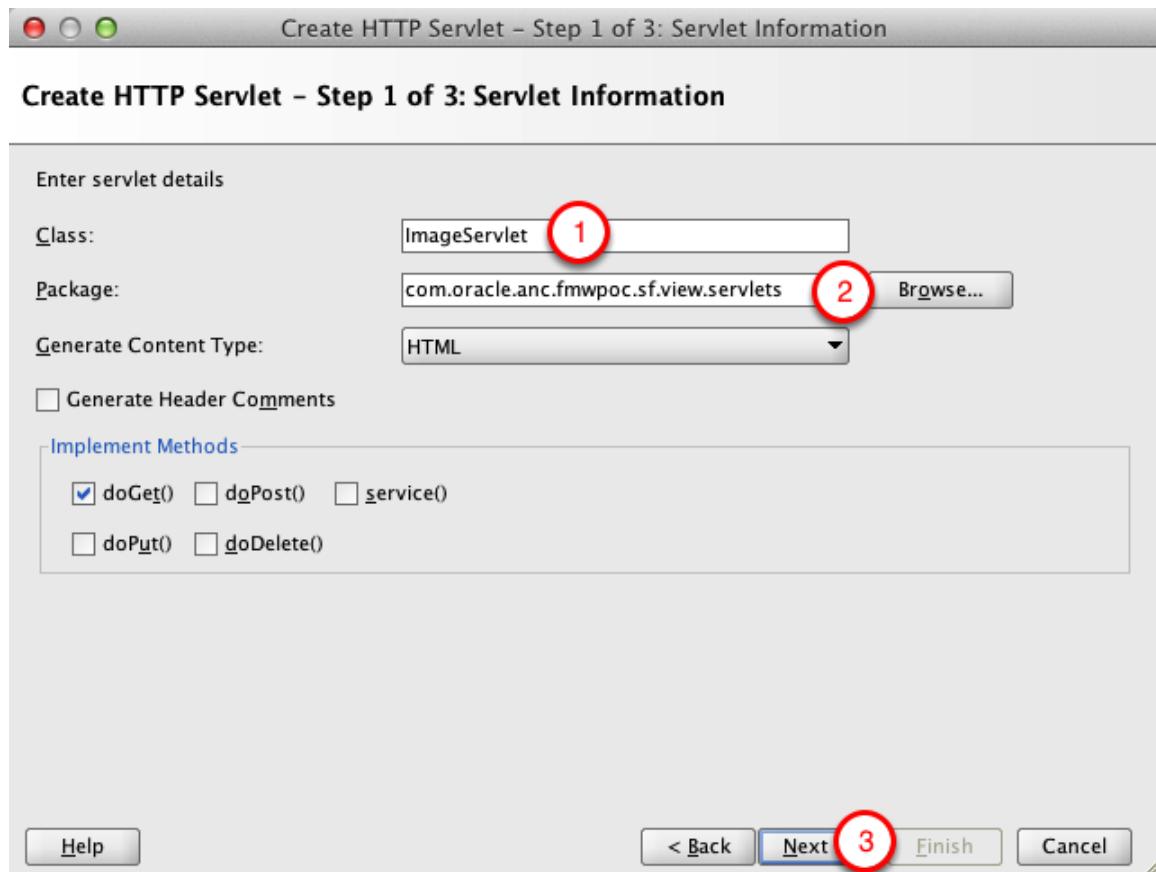
1. Right-click on the ViewController project and select New

New Gallery



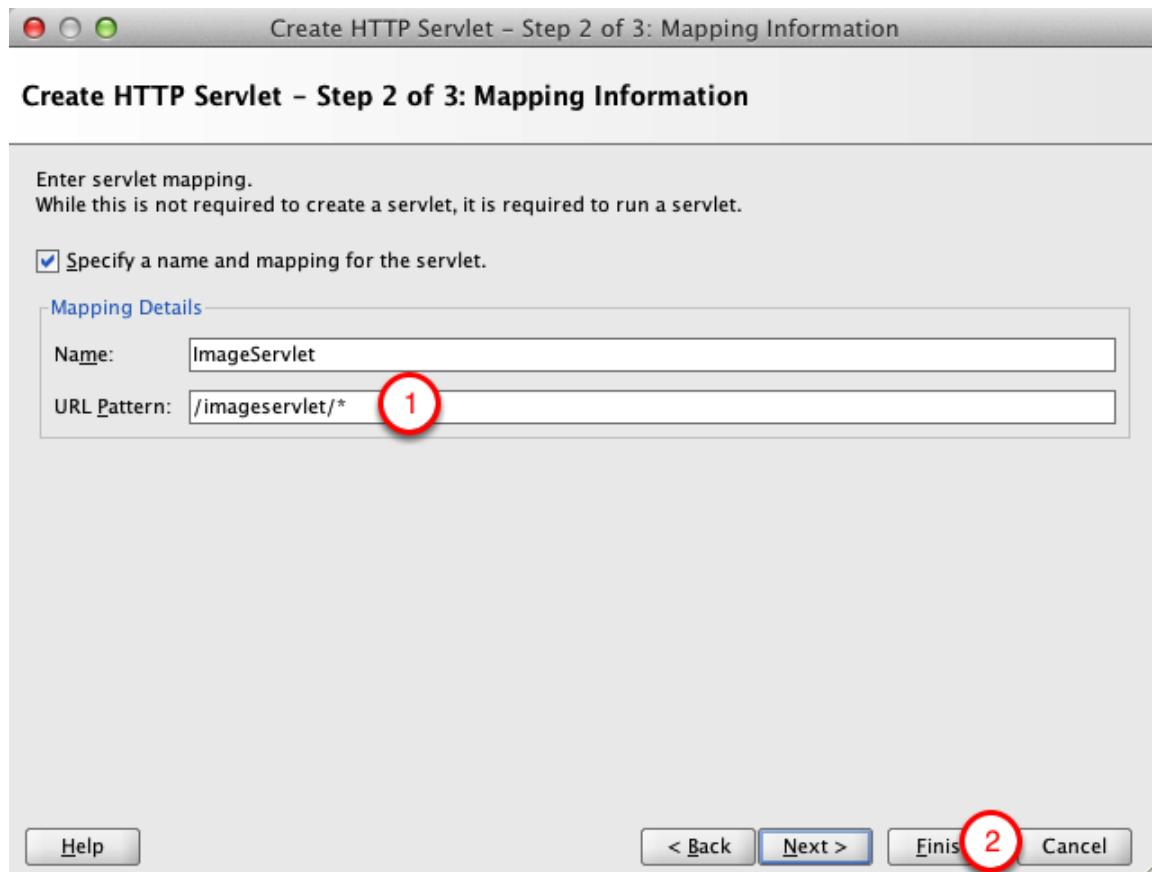
1. Select Servlets
2. Select HTTP Servlet
3. Click OK

Create HTTP Servlet - Step 1 of 3: Servlet Information



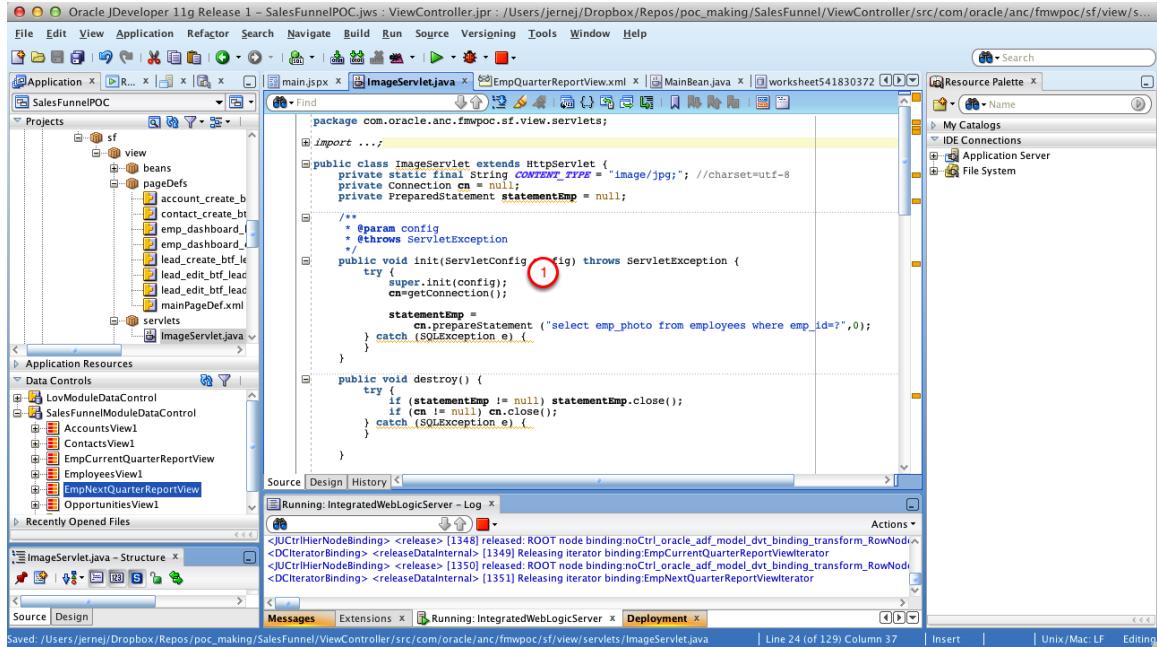
1. Set Class name to ImageServlet
2. Set package to com.oracle.anc.fmwpoc.sf.view.servlets
3. Click Next

Create HTTP Servlet - Step 2 of 3: Mapping Information



1. Set URL Pattern to /imageservlet/*

Paste Servlet implementation



1. Paste / replace the code of the servlet

```
package com.oracle.anc.fmw poc.sf.view.servlets;

import java.io.BufferedInputStream;
import java.io.IOException;
import java.io.OutputStream;

import java.sql.Blob;
import java.sql.Connection;
import java.sql.PreparedStatement;

import java.sql.ResultSet;
import java.sql.SQLException;

import javax.naming.InitialContext;
import javax.naming.NamingException;

import javax.servlet.*;
import javax.servlet.http.*;

import javax.sql.DataSource;
import javax.sql.rowset.serial.SerialBlob;

import oracle.jbo.domain.BlobDomain;

public class ImageServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "image/jpg"; //charset=utf-8
    private Connection cn = null;
    private PreparedStatement statementEmp = null;

    /**
     * @param config
     * @throws ServletException
     */
    public void init(ServletConfig config) throws ServletException {
        try {
            super.init(config);
            cn= getConnection();

            statementEmp =
                cn.prepareStatement ("select emp_photo from employees where emp_id=?",0);
        } catch (SQLException e) {
        }
    }

    public void destroy() {

```

```

        try {
            if (statementEmp != null) statementEmp.close();
            if (cn != null) cn.close();
        } catch (SQLException e) {
        }

    }

    /**
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException,
                                                     IOException {

        response.setContentType(CONTENT_TYPE);
        response.addHeader("Cache-Control", "max-age=0");
        String empId = request.getParameter("empId");
        String oppId = request.getParameter("oppId");
        OutputStream os = response.getOutputStream();

        try {
            if (empId != null) {
                processEmp(empId, os, request);
            }
            response.flushBuffer();
        } catch (Exception e) {
            System.out.println(e);
        } finally {
        }
    }

    private void processEmp(String empId, OutputStream os,
                           HttpServletRequest request) throws SQLException,
                                                     IOException {
        Blob blob = null;
        //this part is serving image during HTML5 upload
        if (request.getSession().getAttribute("tmpPhoto") != null) {
            blob =
                new SerialBlob((BlobDomain)request.getSession().getAttribute("tmpPhoto")).getStorageByteArray());
            request.getSession().removeAttribute("tmpPhoto");
        }

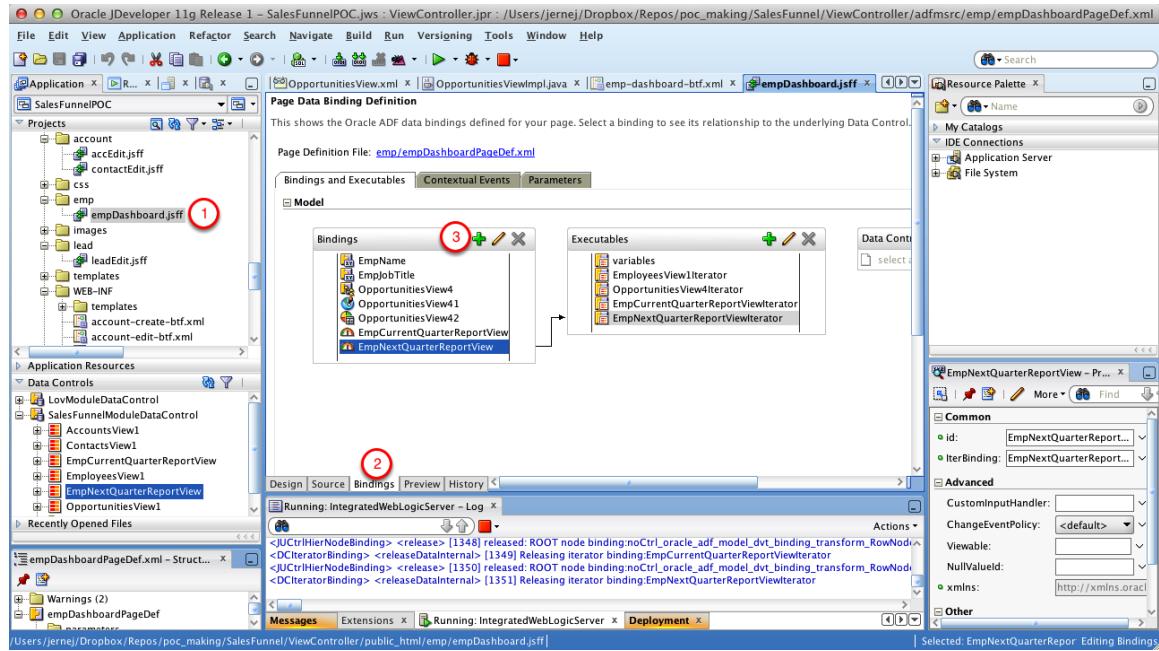
        if (blob == null) {
            synchronized (this) {
                statementEmp.setInt(1, Integer.parseInt(empId));
                ResultSet rs = statementEmp.executeQuery();
                if (rs.next())
                    blob = rs.getBlob("EMP_PHOTO");
            }
        }

        if (blob != null) {
            BufferedInputStream in =
                new BufferedInputStream(blob.getBinaryStream());
            int b;
            byte[] buffer = new byte[10240];
            while ((b = in.read(buffer, 0, 10240)) != -1) {
                os.write(buffer, 0, b);
            }
        }
    }

    private Connection getConnection() {
        Connection connection = null;
        try {
            InitialContext context = new InitialContext();
            DataSource dataSource = (DataSource)context.lookup("jdbc/sfDS");
            connection = dataSource.getConnection();
        } catch (NamingException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }
}

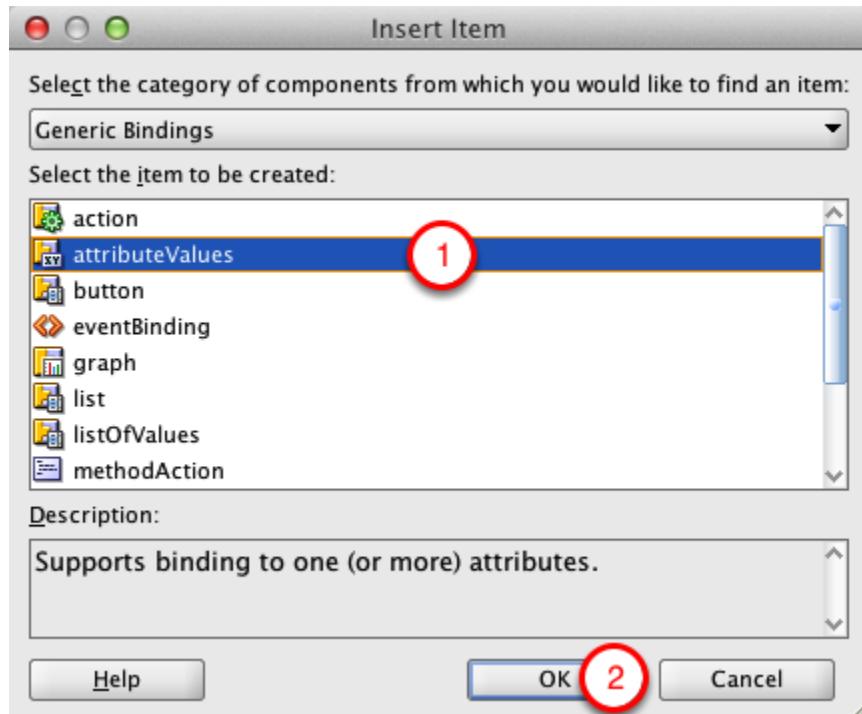
```

Add EmpId attribute binding



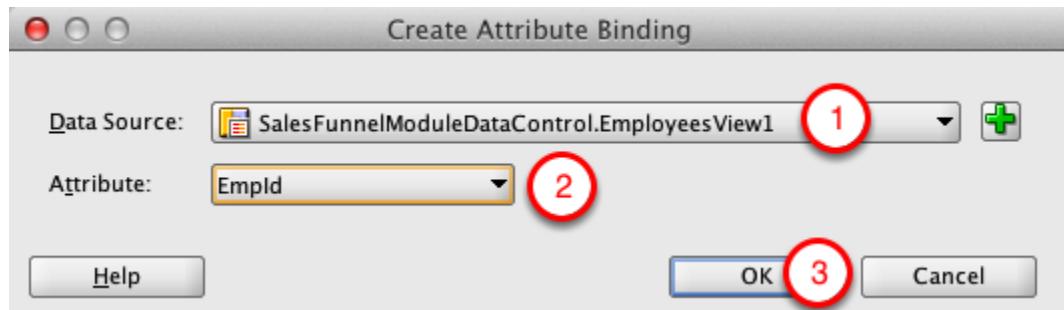
1. Open empDashboard.jsff
2. Open Bindings tab
3. Click the plus next to Bindings

Insert Item



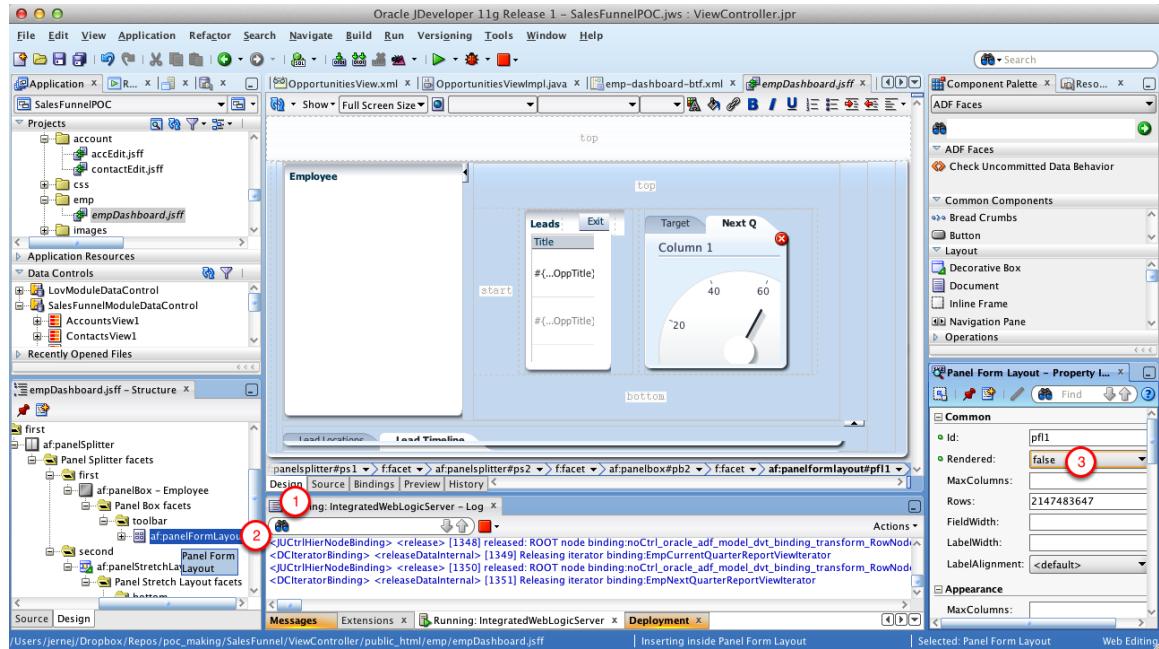
1. Select attributeValues
2. Click OK

Create Attribute Binding



1. Select SalesFunnelModuleDataControl.EmployeesView1 as Data Source
2. Select Empld attribute
3. Click OK

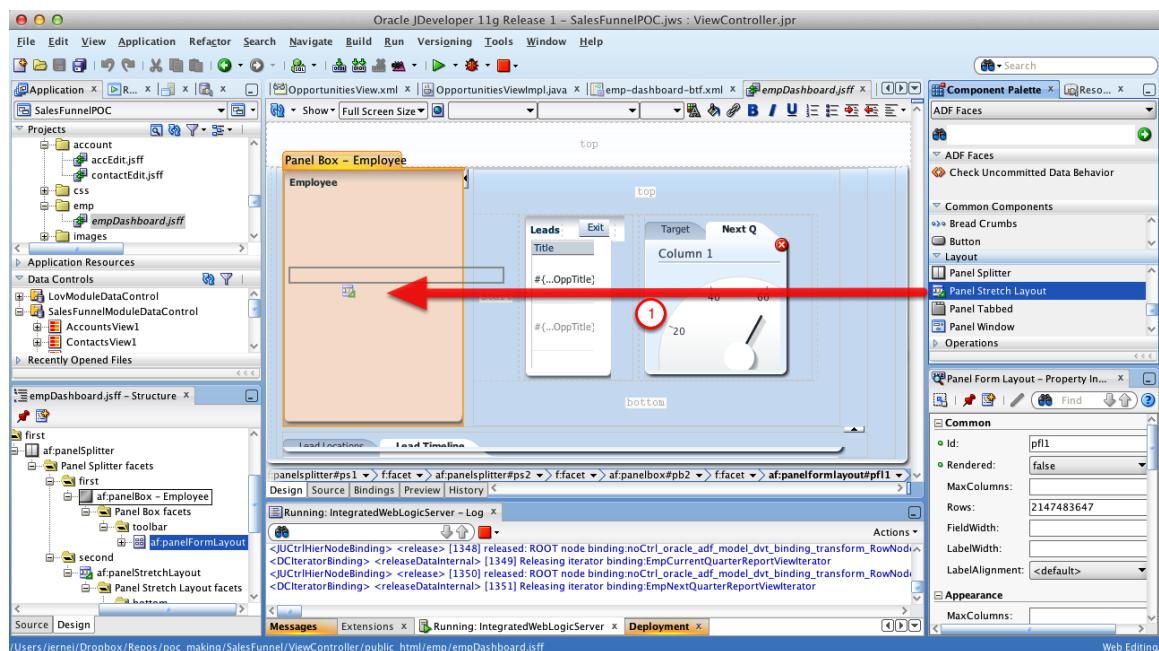
Make Space for new components



1. Open Design View
2. Find panelFormLayout with employee details, move it to the Panel Box toolbar*
3. Set render property to false

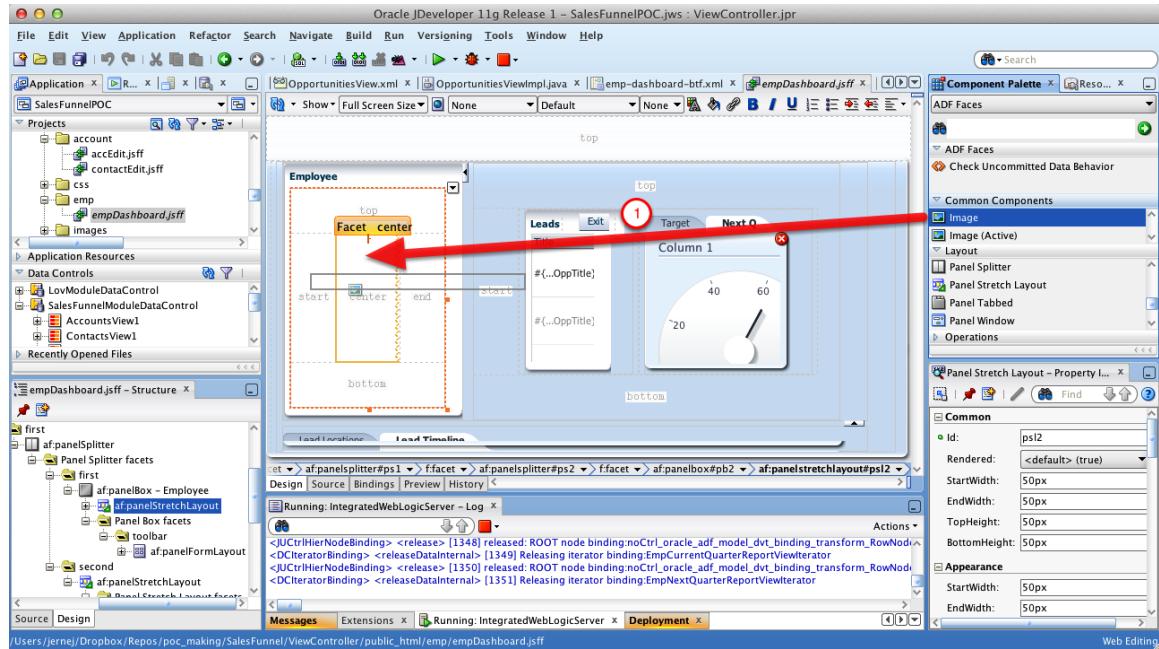
*we just make place for a new layout control that we are going to use

Add Panel Stretch Layout



1. Drag and drop Panel Stretch Layout to the Employee Panel Box

Add Image



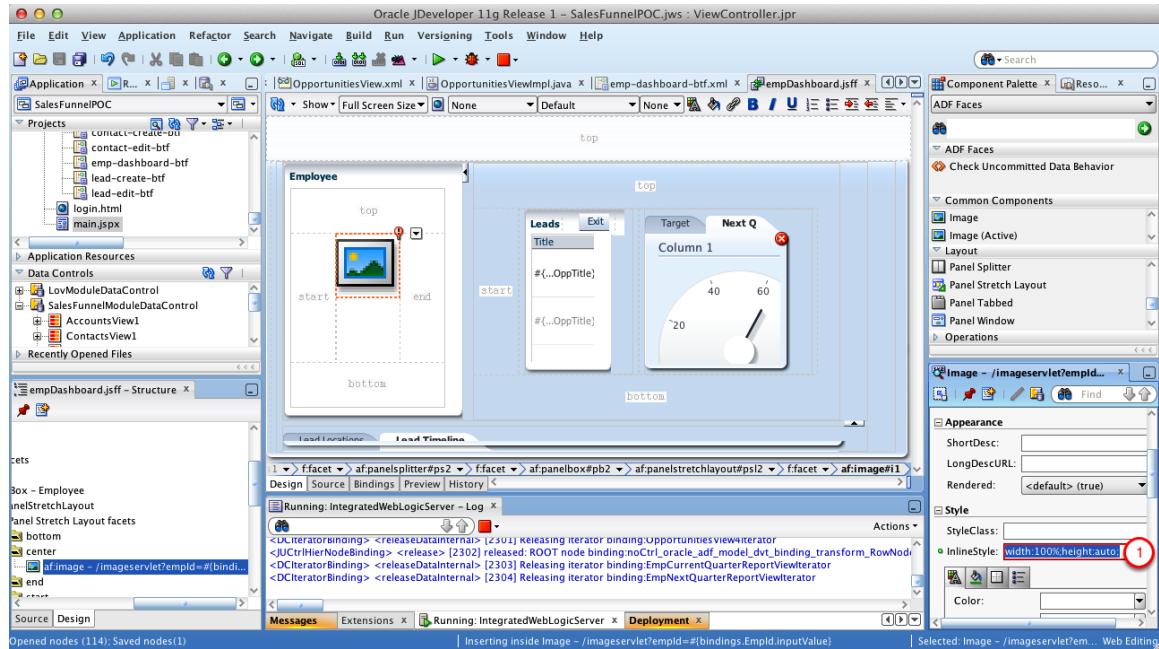
1. Drag and drop Image to the center facet

Insert Image



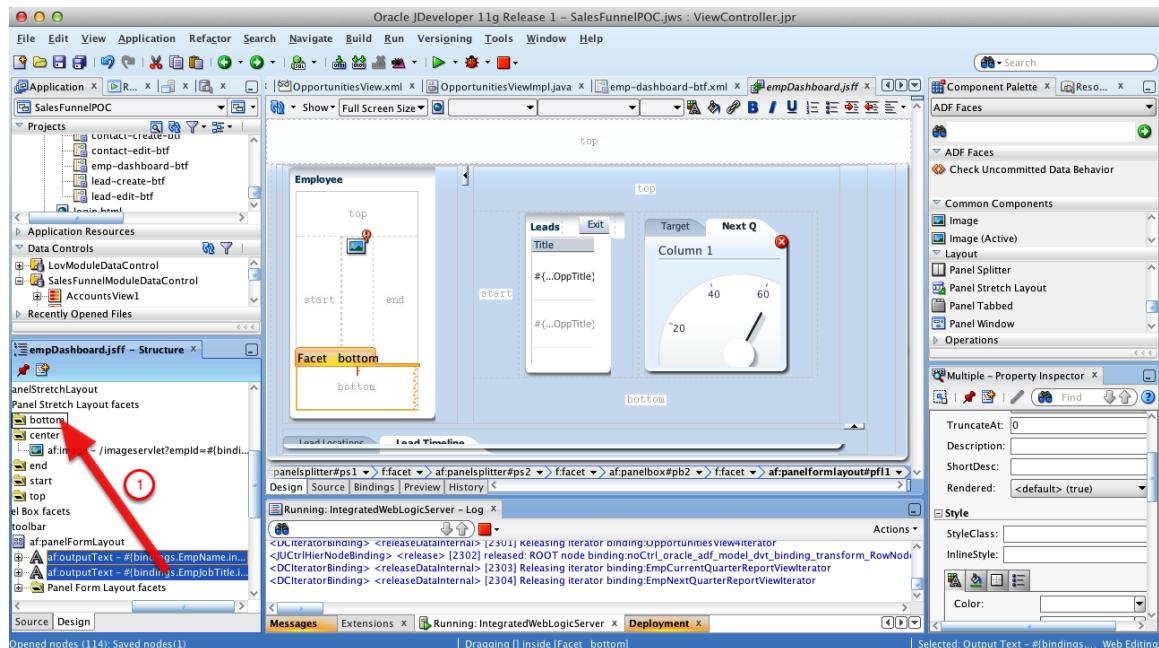
1. Set Source to /imageservlet?empId=#{bindings.EmpId.inputValue}

Set Image inlineStyle



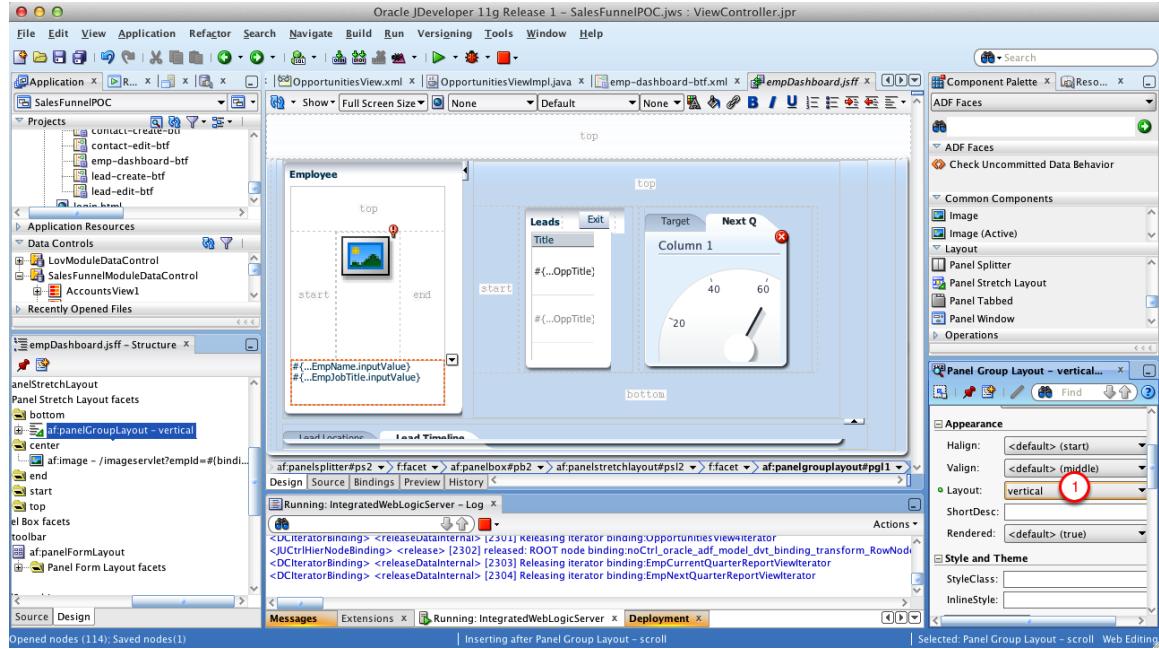
1. Set image InlineStyle to "width:100%; height:auto; " (without quotes)

Rearrange Layout



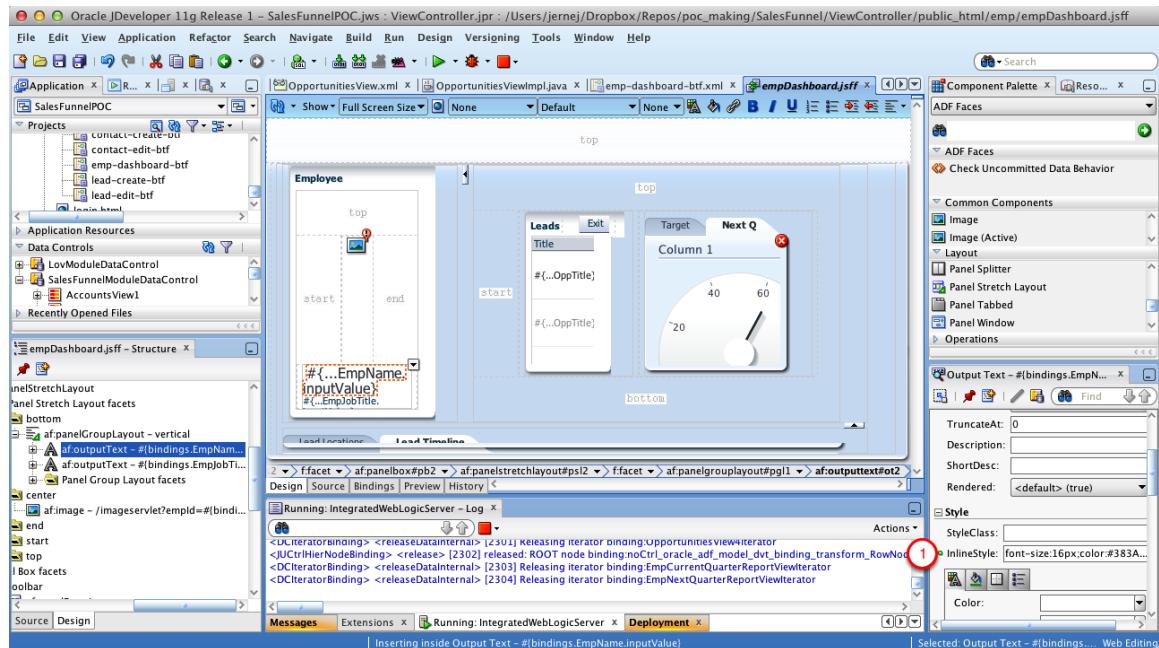
1. Drag and drop EmpName and EmpJobTitle outputText to the bottom facet of the image stretch layout

Change Panel Group Layout orientation



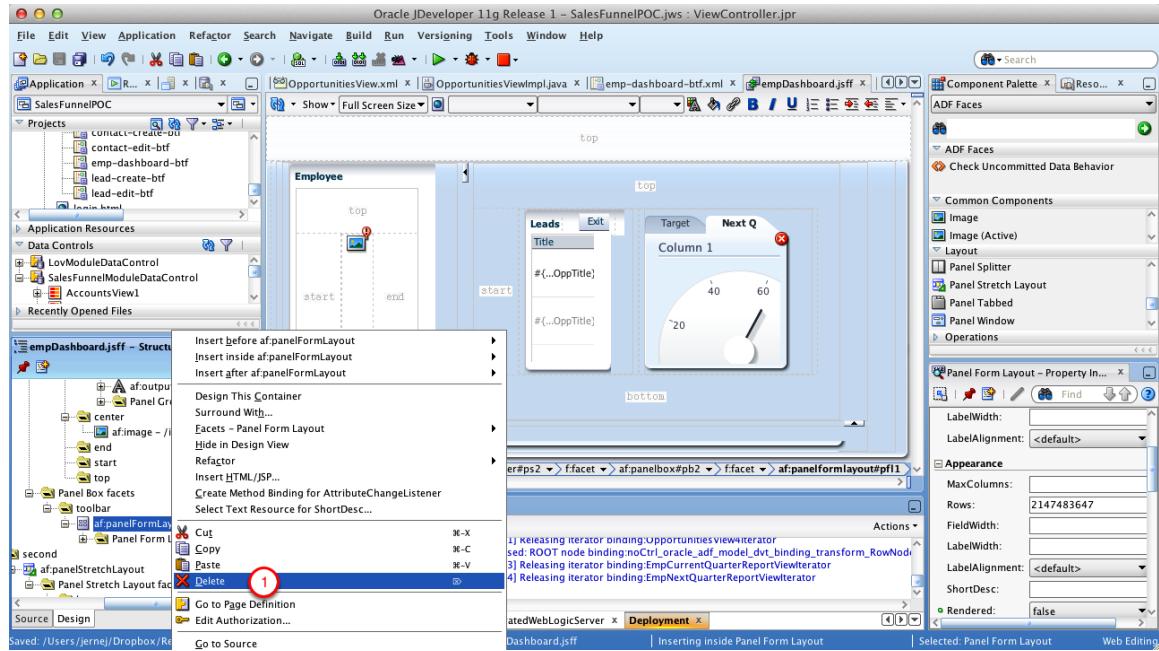
1. Change panelGroupLayout Layout property to vertical

Set Text Style



1. Set InlineStyle of EmpName output text to "font-size:16px;color:#383A47" (without quotes)
2. Set InlineStyle of EmpJobTitle output text to "font-size:11px;font-style:italic;color:#383A47" (without quotes)

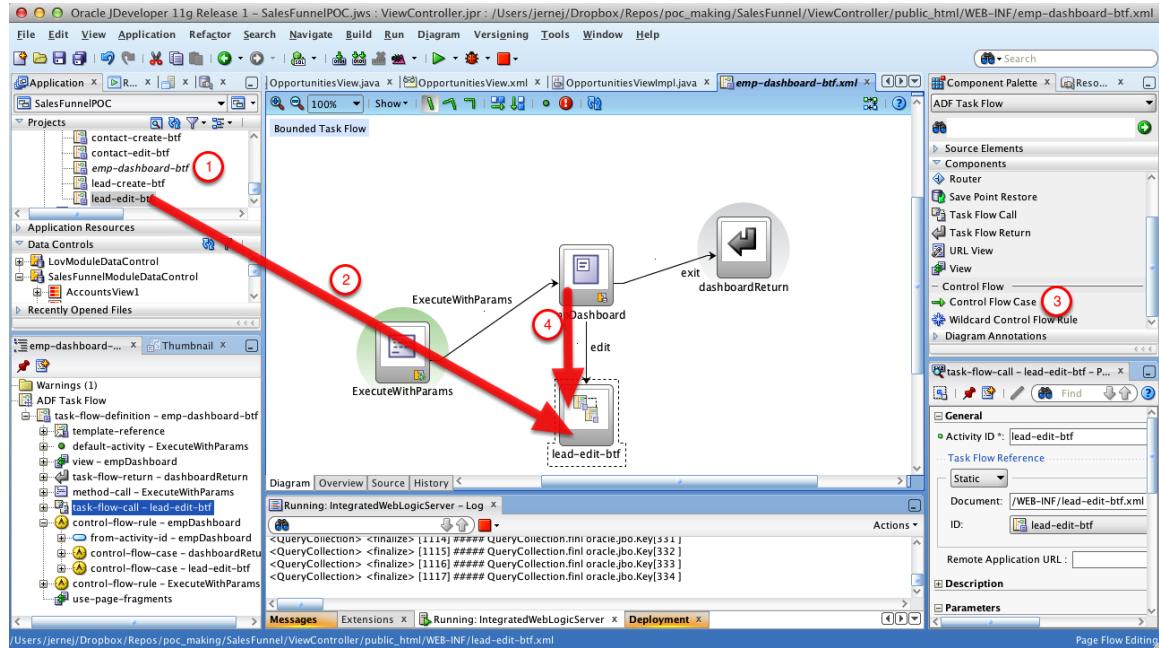
Delete unused Panel Form Layout



1. Right-click and delete panelFormLayout

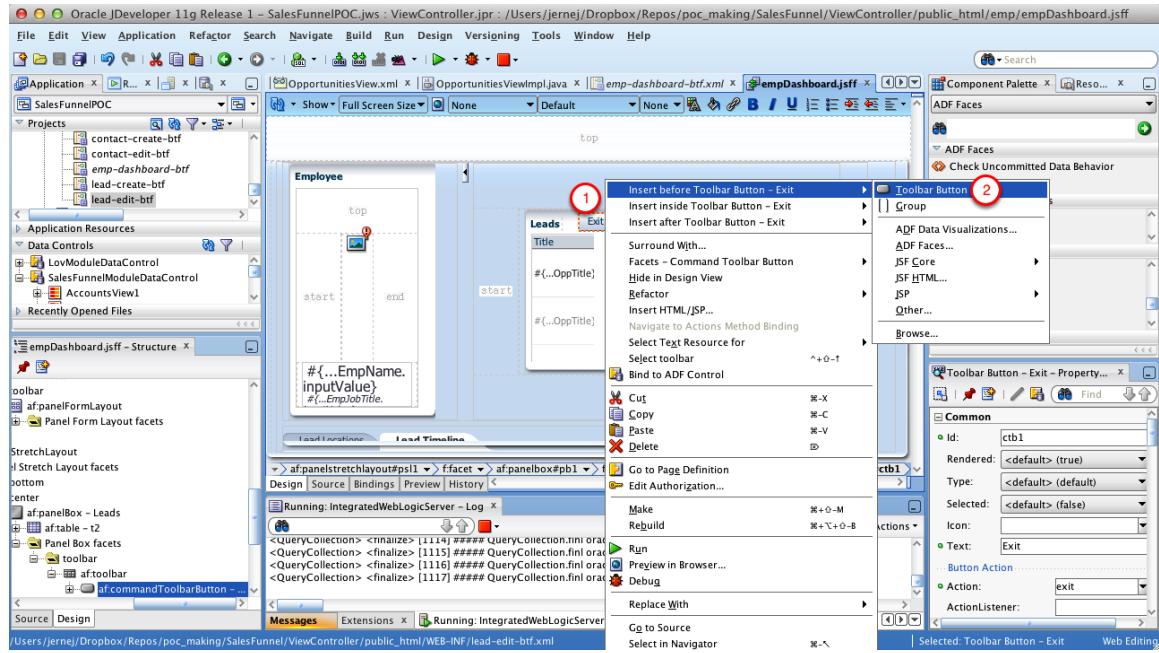
40. Connecting Dashboard to Lead Edit

Add lead-edit-btf to emp-dashboard-btf



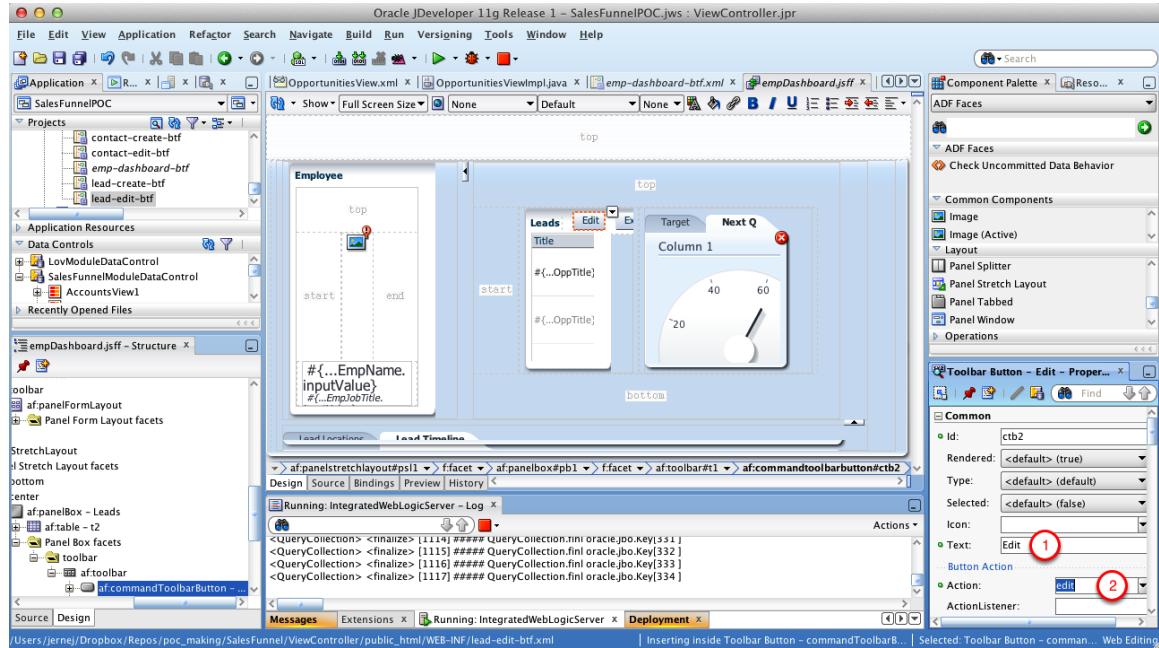
1. Open emp-dashboard-btf
2. Drag and drop lead-edit-btf to the diagram
3. Select Control Flow Case in the Palette
4. Connect empDashboard to lead-edit-btf and name the action "edit" (without quotes)
5. Double-click empDashboard to open it

Create Button



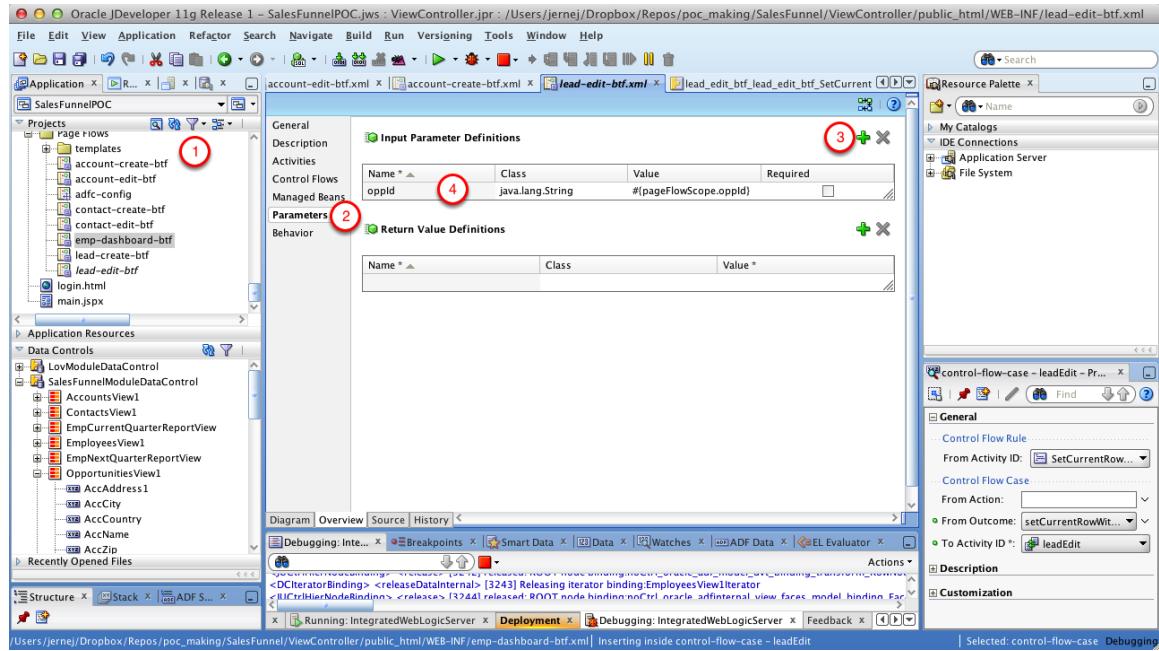
1. Right-click Exit button in the Leads toolbar
2. Select Insert Before > Toolbar Button

Set Button Properties



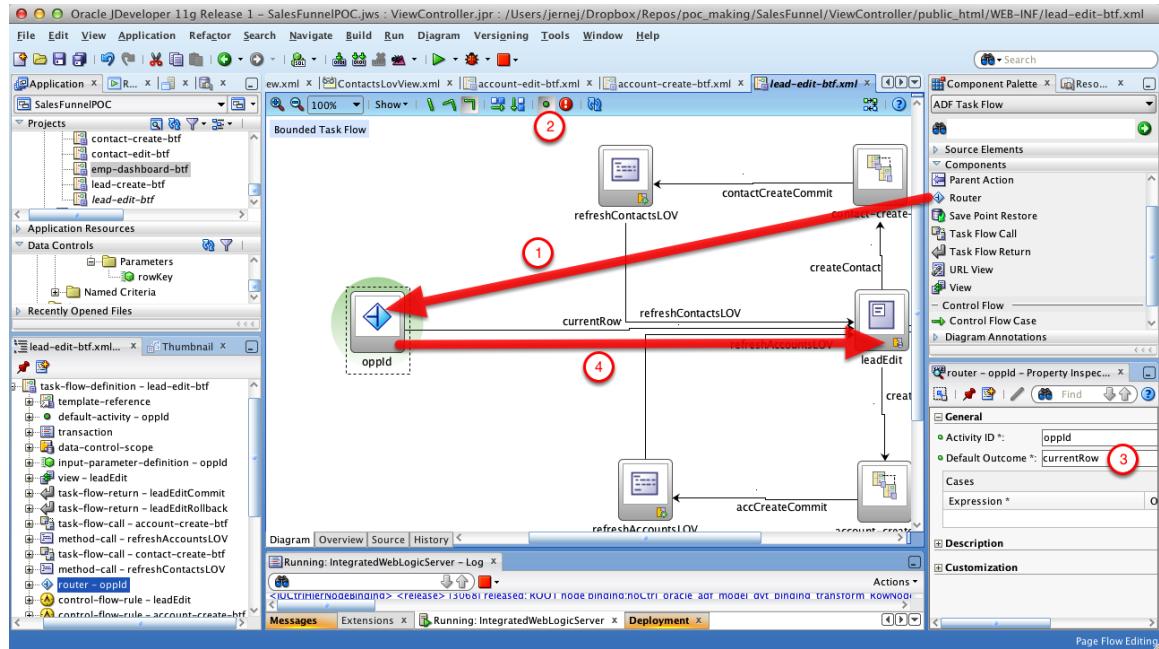
1. Set Text to Edit
2. Set Action to edit

Add Optional Parameter to lead-edit-btf



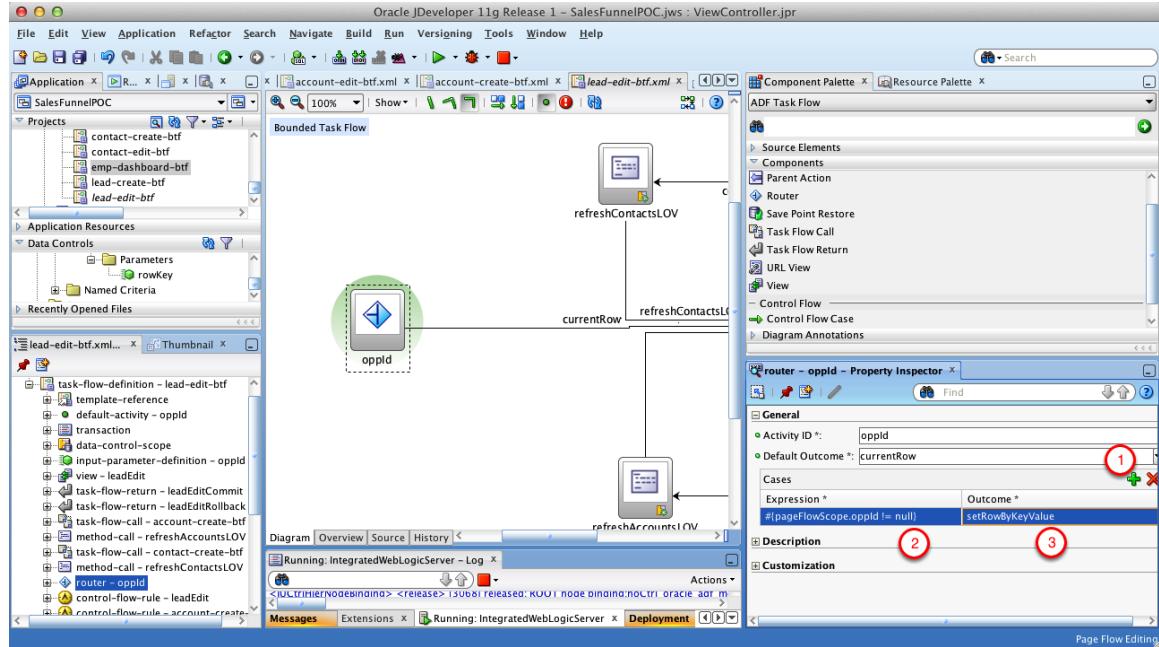
1. Open lead-edit-btf
2. Open Parameters tab
3. Click plus to add new parameter
4. Set properties: Name: oppId, Class: java.langString

Add Router Activity



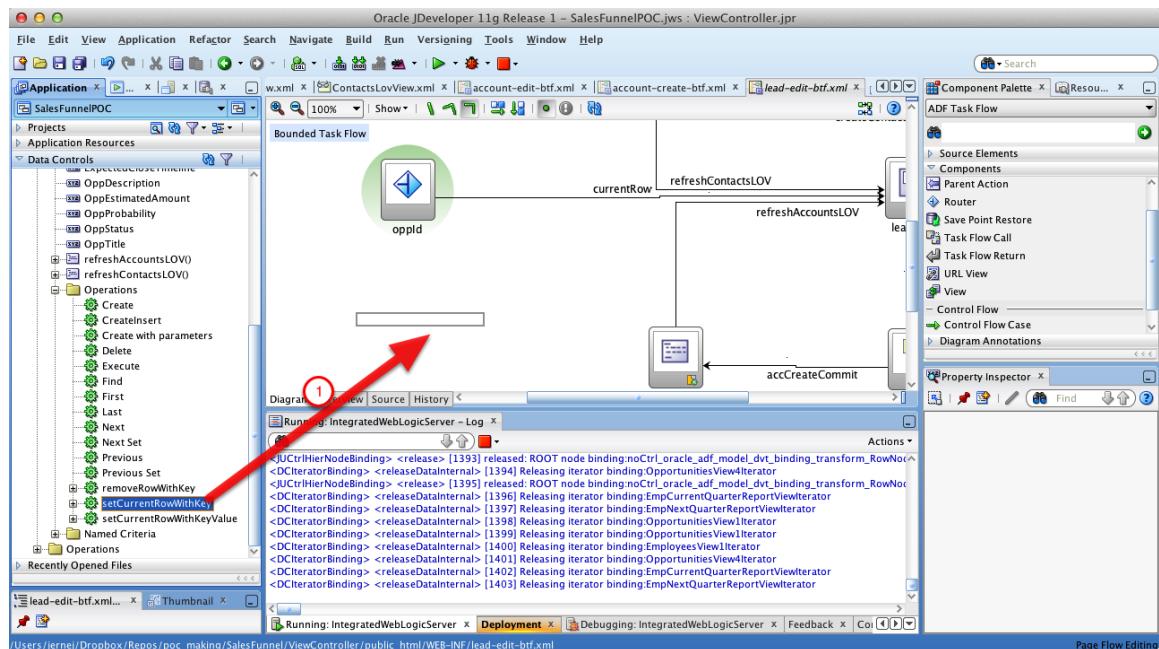
1. Drag and drop router to the page
2. Mark it as default activity
3. Set default outcome to currentRow
4. Connect the routes to leadEdit, the action should be named currentRow

Set Router default outcome



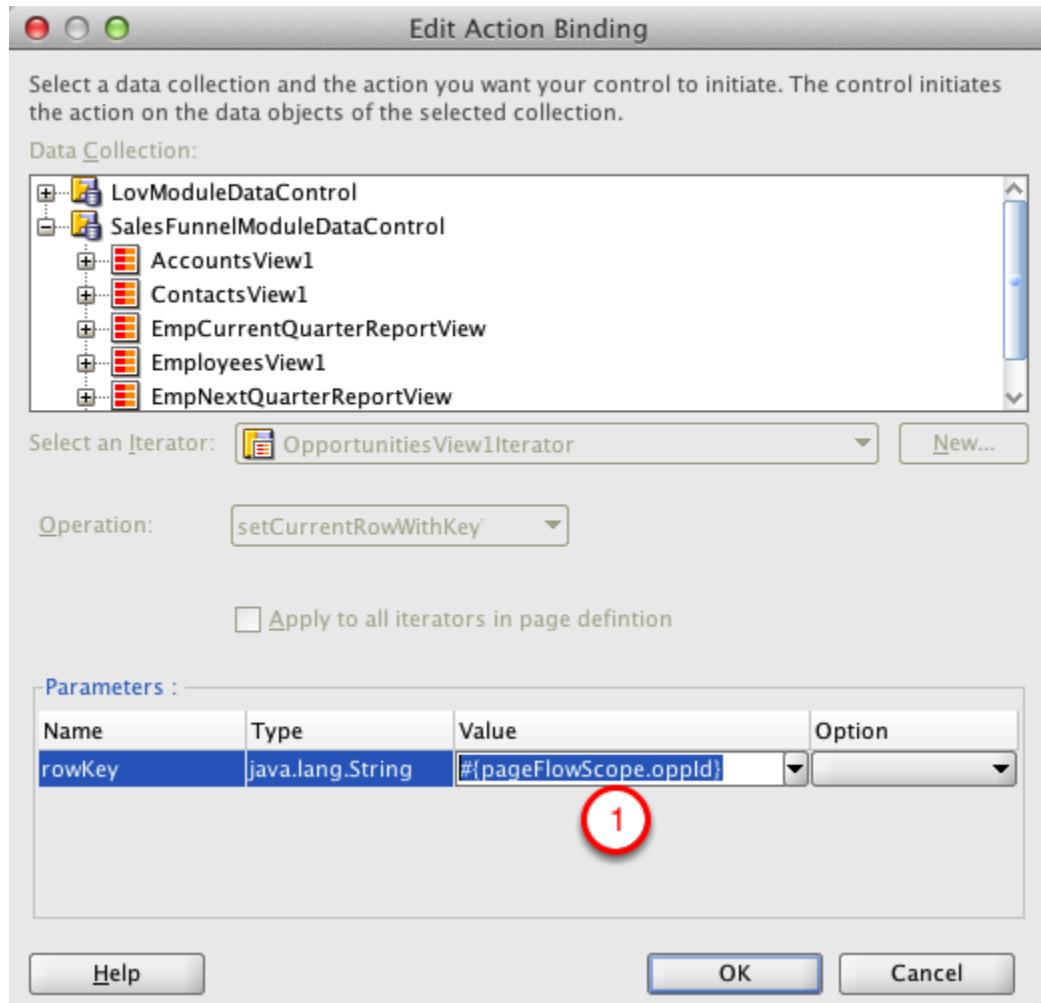
1. Click plus to create new case
2. Set Expression to `#{pageFlowScope.oppId != null}`
3. Set Outcome to `setRowByKeyValue`

Add `setCurrentRowWithKey` method



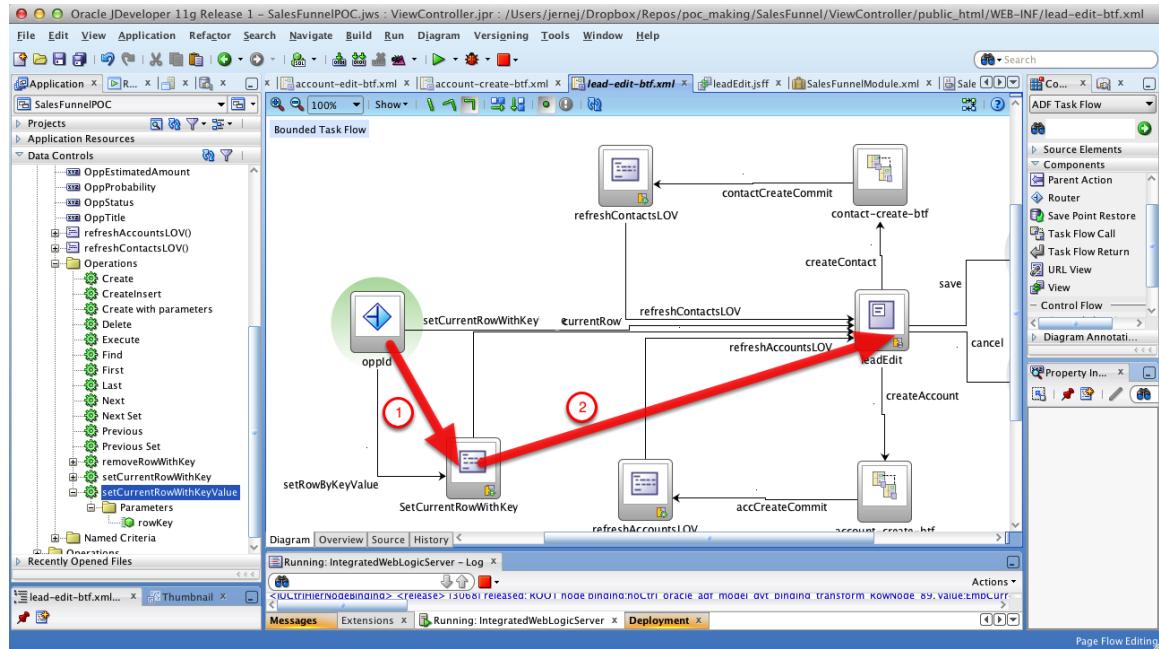
1. Expand OpportunitiesView1 (the one on the root level!), operations
2. Drag and drop `setCurrentRowWithKey` to the diagram

Edit Action Binding



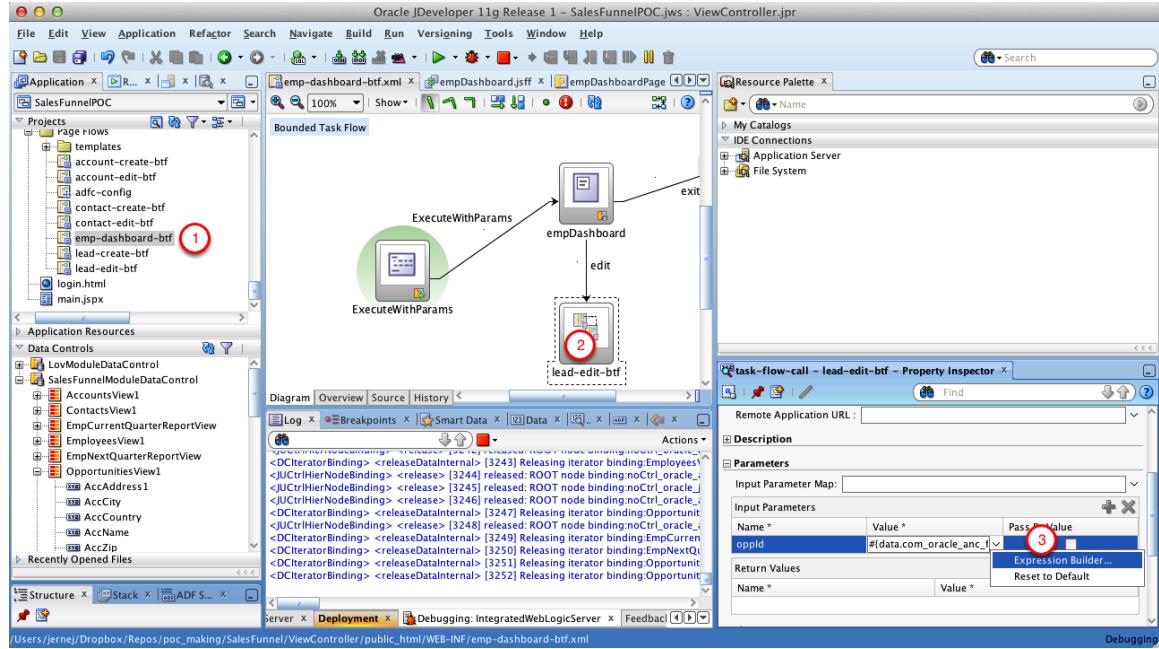
1. Set rowKey value to #{pageFlowScope.oppId}

Connect Router to SetCurrentRowWithKey



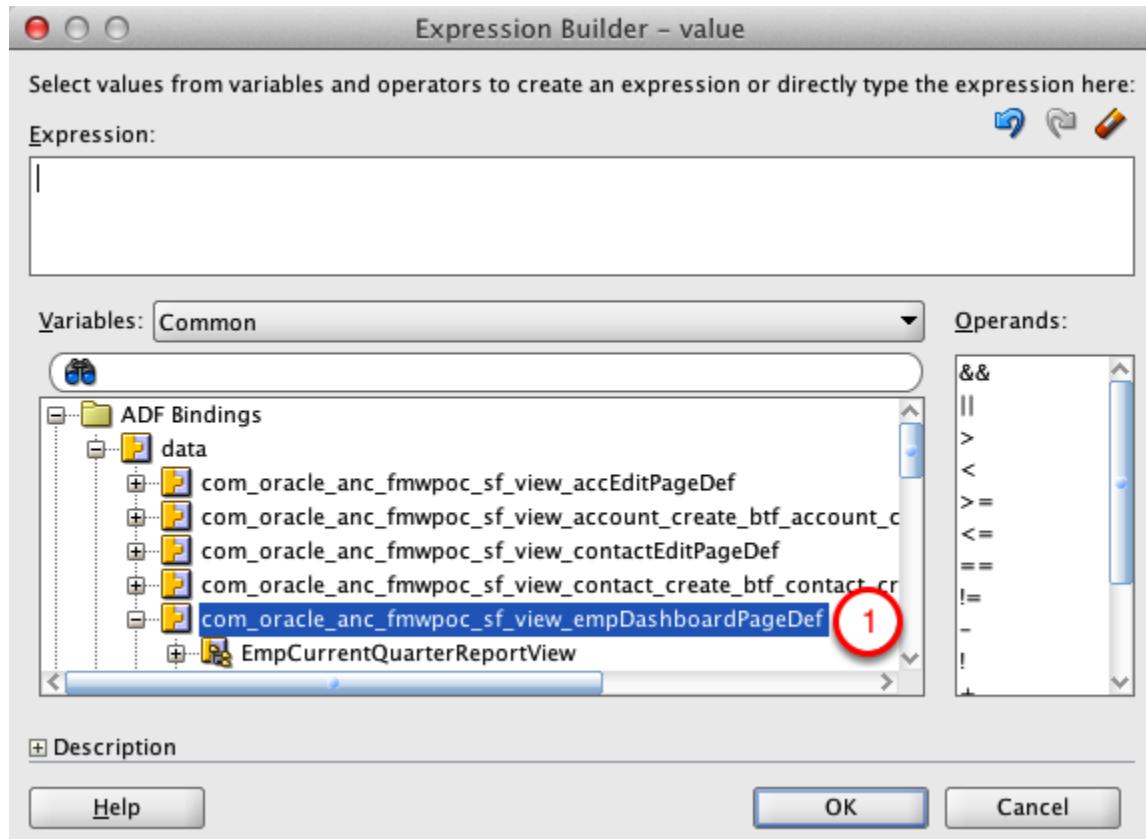
1. With Control Flow Case, connect oppId router to setCurrentRowWithKey and make sure from outcome is set to setRowByKeyValue
2. Connect SetCurrentRowWithKey to leadEdit

Set lead-edit-btf oppId Parameter Value



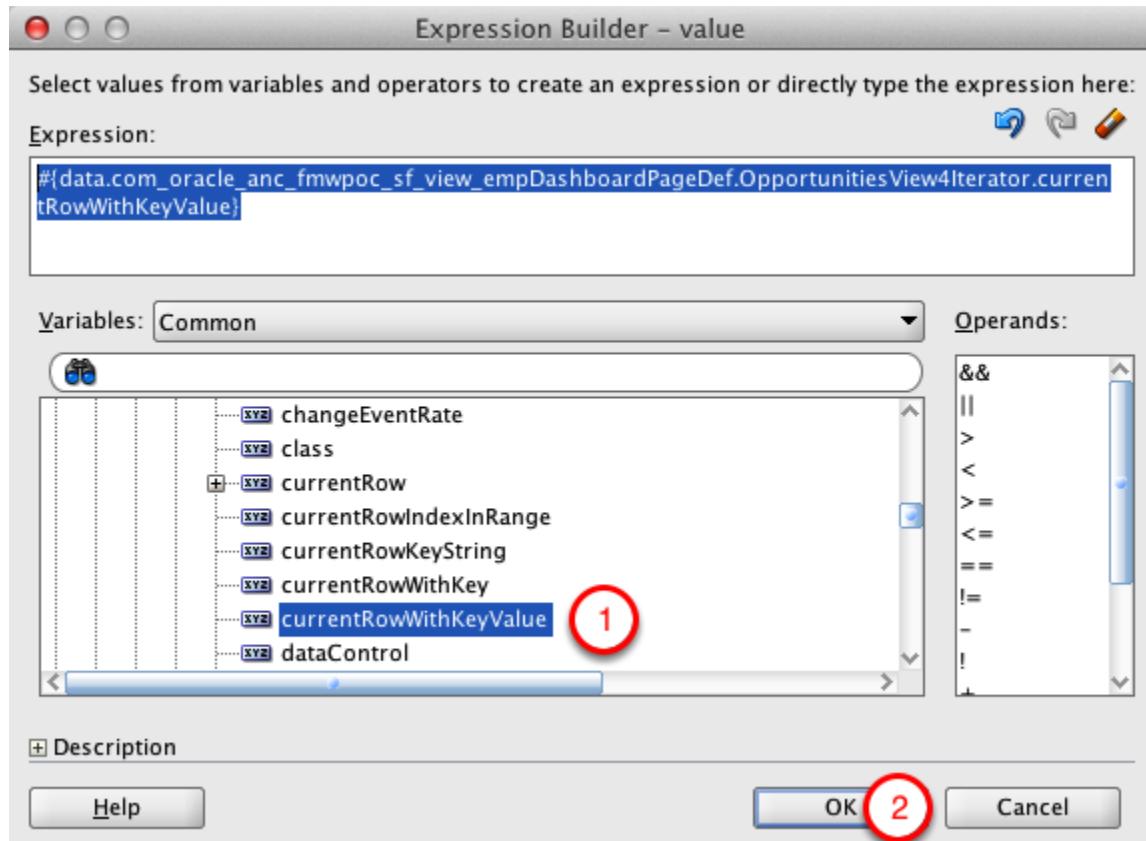
1. Open emp-dashboard-btf
2. Select lead-edit-btf
3. Click on the down arrow next to oppId param value and select Expression builder

Expression Builder - value



1. Expand ADF Binding -> data -> com_oracle_anc_fmw poc_sf_view_empDashboardPageDef -> OpportunitiesView4Iterator ->

Expression Builder - value

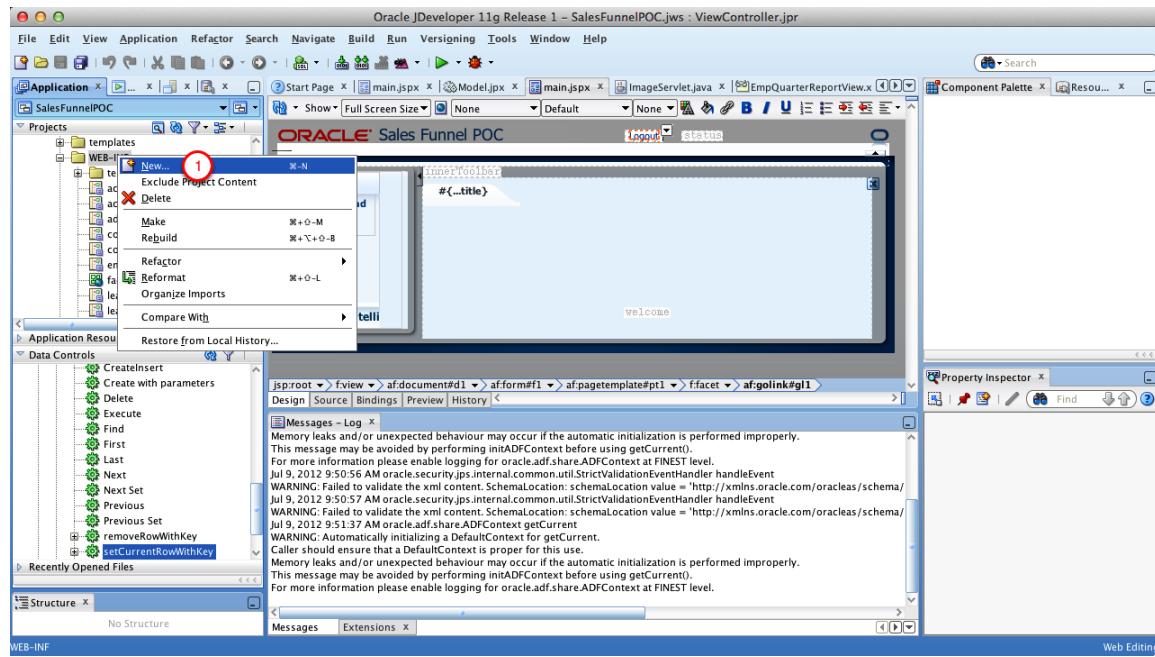


1. Select currentRowWithValue. The expression should be "#{data.com_anc_fmw poc_sf_view_empDashboardPageDef.OpportunitiesView4Iterator.currentRowWithValue}" without quotes
2. Click OK

V. Implementing Hierarchy Viewer

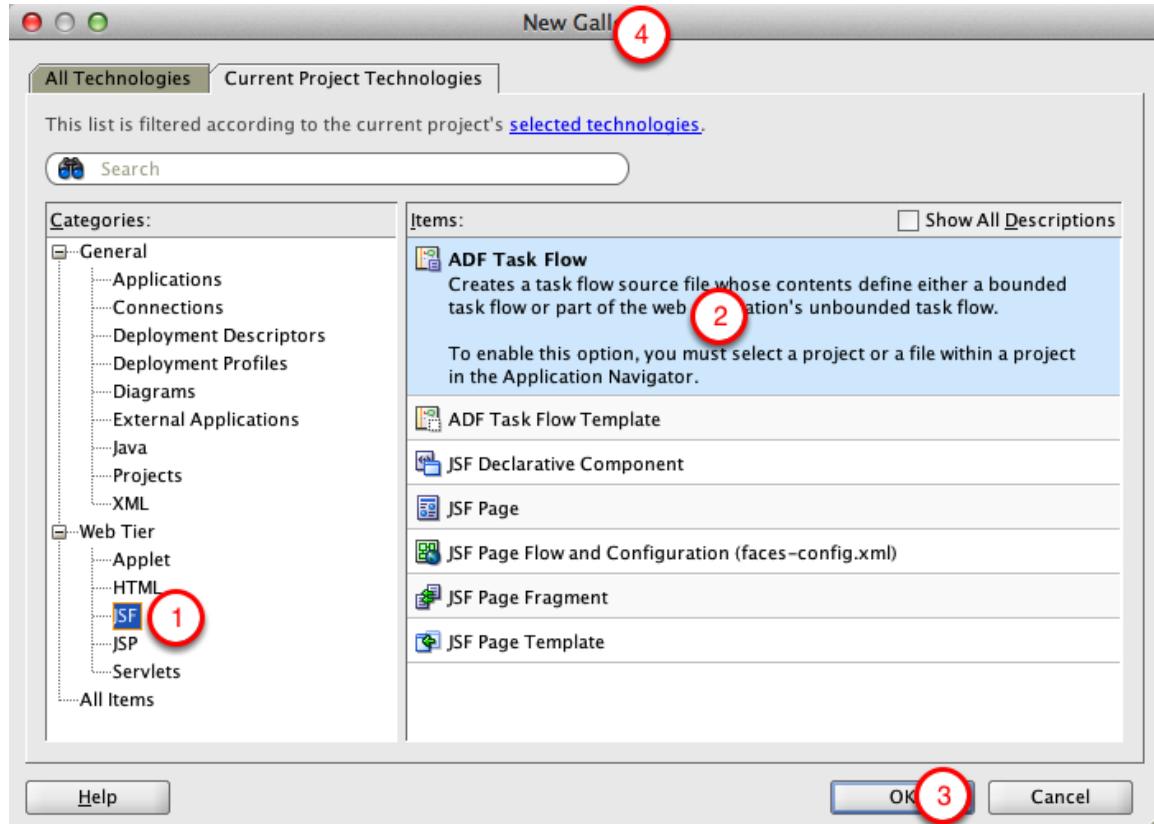
41. Hierarchy viewer

Create New Task Flow



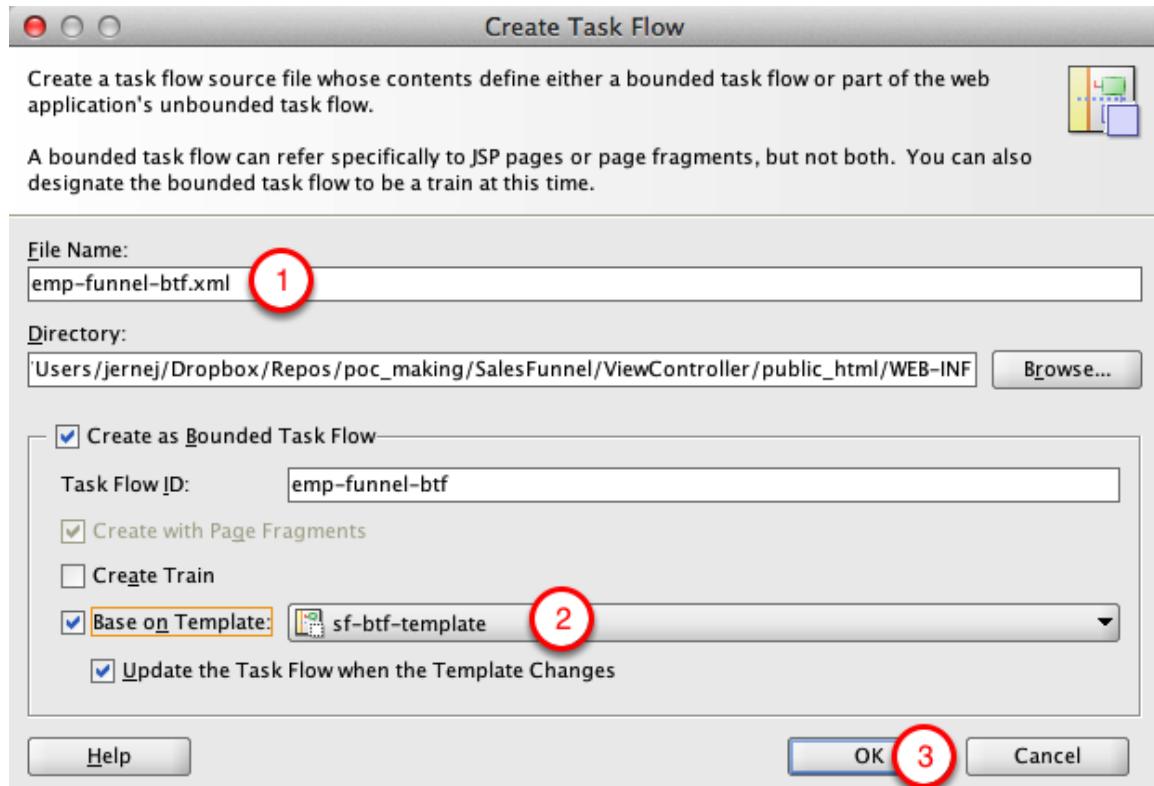
1. Right-click WEB-INF and select New

New Gallery



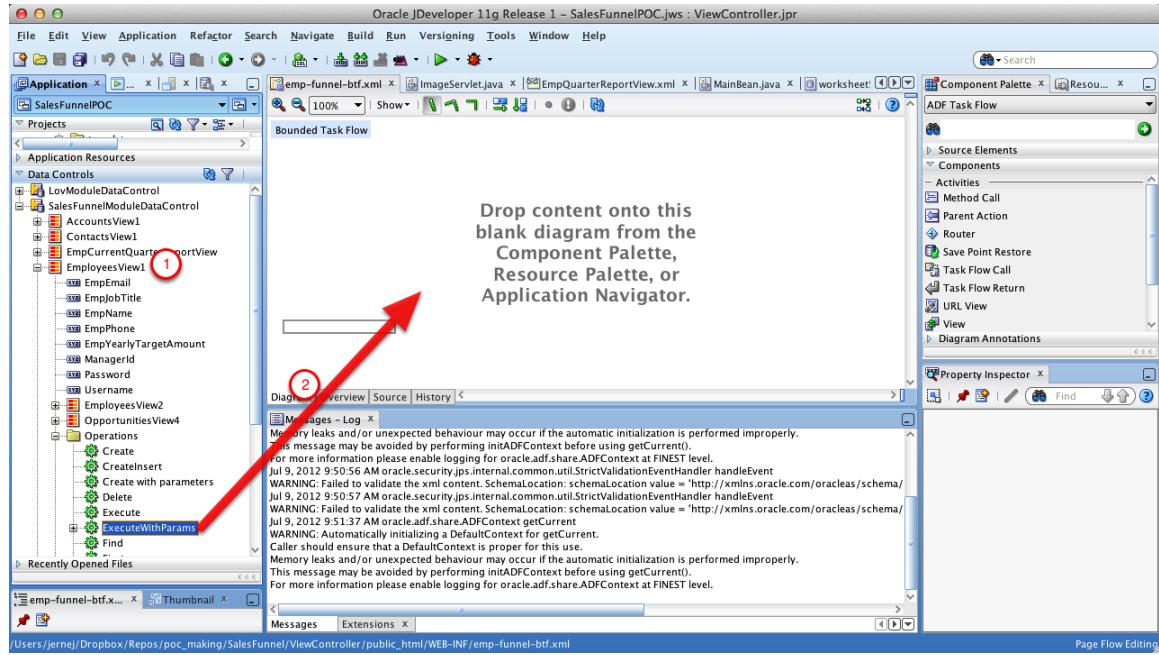
1. Select JSF from Web Tier
2. Select ADF Task Flow
3. Click OK

Create Task Flow



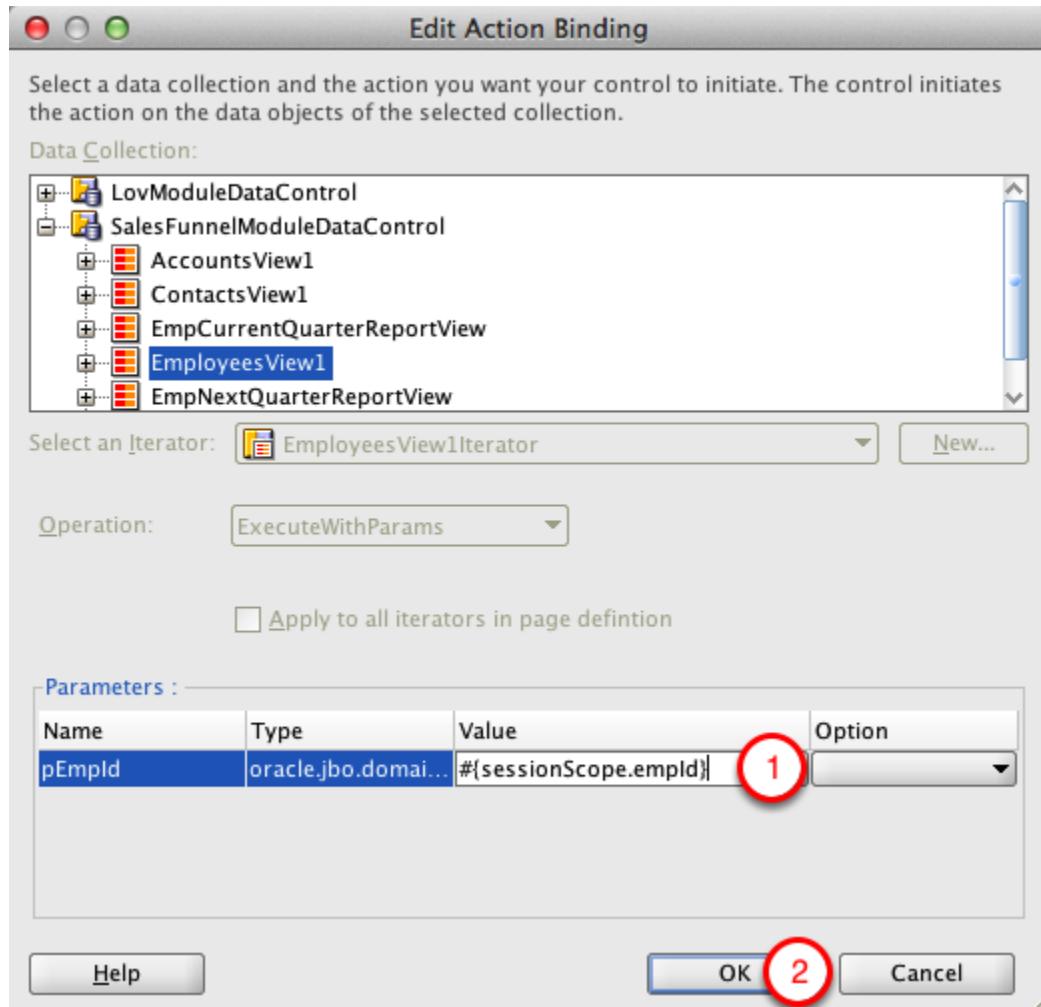
1. Name it `emp-funnel-btf`
2. Base it on `sf-btf-template`
3. Click OK

Add EmployeesView1 ExecuteWithParams to the diagram



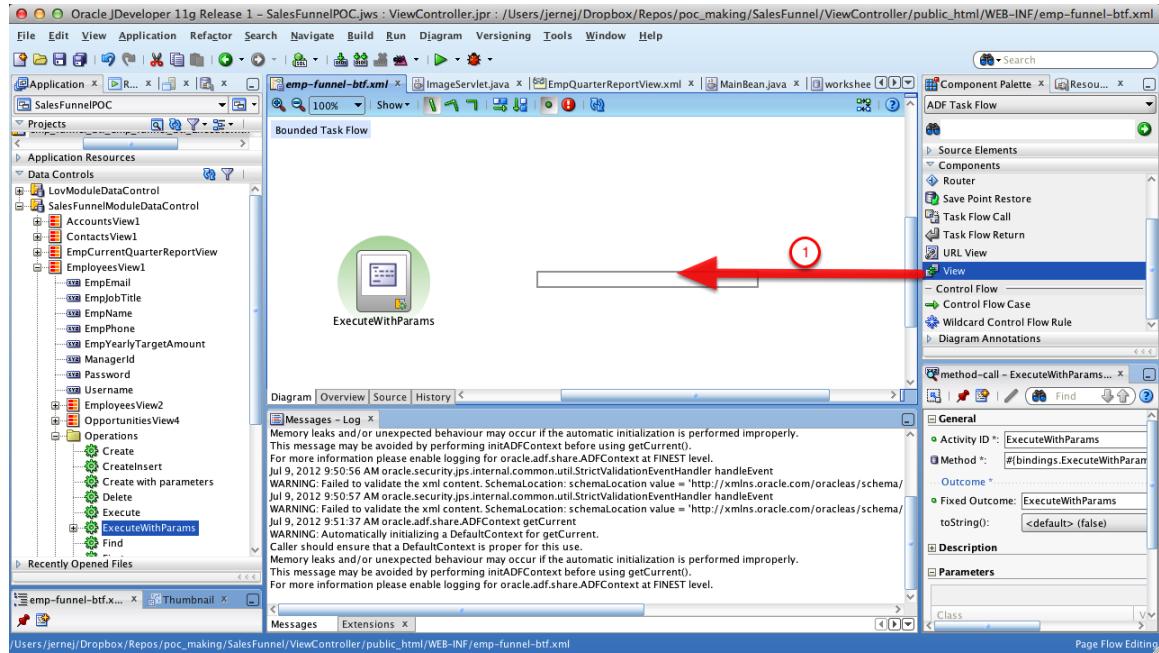
1. Expand EmployeesView1 -> Operations
2. Drag and drop ExecuteWithParams to the diagram

Edit Action Binding



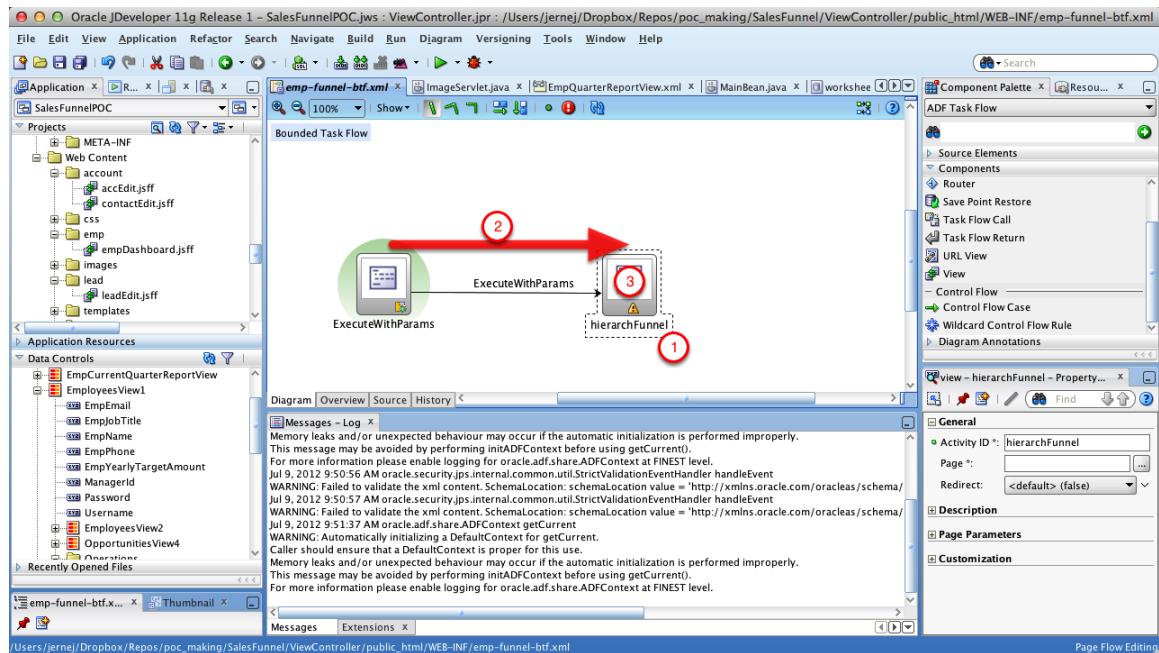
1. Set pEmpld value to #{sessionScope.empld}
2. Click OK

Add new View



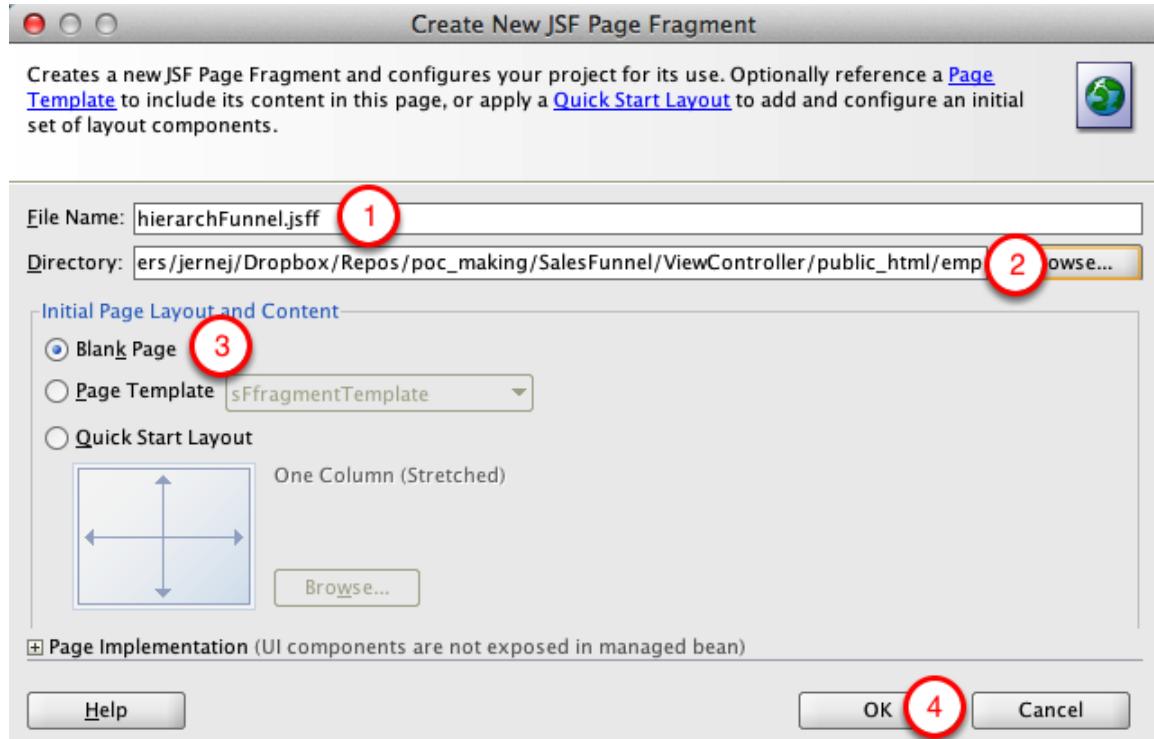
1. Drag and drop view activity to the diagram

Connect ExecuteWithParams to hierarchyFunnel



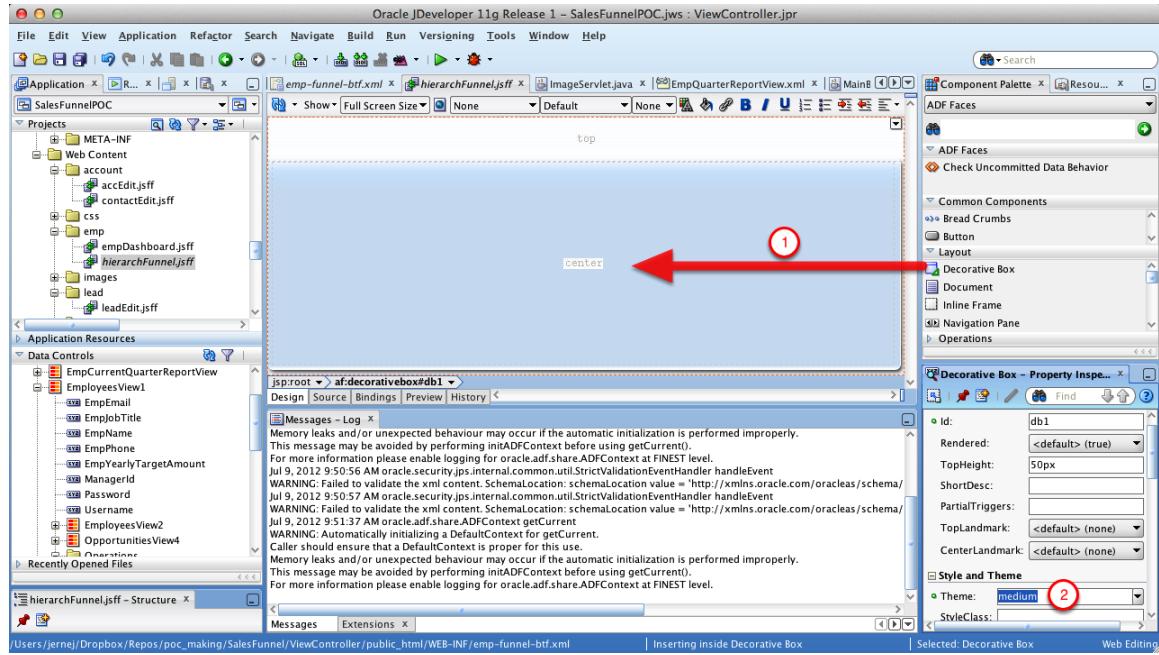
1. Name the view hierarchFunnel
2. Connect ExecuteWithParams to hierarchFunnel with a controlFlowCase
3. Double-click hierarchFunnel to open view dialog

Create New JSF Page Fragment



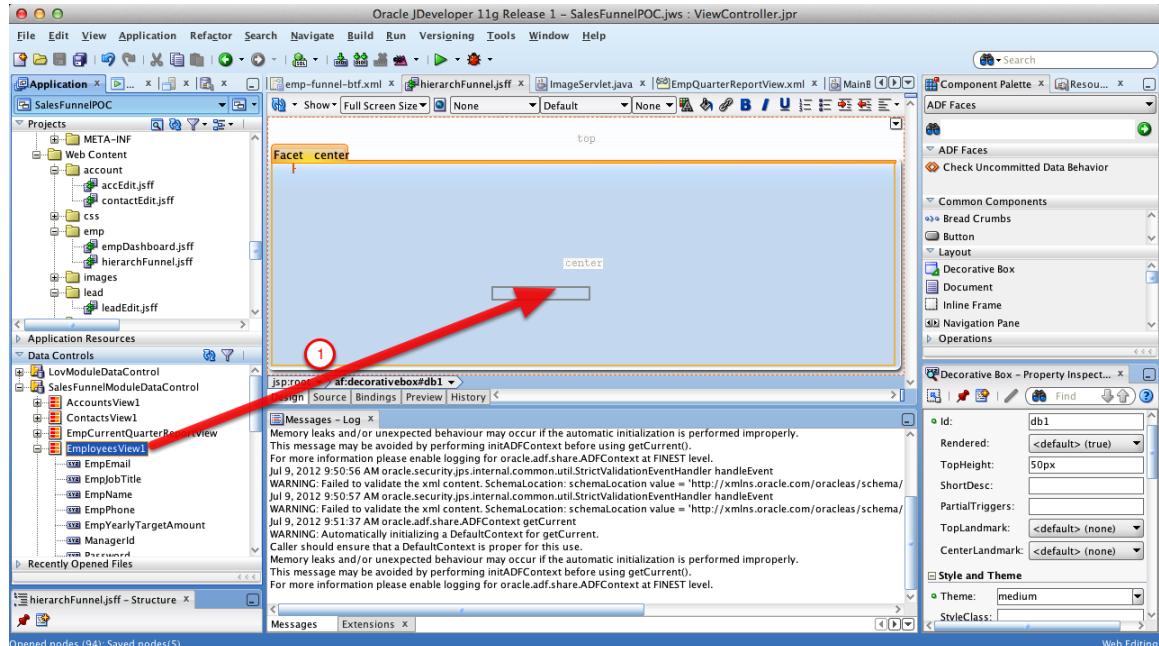
1. Name it hierarchFunnel.jsff
2. Append /emp to Directory
3. Start with a Blank Page
4. Click OK

Add DecorativeBox



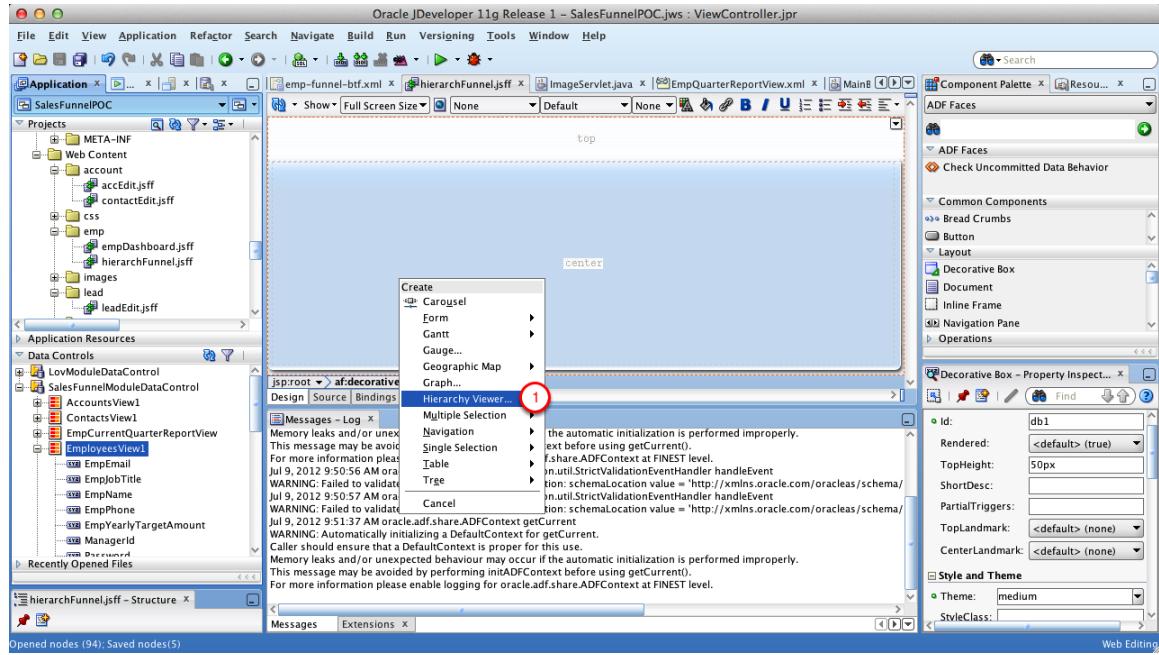
1. Drag and drop decorative box to the page
2. Set theme to medium

Drag And Drop EmployeesView1 on the form



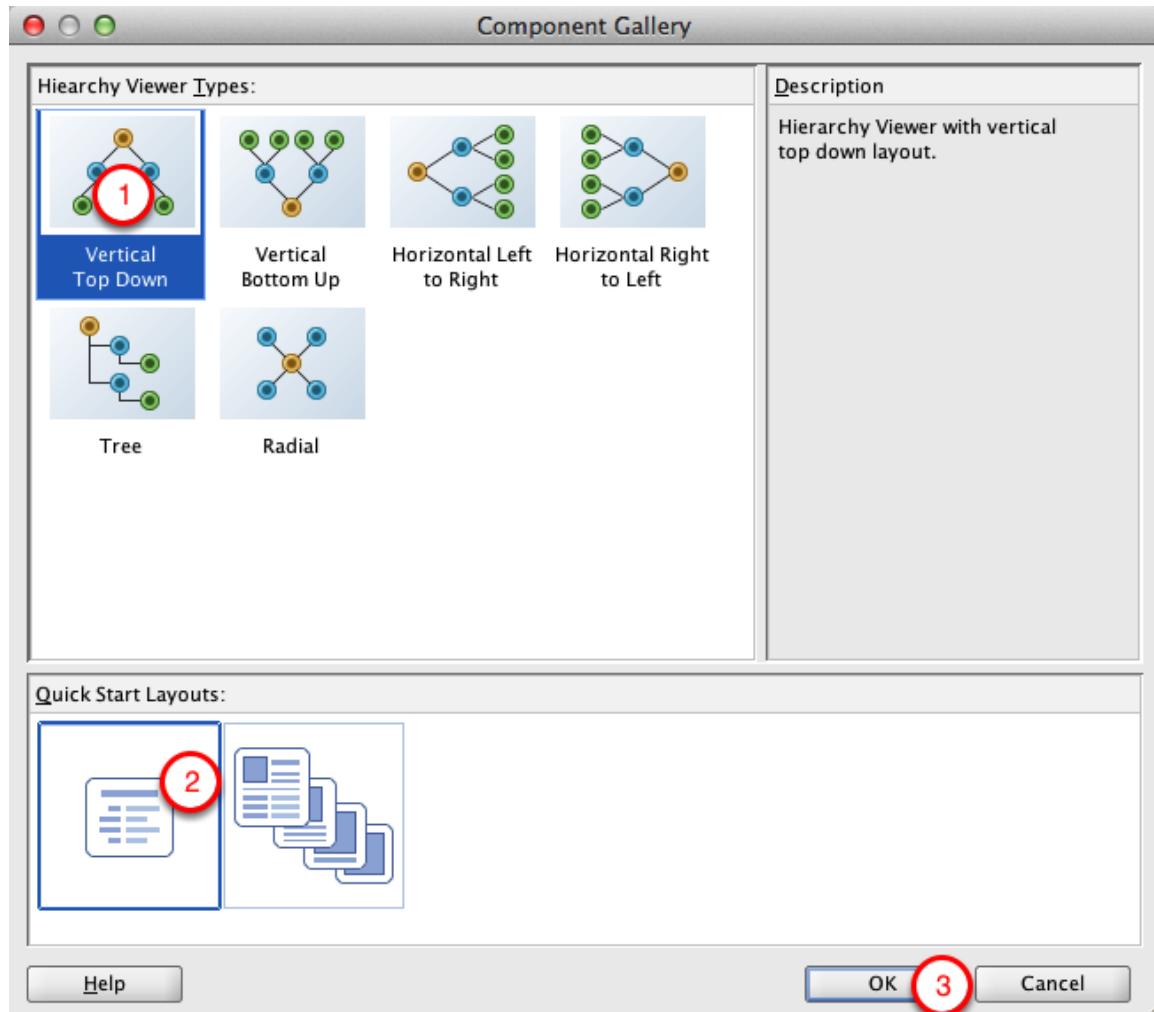
1. Drag and drop EmployeesView1 on the DecorativeBox

Create HierarchyViewer



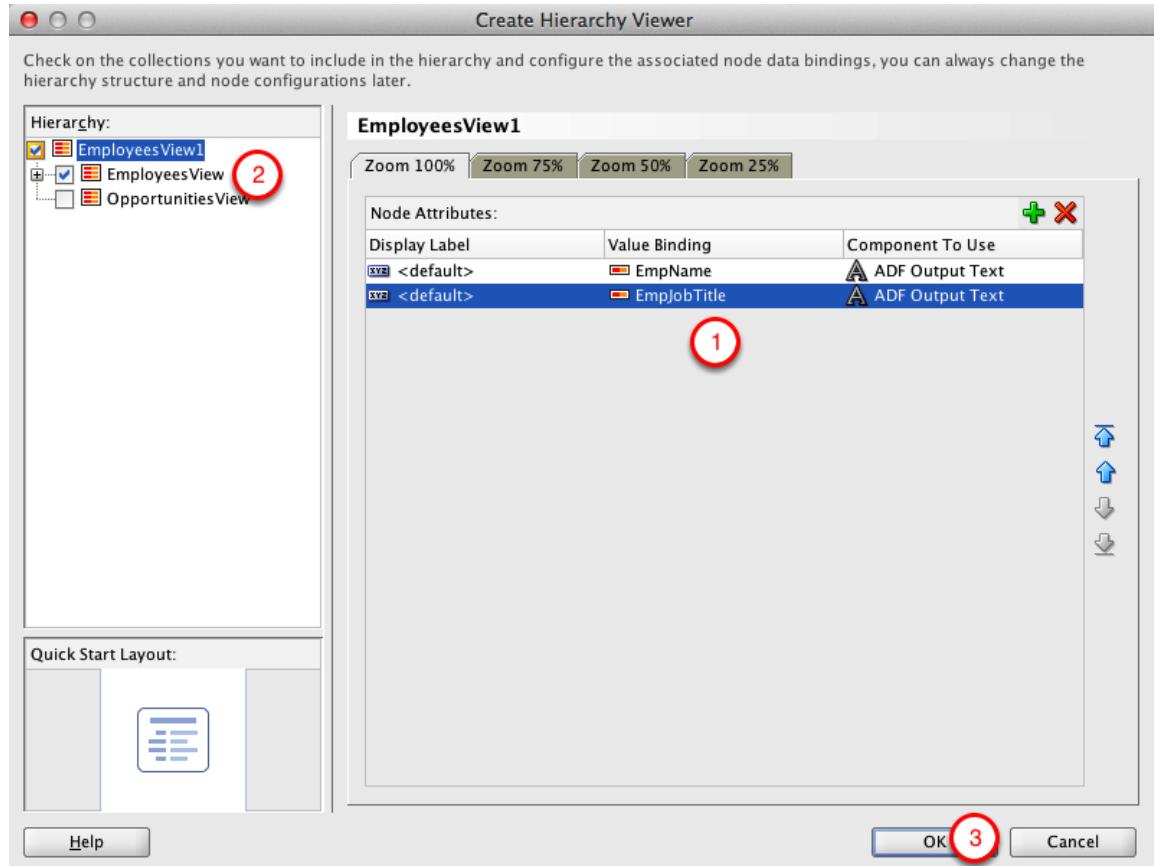
1. Select Hierarchy Viewer from the menu

Component Gallery



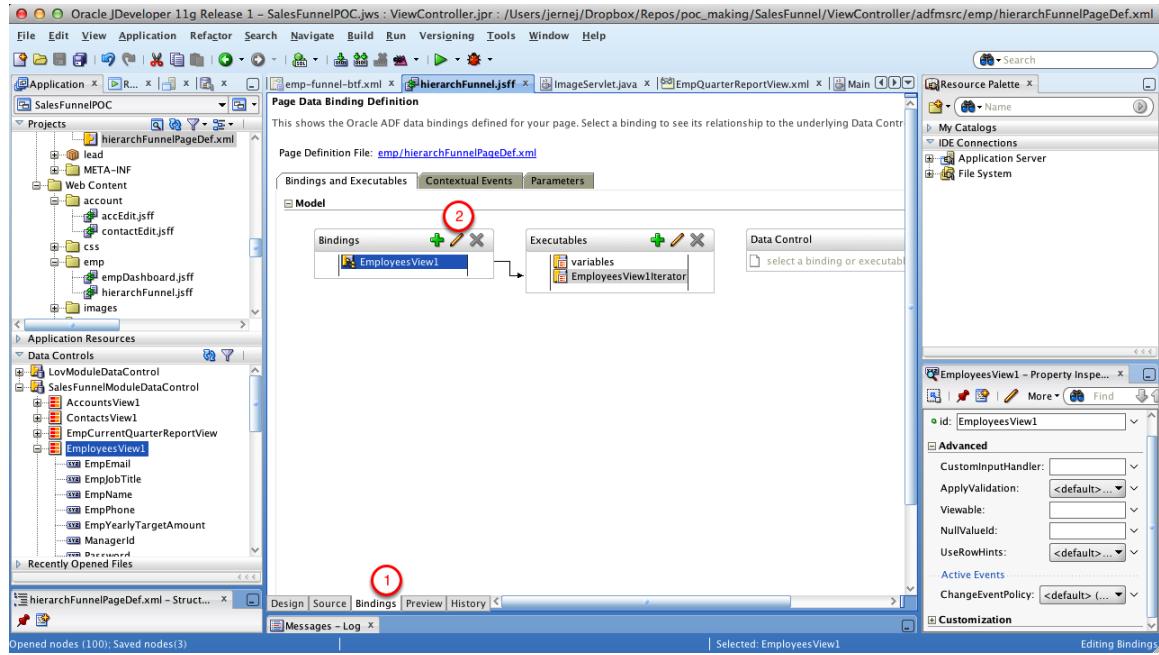
1. Select Vertical Top Down Type
2. Select the first layout
3. Click OK

Create Hierarchy Viewer



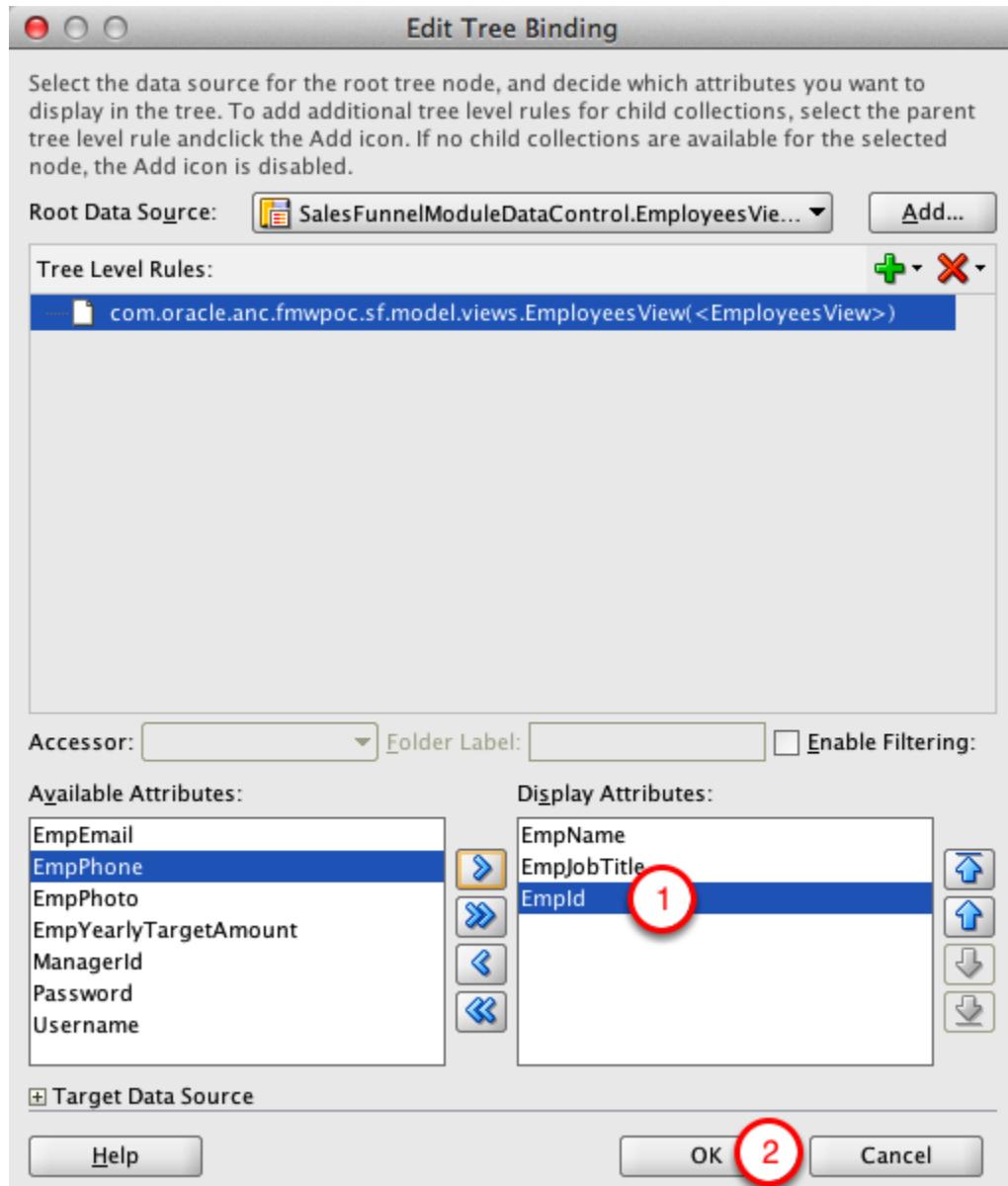
1. Using control buttons (up, down, delete, add) match the attributes the screenshot
2. When done, check the combo box next to EmployeesView in the hierarchy
3. Click OK

Edit Bindings



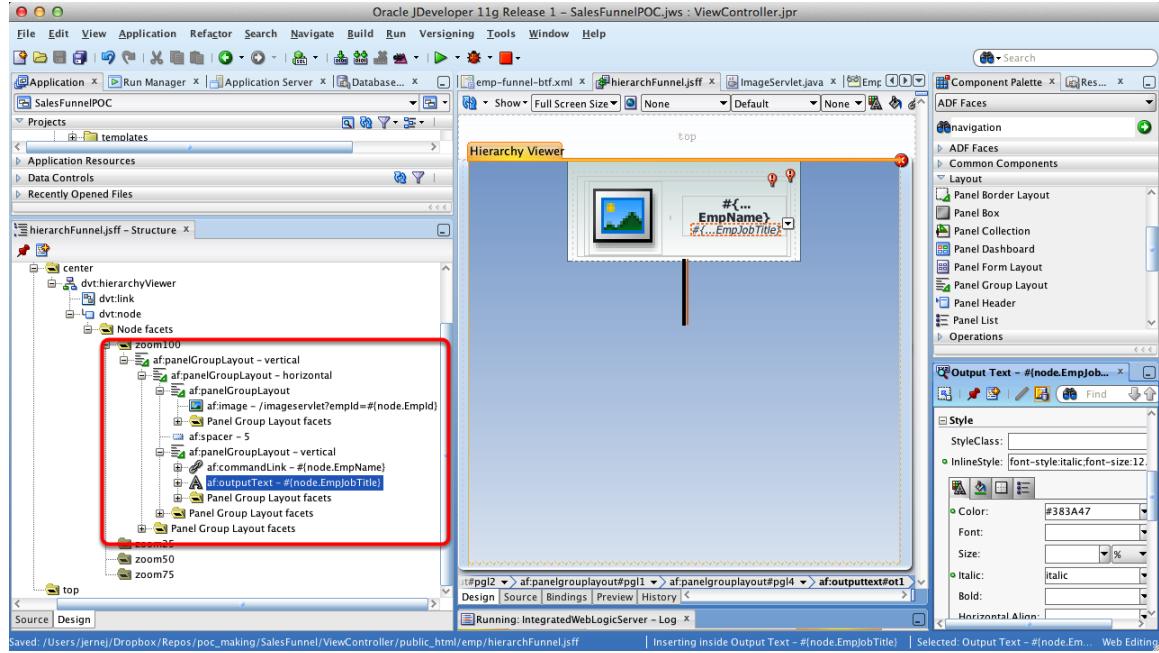
1. Open Bindings tab
2. With EmployeesView1 selected, click ok the edit icon

Edit Tree Binding



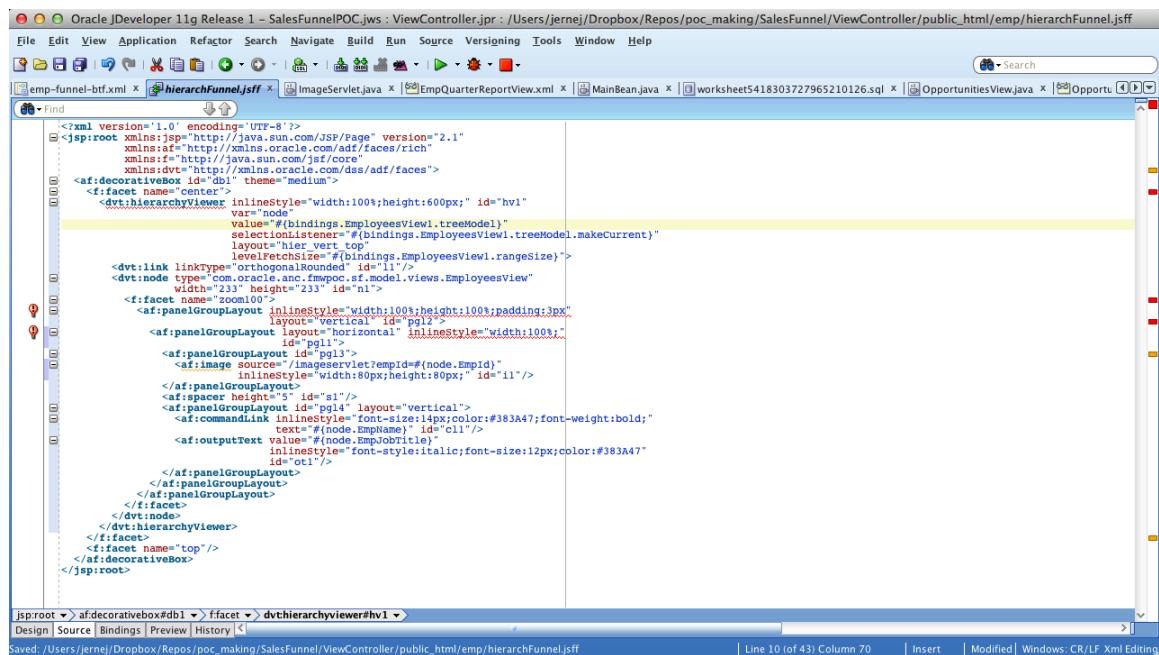
1. Slide empld to the right
2. Click OK

Exercise



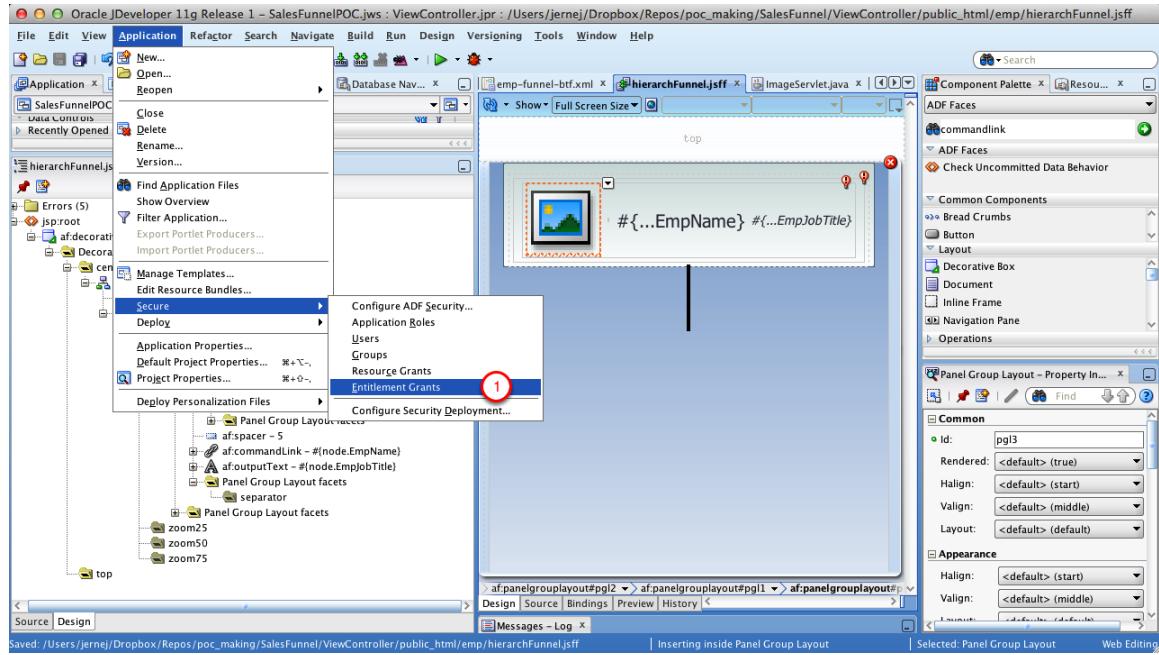
Create the layout of components matching the one on the screenshot

Exercise, part 2



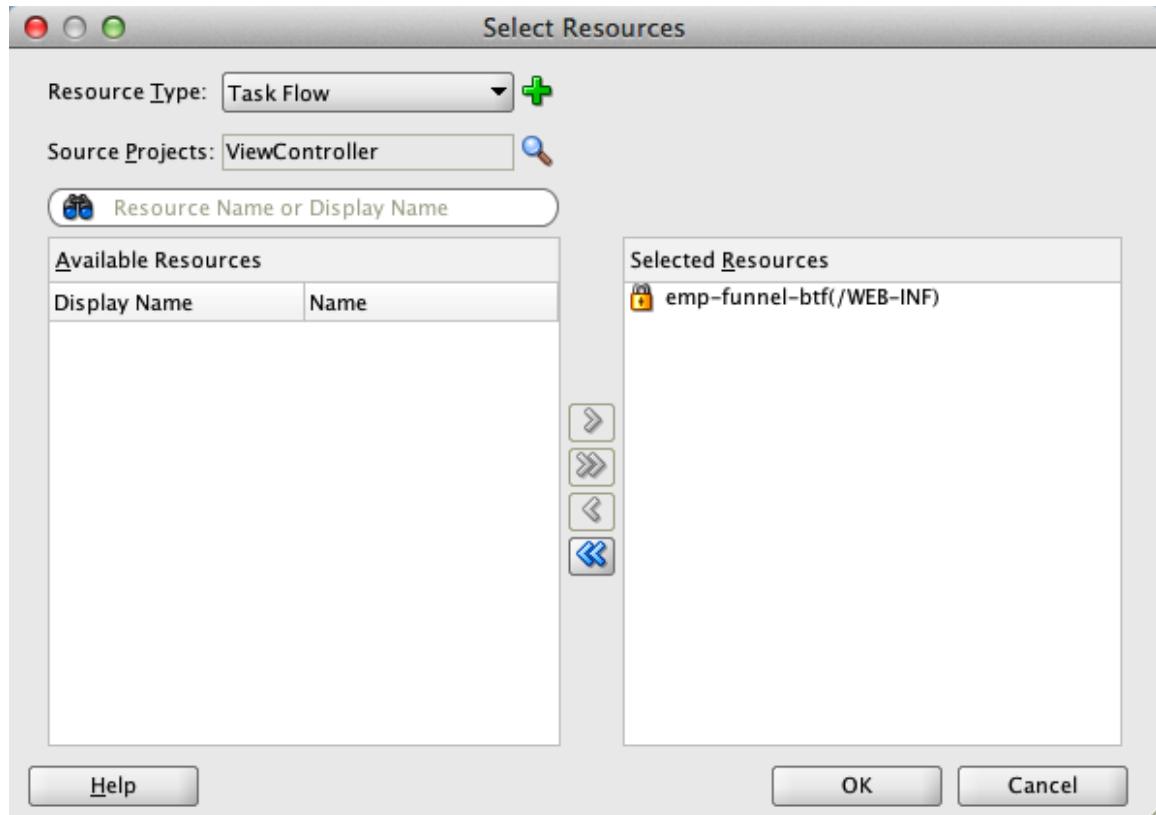
Use also the screenshot below to understand the structure, and set components properties

Configure Grants and Security



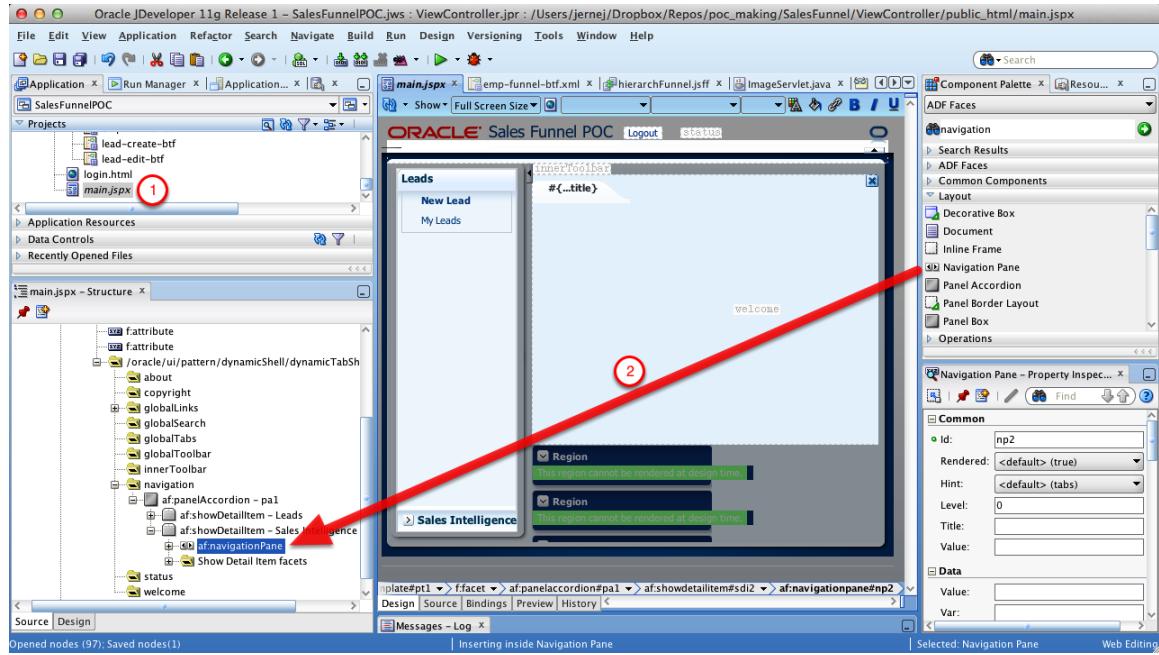
- From the menu select Application > Secure > Entitlement Grants

Select Resources



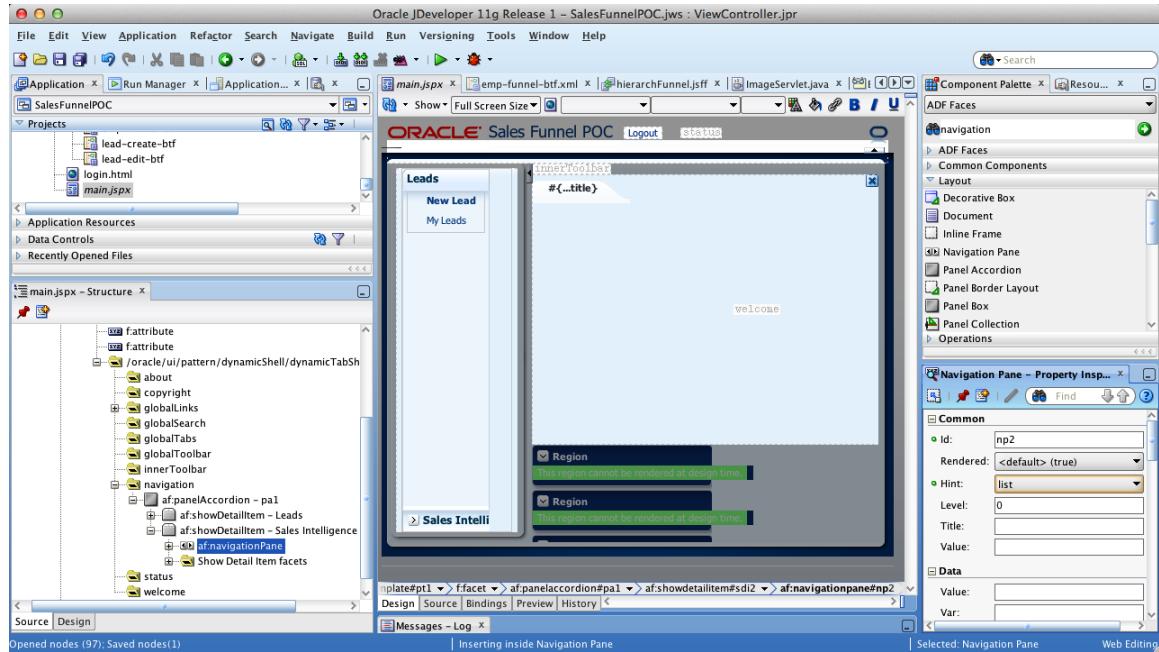
Add emp-funnel-btf to the emp entitlement.

Add NavigationPane to the main Form



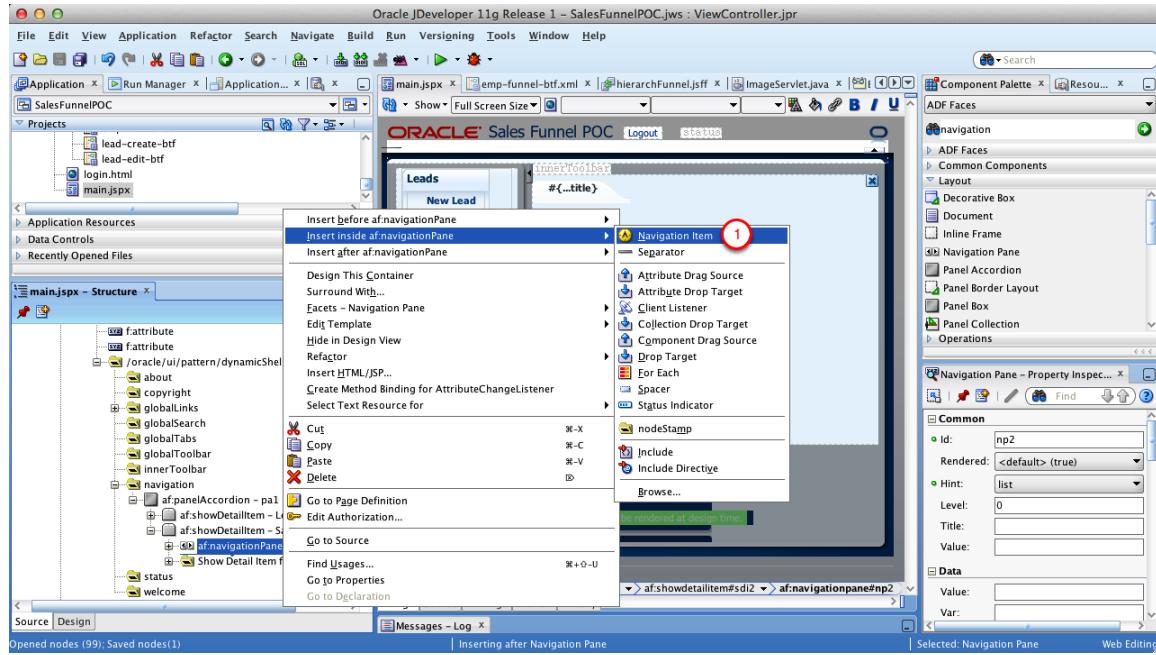
1. Open main.jspx
2. Drag and drop Navigation Pane to Sales Intelligence showDetailItem

Set Navigation Pane Properties



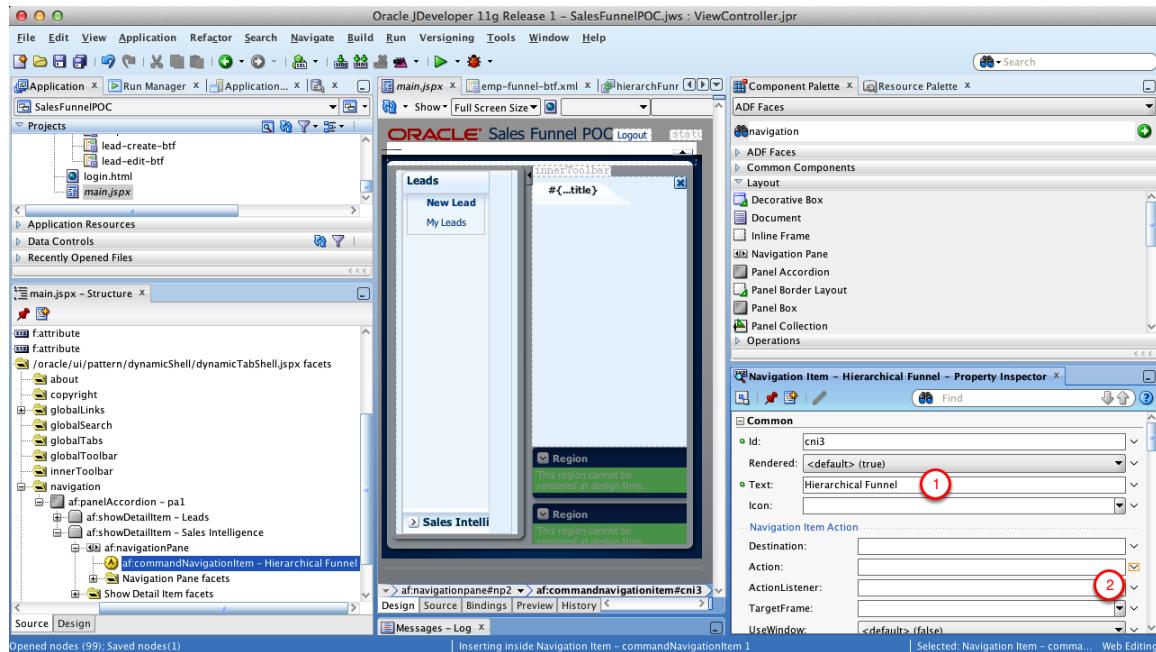
1. Set navigation Pane's hint property to list

Add Navigation Item



1. Right click the navigation pane and select Insert inside > Navigation Item from the menu

Set Navigation Item Properties



1. Set Text to Hierarchical Funnel
2. Click the down arrow next to Action Listener property and select edit from the menu

Edit Property: ActionListener



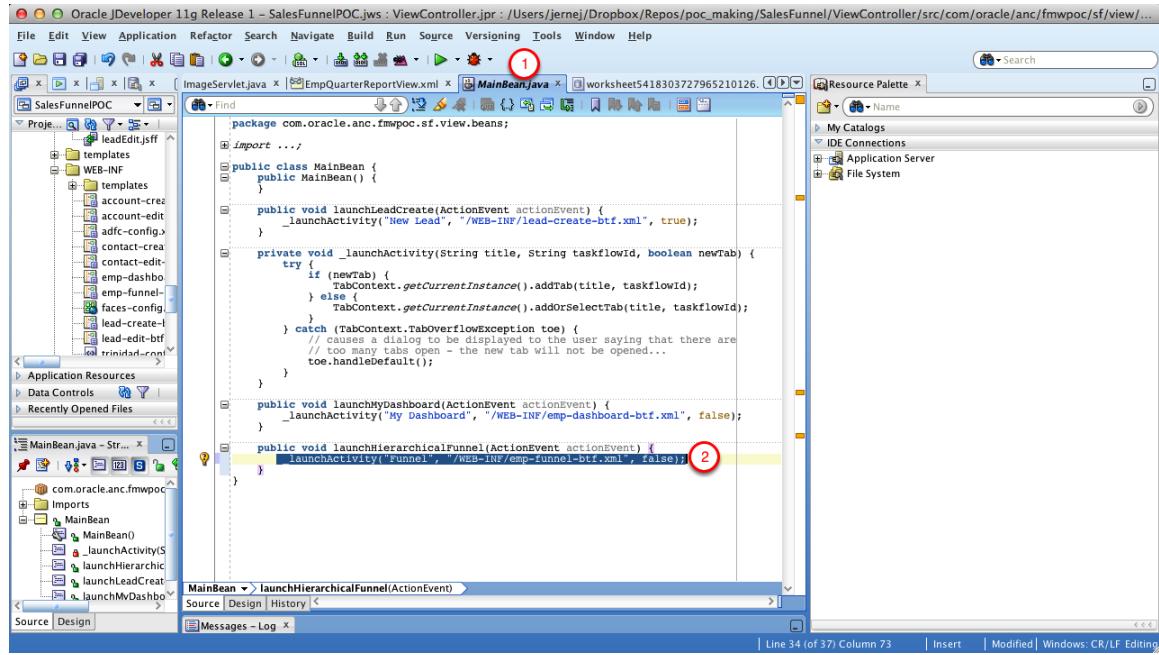
1. Select mainBean for Managed Bean
2. Click New next to Method

Create Method



1. Set name to launchHierarchicalFunnel
2. Click ok
3. Click OK to close the ActionListener Edit Property dialog

Implement launchHierarchicalFunnel

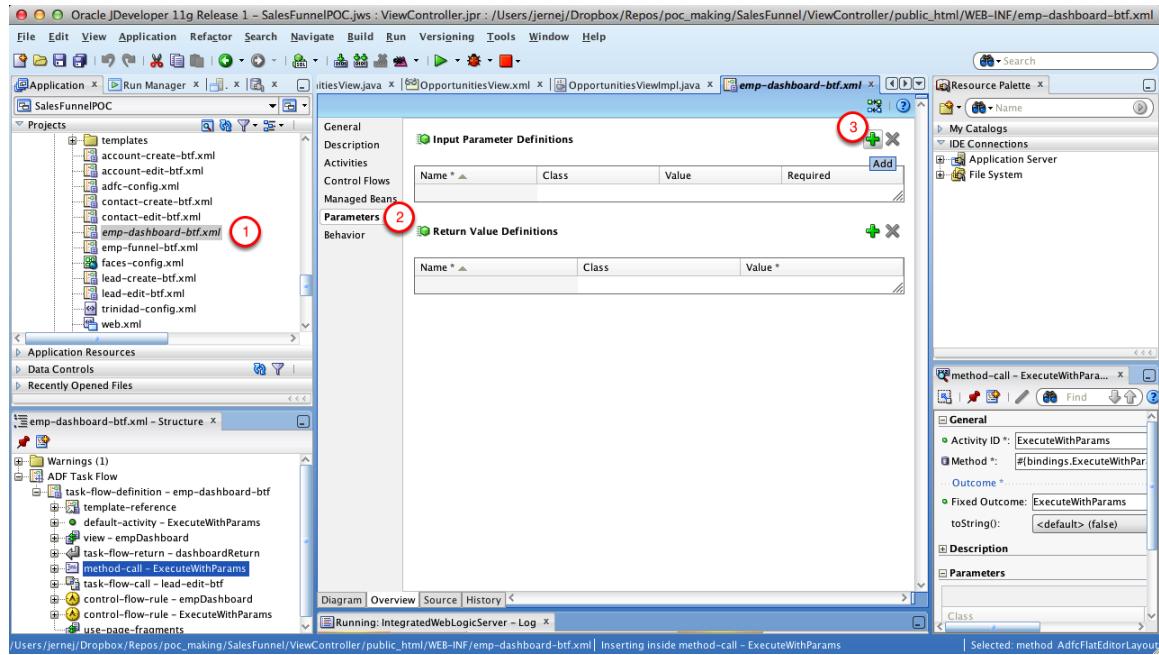


1. Open MainBean.java
2. Paste the method implementation

```
_launchActivity("Funnel", "/WEB-INF/emp-funnel-btf.xml", false);
```

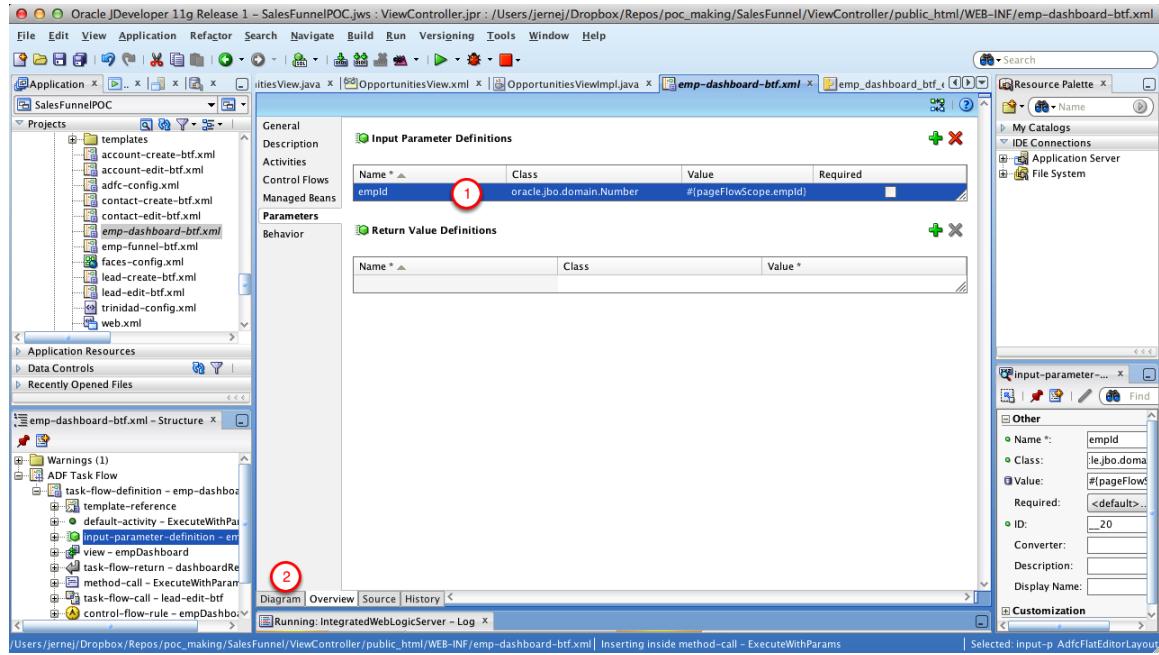
42. Connecting hierarchy viewer to the dashboard

Add Parameter to emp-dashboard-btf



1. Open emp-dashboard-btf
2. Open Parameters tab
3. Click the plus icon to add a parameter

Configure empld Parameter

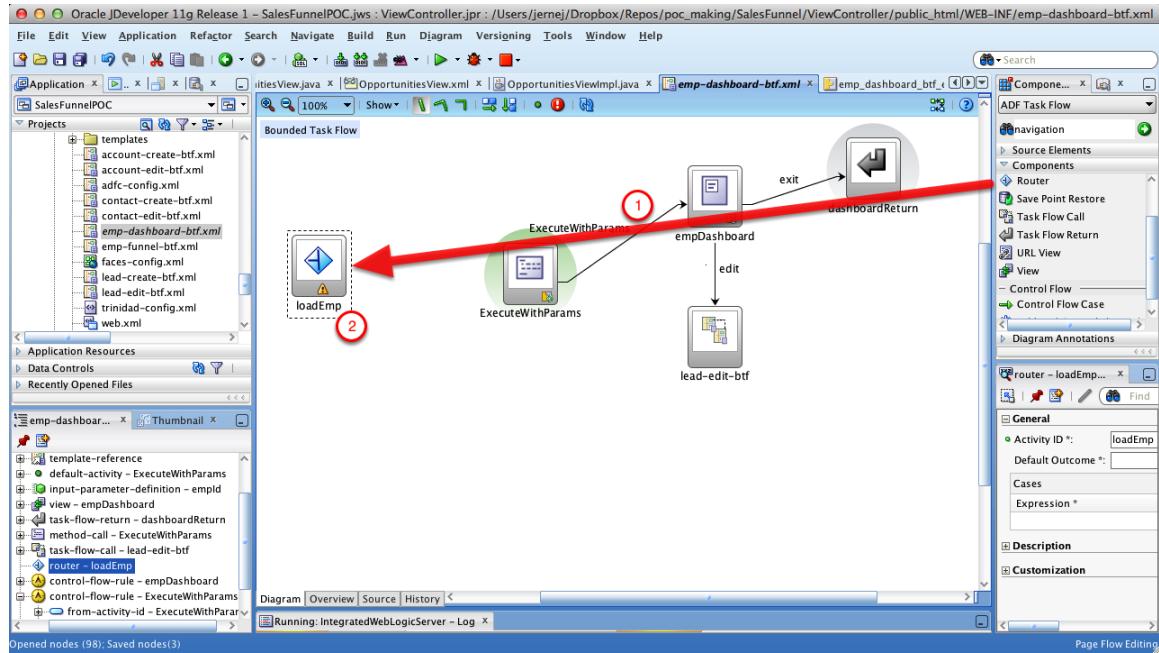


1. Set properties:

- Name: empld
- Class: oracle.jbo.domain.Number
- Value will automatically be set to #{pageFlowScope.empld}
- Leave Required unchecked

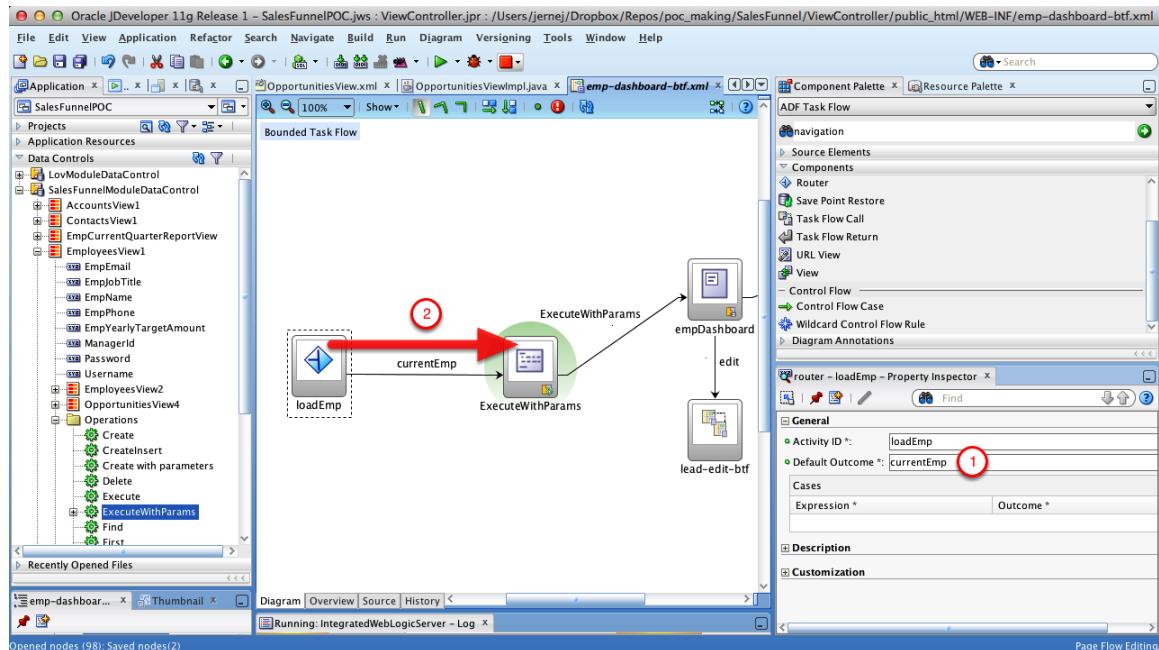
2. Open Diagram

Add Router



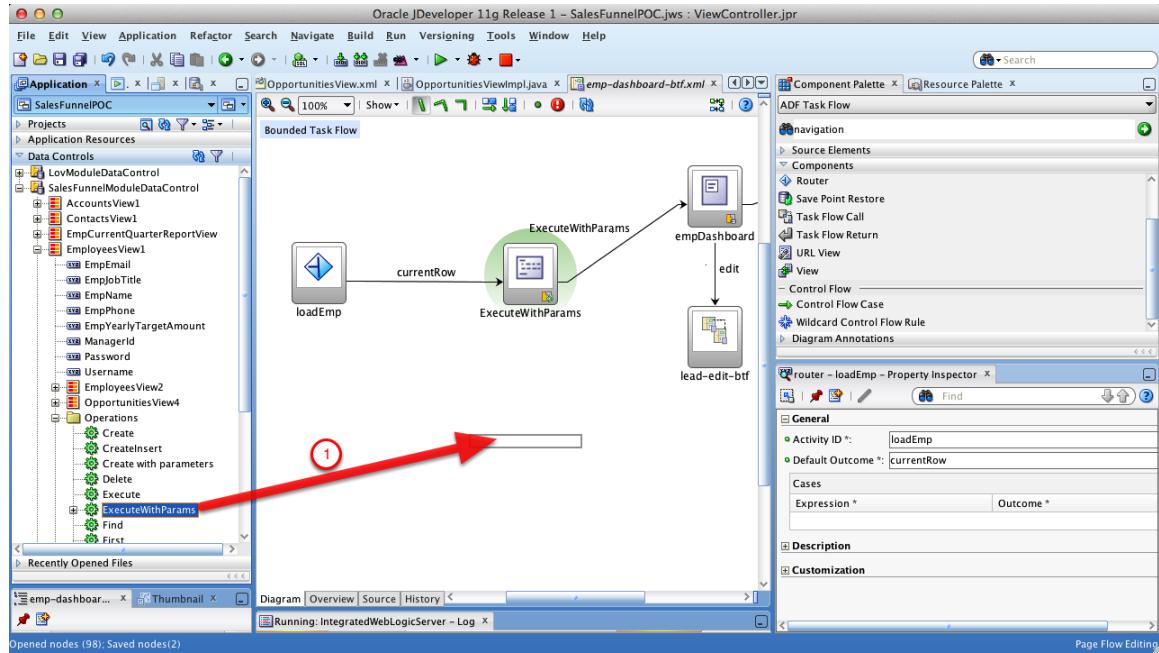
1. Drag and drop Router component to the page
2. Name it loadEmp

Add Default Outcome



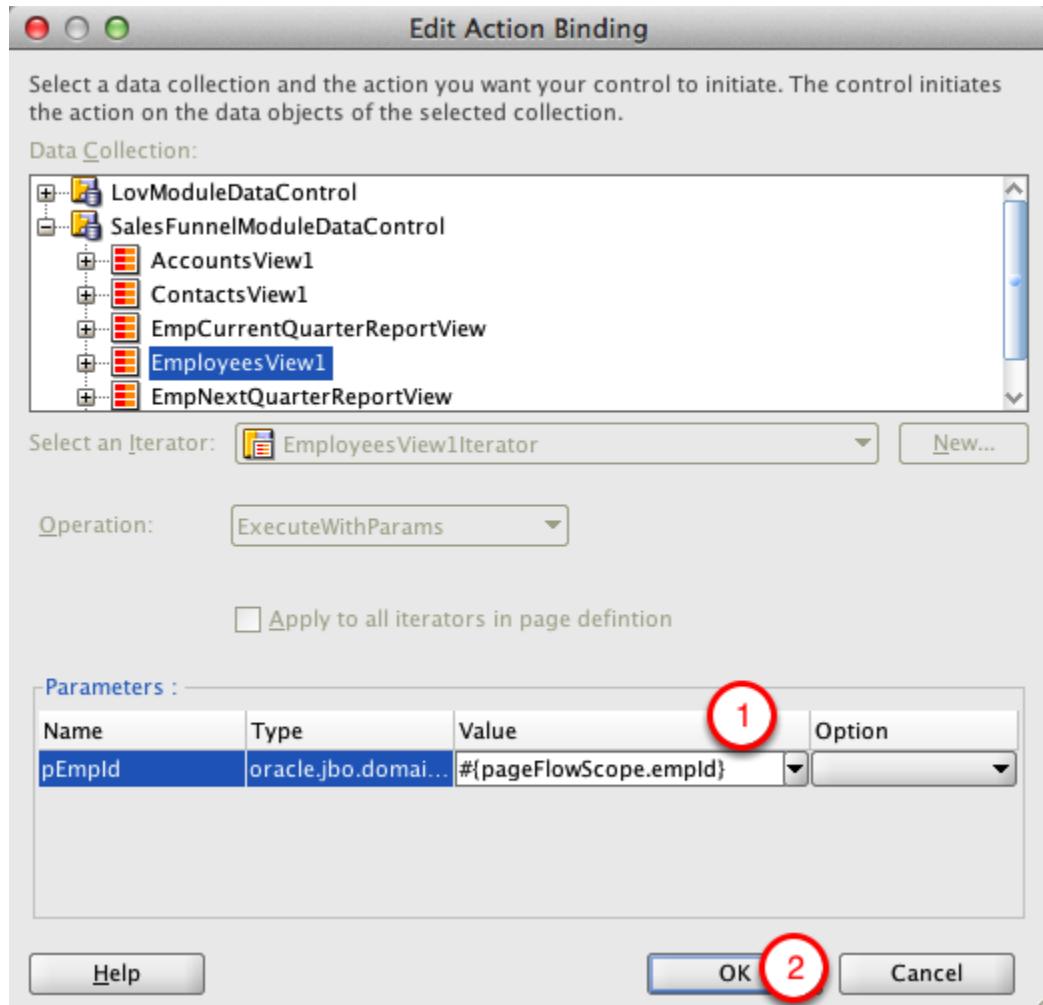
1. Set Default Outcome to currentEmp
2. Connect loadEmp to ExecuteWithParams, the action name should be currentEmp

Add EmployeesView1 ExecuteWithParams



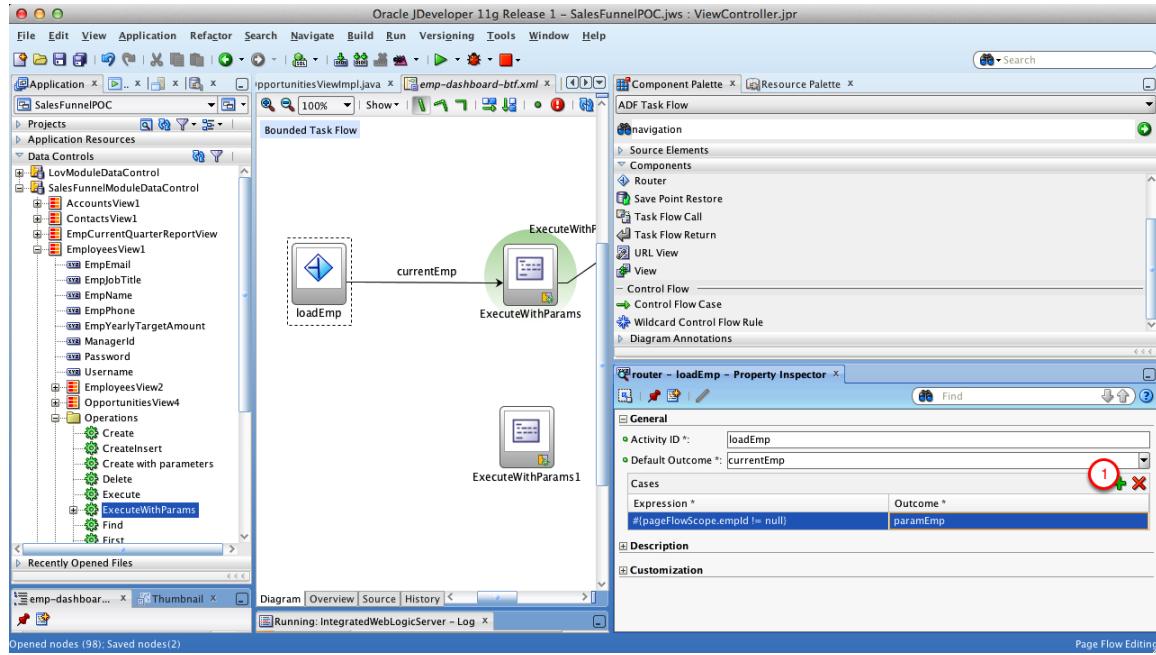
1. Drag and drop EmployeesView1 -> Operations -> ExecuteWithParams to the diagram

Edit Action Binding



1. Set Value to #{pageFlowScope.empld}
2. Click OK

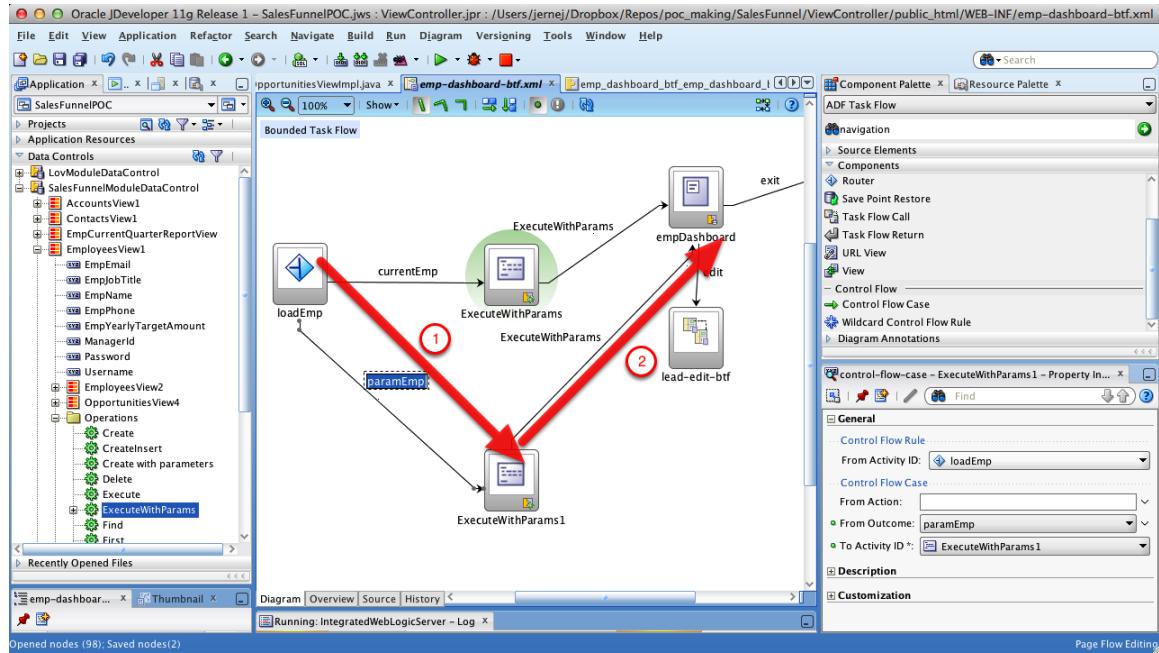
Add Router Outcome



1. Add new case by clicking plus icon
2. Set properties:

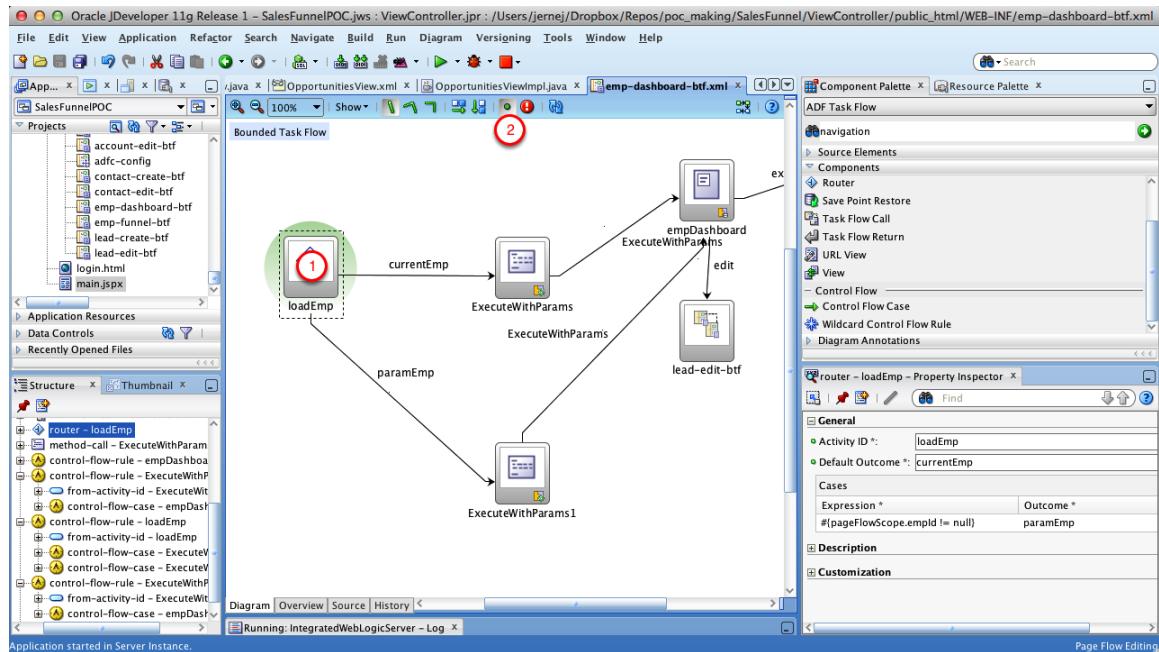
- Expression: #{pageFlowScope.empId != null}
- Outcome: paramEmp

Connect Activities



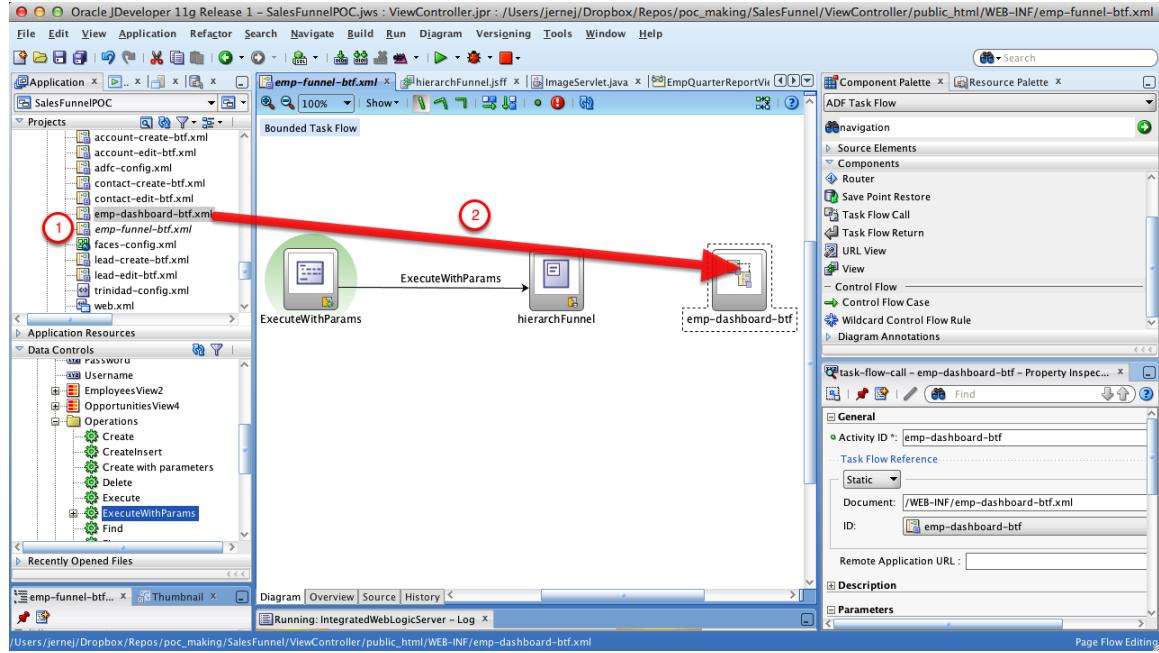
1. Connect loadEmp to ExecuteWithParams1, make sure from outcome is paramEmp
2. Connect ExecuteWithParams1 to empDashboard

Set Router As Default Activity



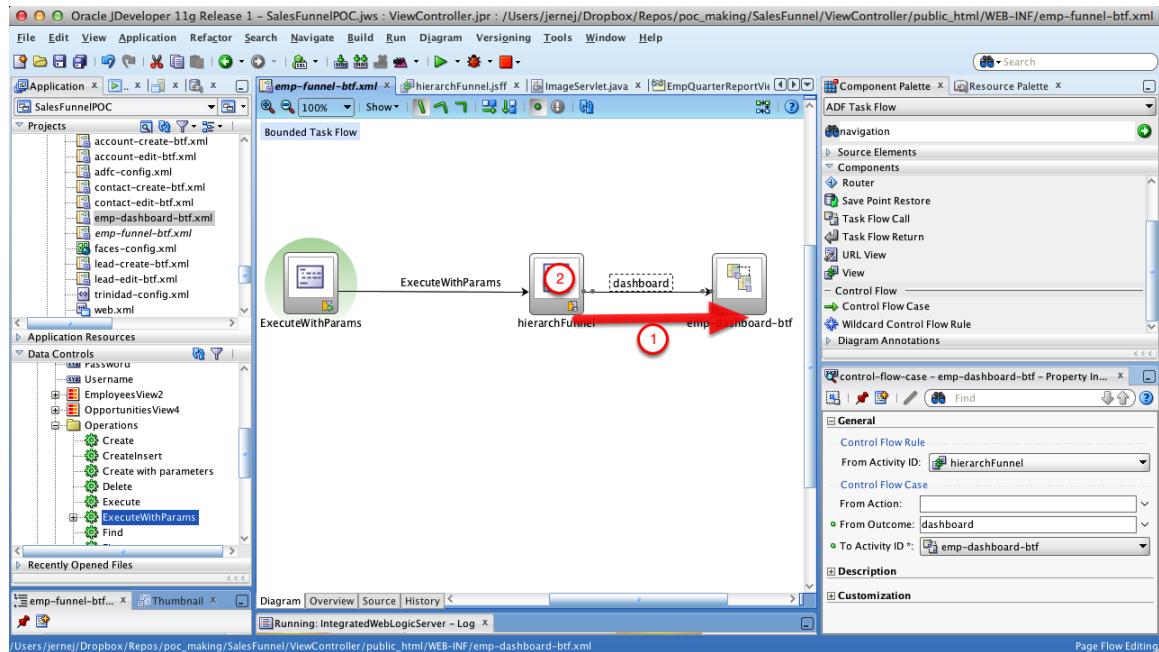
1. Select loadEmp
2. Set it the main activity

Add emp-dashboard-btf to emp-funnel-btf



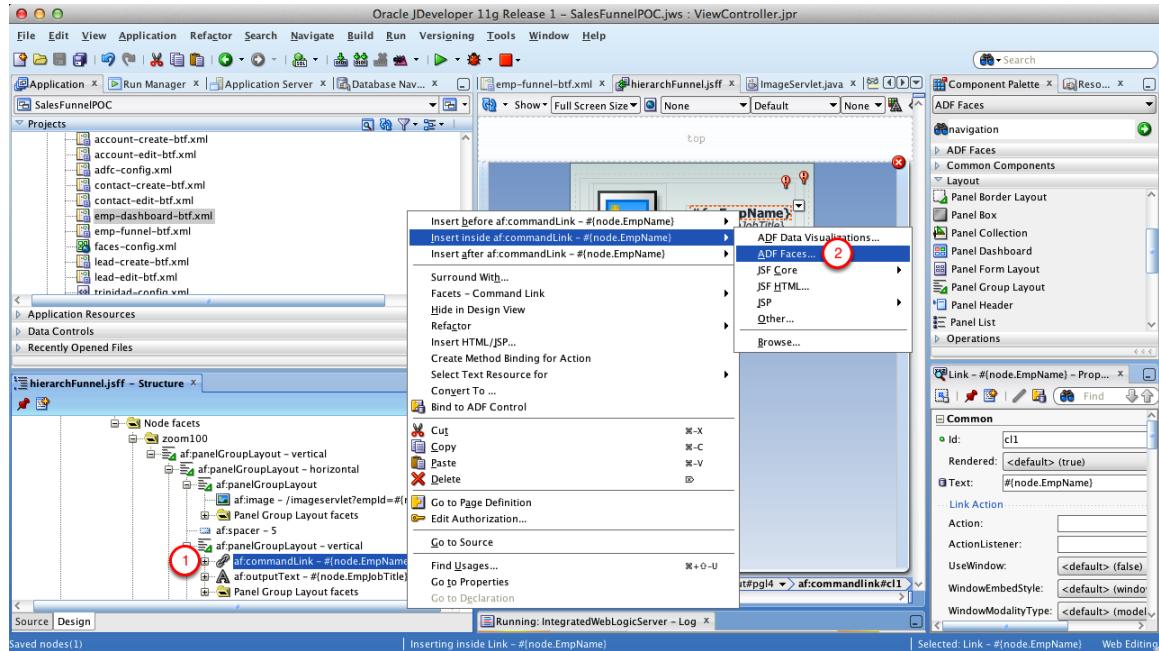
1. Open emp-funnel-btf
2. Drag and drop emp-dashboard-btf to the page

Connect hierarchFunnel to emp-dashboard-btf



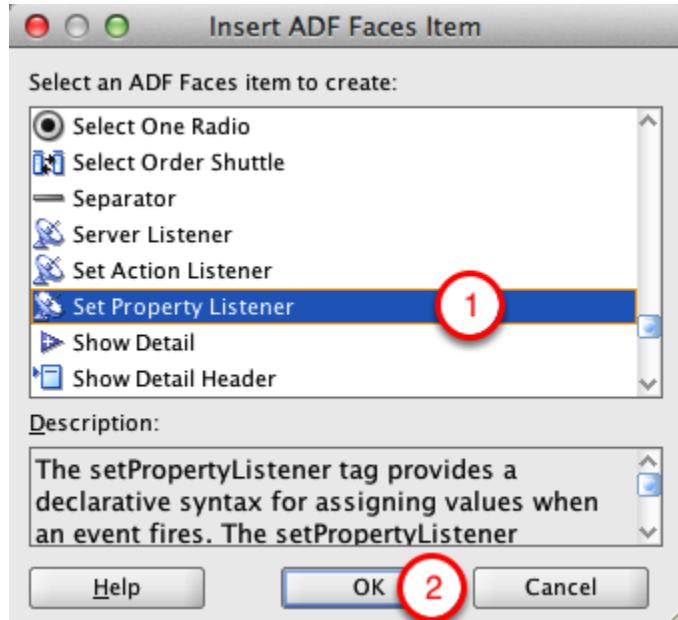
1. Connect hierarchFunnel to emp-dashboard-btf, name the action dashboard
2. Double-click hierarchFunnel to open it

Create Property Listener



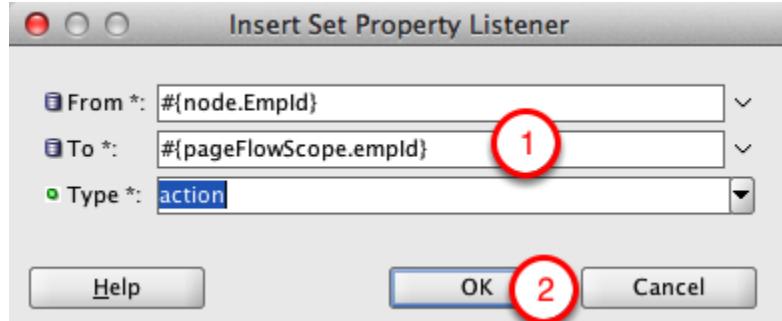
1. Right-click on the commandLink
2. Select Insert inside -> ADF Faces

Insert ADF Faces Item



1. Select "Set Property Listener"
2. Click OK

Insert Set Property Listener

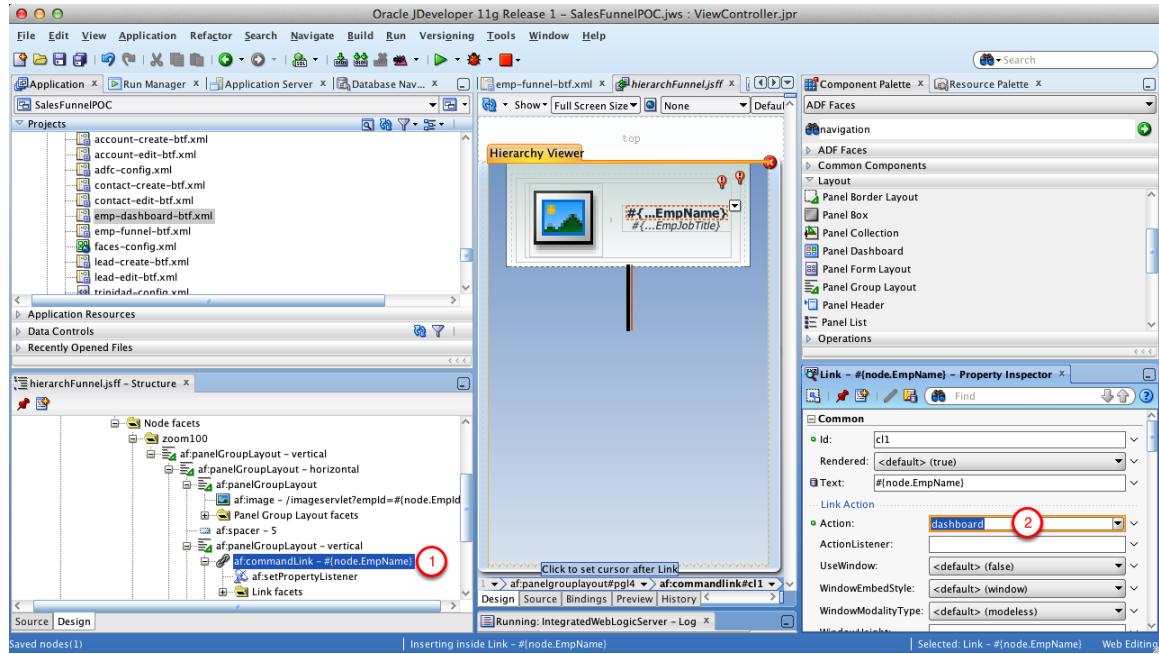


1. Set properties:

- From: #{node.EmpId}
- To: #{pageFlowScope.empId}
- Type: action

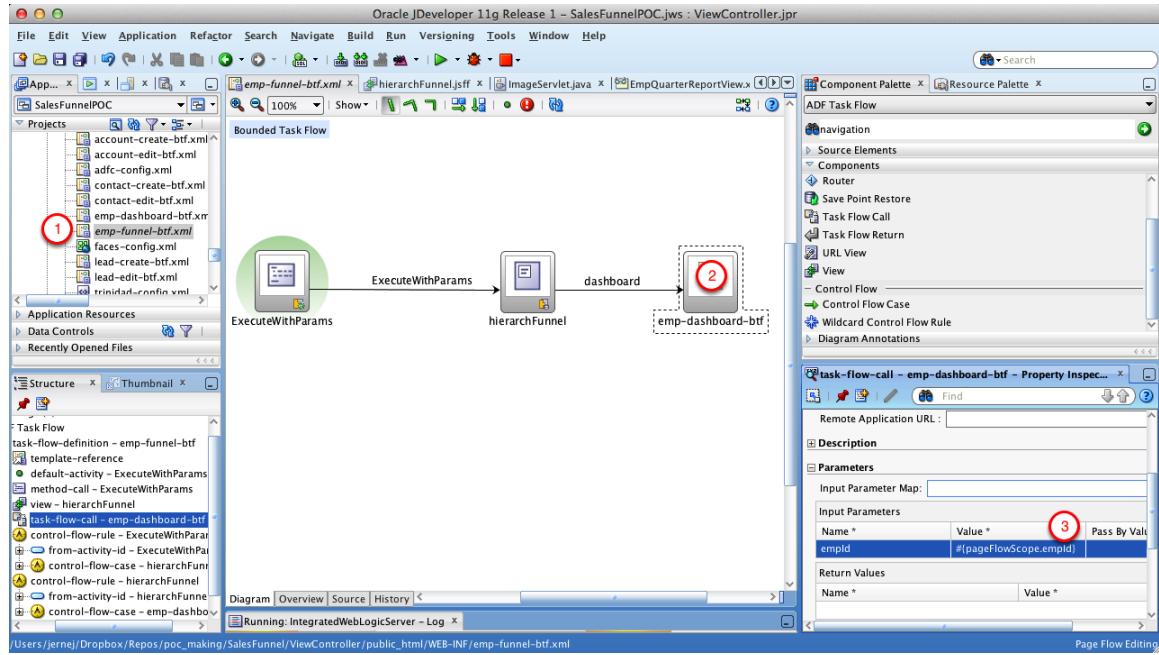
2. Click OK

Set Command Link Action



1. Select commandLink
2. Set Action to dashboard

Set Dashboard empld Parameter Value



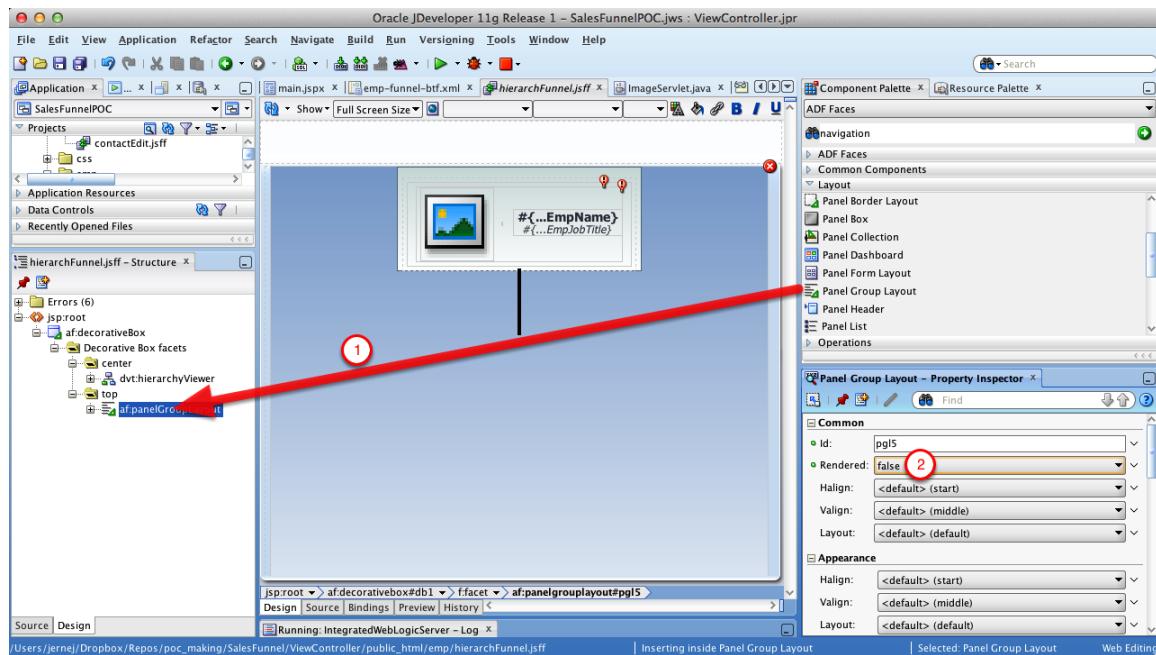
1. Open `emp-funnel-btf`
2. Select `emp-dashboard-btf`
3. Set `empld` parameter value to `#{{pageFlowScope.empld}}`

VI. Extending Hierarchy Viewer with DVT Components

43. Adding DVT to Hierarchy viewer

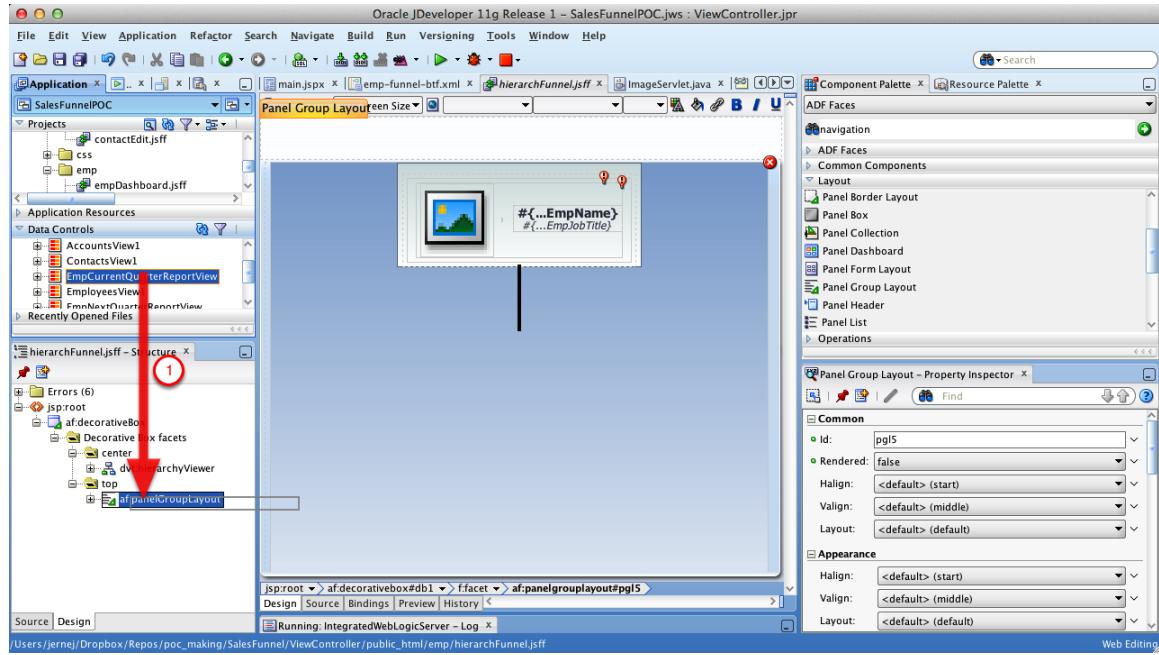
Out of the box Hierarchy Viewer does not support DVT components. But there is a way to display DVT images inside the viewer.

Add A Placeholder for DVT Components



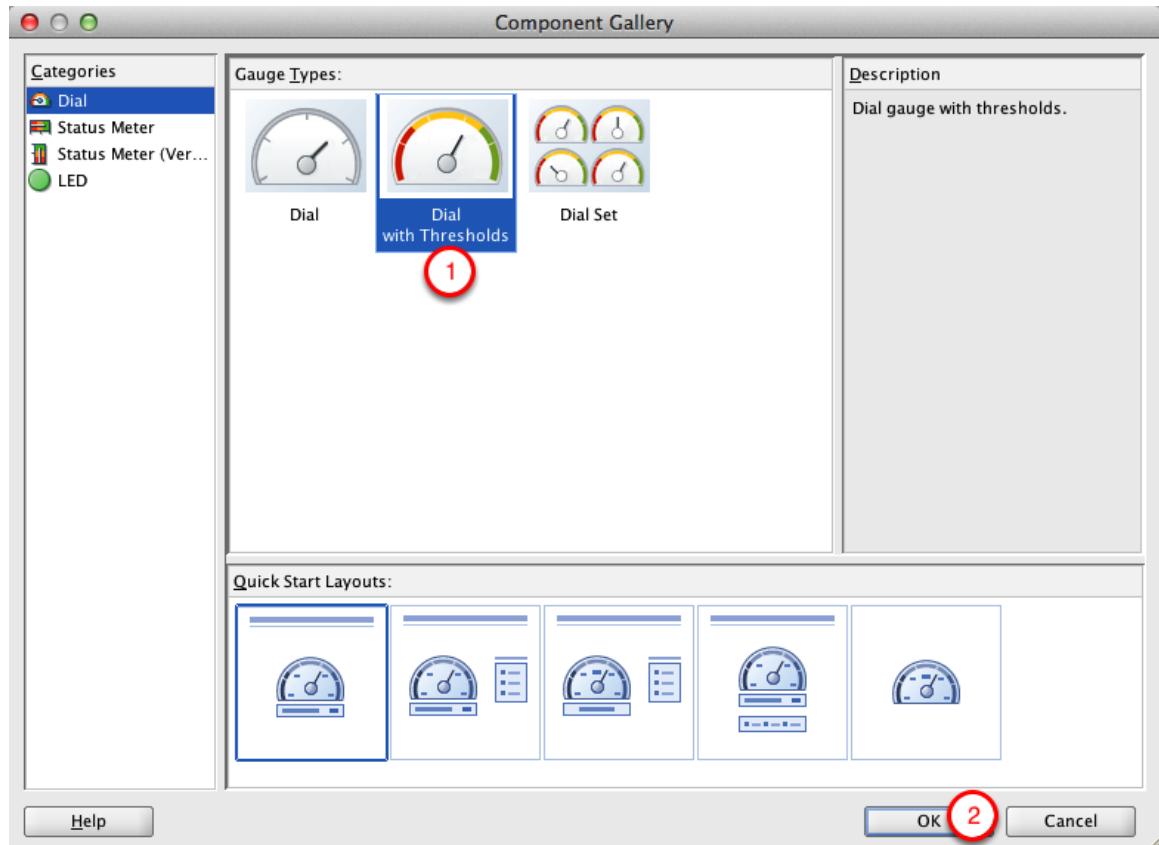
1. Drag and drop Panel Group Layout to the top facet of Decorative Box
2. Set Rendered property to false

Create Target Gauge



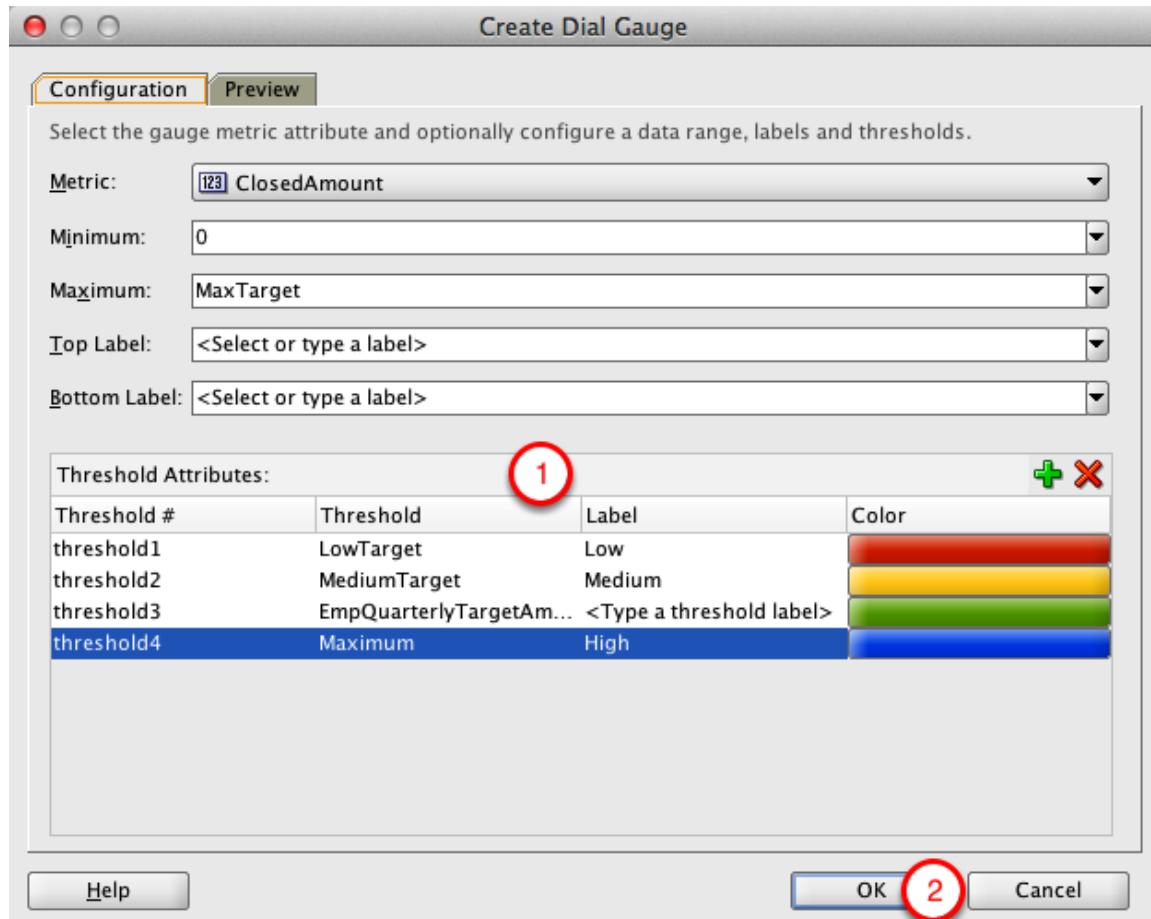
1. Drag And Drop EmpCurrentQuarterReportView to the Panel Group Layout
2. Select Gauge from the Menu

Component Gallery



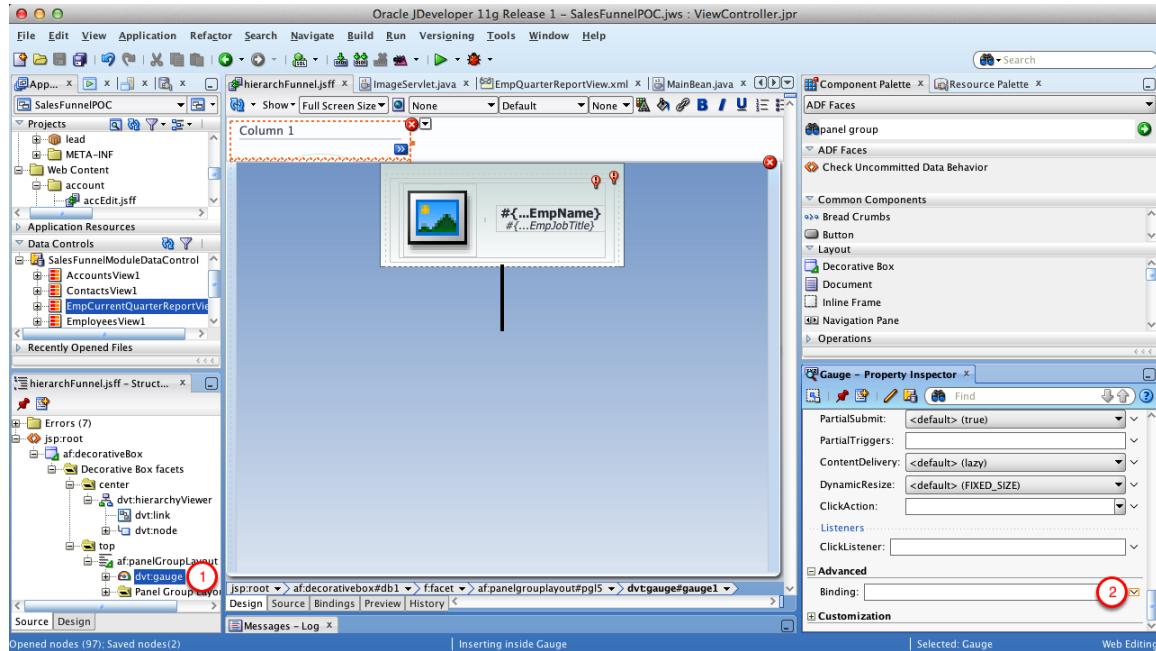
1. Select Dial With Thresholds
2. Click OK

Create Dial Gauge



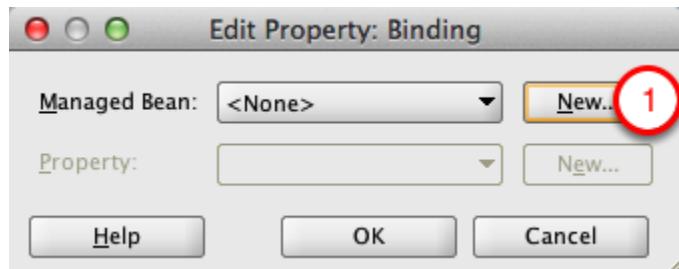
1. Match form properties to the screenshot
2. Click OK

Set Gauge Binding



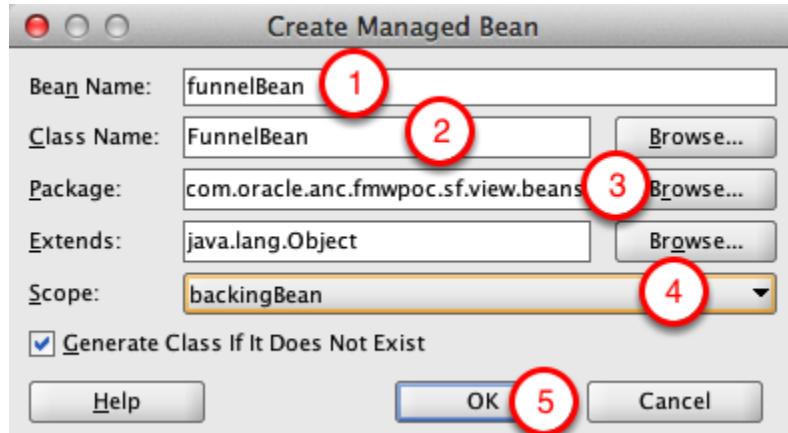
1. Select the Gauge
2. Click the down arrow next to Binding property, click Edit in the popup menu

Edit Property: Binding



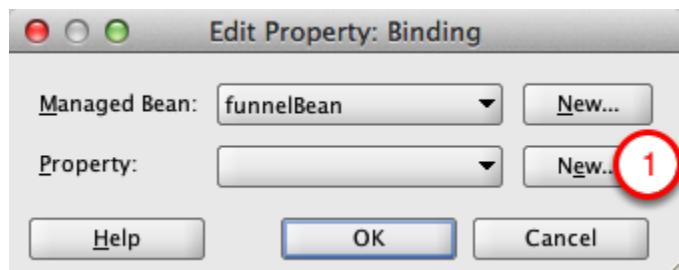
1. Click New to create new bean

Create Managed Bean



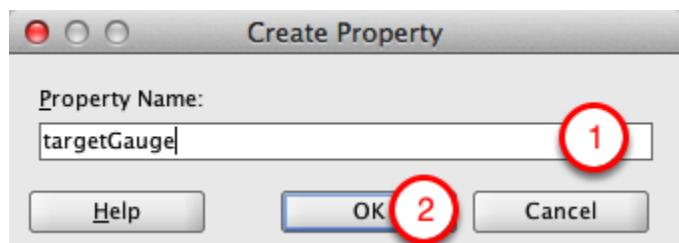
1. Name the bean funnelBean
2. Name the class FunnelBean
3. Set Package to com.oracle.anc.fmw poc.sf.view.beans
4. Set Scope to backingBean
5. Click OK

Edit Property: Binding



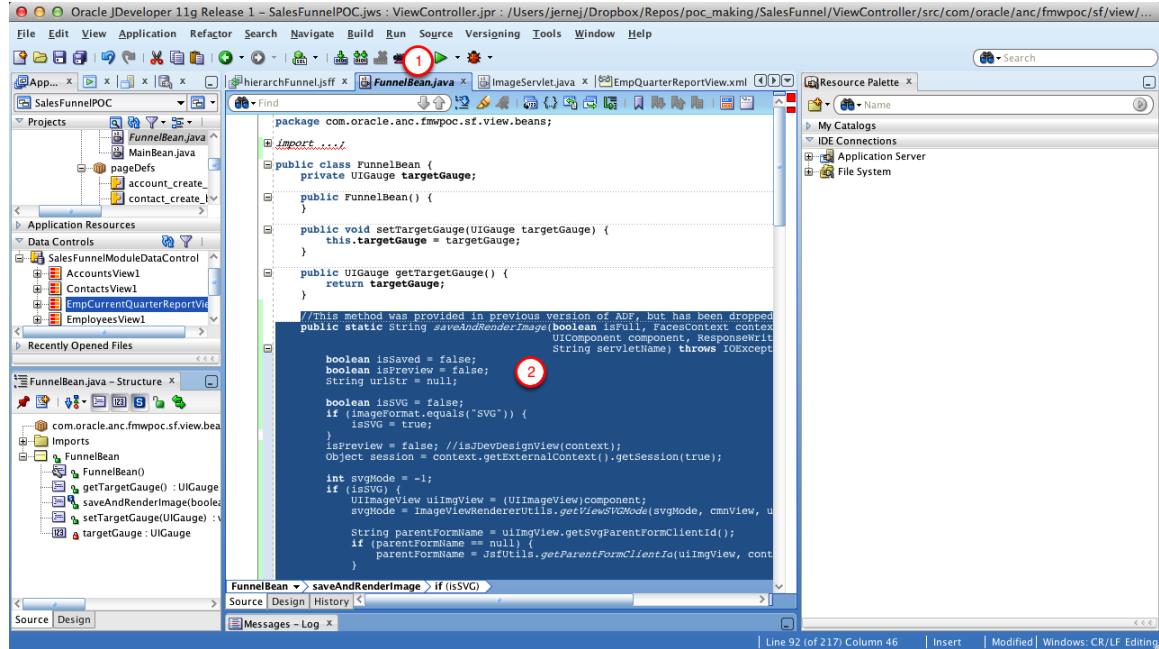
1. Click new to create new property

Create Property



1. Set Name to targetGauge
2. Click OK
3. Click OK to close Property Binding dialog

Paste Implementation



1. Open FunnelBean.java
2. Replace FunnelBean implementation with the code below

```

package com.oracle.anc.fmwpoc.sf.view.beans;

import com.oracle.anc.fmwpoc.sf.view.utils.ADFUtils;
import com.oracle.anc.fmwpoc.sf.view.utils.JSFUtils;

import java.io.BufferedOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.io.Serializable;
import java.io.Writer;

import java.util.Calendar;
import java.util.Date;
import java.util.Map;

import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.context.ResponseWriter;

import oracle.adf.model.BindingContext;
import oracle.adf.view.faces.bi.component.gauge.UIGauge;
import oracle.adf.view.faces.bi.component.imageView.ShapeAttributesCallback;
import oracle.adf.view.faces.bi.component.imageView.ShapeAttributesSet;
import oracle.adf.view.faces.bi.component.imageView.UImageview;
import oracle.adf.view.faces.bi.model.DataModel;

import oracle.adfinternal.view.faces.bi.renderkit.imageView.ImageViewRendererutils;
import oracle.adfinternal.view.faces.bi.renderkit.imageView.ImageViewSGWwriterProviderImpl;
import oracle.adfinternal.view.faces.bi.renderkit.imageView.UImageviewWrapper;
import oracle.adfinternal.view.faces.bi.util.Fileutils;
import oracle.adfinternal.view.faces.bi.util.Jsfutils;
import oracle.adfinternal.view.faces.dvt.model.binding.gauge.FacesGaugeBinding;

import oracle.binding.BindingContainer;
import oracle.binding.OperationBinding;

import oracle.dss.databind.ImageView;
import oracle.dss.databind.SVGwriterProvider;

```

```

import org.apache.myfaces.trinidad.render.CoreRenderer;

public class FunnelBean {
    private UIGauge targetGauge;

    public FunnelBean() {
    }

    public void setTargetGauge(UIGauge targetGauge) {
        this.targetGauge = targetGauge;
    }

    public UIGauge getTargetGauge() {
        return targetGauge;
    }

    public synchronized String getTargetGaugeImage() {
        oracle.jbo.domain.Number empId =
            (oracle.jbo.domain.Number)JSFUtils.resolveExpression("#{node.EmpId}");

        setOperationParam("ExecuteTargetWithParams", "pQuarter",
                           getQuarter(false));
        executeWithParam("ExecuteTargetWithParams", "pEmpId", empId);

        BindingContext bcx = (BindingContext)BindingContext.getCurrent();
        DataModel dm =
            ((FacesGaugeBinding)bcx.getCurrentBindingsEntry()).getControlBinding("EmpCurrentQuarterReportView")).getDataModel();

        UIGauge gauge = getTargetGauge();
        gauge.setRendered(true);
        gauge.setDataModel(dm);
        gauge.transferProperties();
        String url = "";
        try {
            url =
                saveAndRenderImage(false, FacesContext.getCurrentInstance(), gauge.getImageView(),
                                   gauge, null, "PNG", "GaugeServlet");
        } catch (IOException e) {
        }
        gauge.setRendered(false);
        gauge = null;
        url = url.substring(url.indexOf("/", 1));
        return url;
    }

    private String getQuarter(boolean next) {
        Calendar cal = Calendar.getInstance();
        cal.setTime(new Date());
        if (next)
            cal.add(Calendar.MONTH, 3);
        int month = cal.get(Calendar.MONTH);
        return ((Integer)cal.get(Calendar.YEAR)).toString() + "\\" +
            ((Integer)((month / 3) + 1)).toString();
    }

    //This method was provided in previous version of ADF, but has been dropped

    public static String saveAndRenderImage(boolean isFull,
                                            FacesContext context,
                                            ImageView cmnView,
                                            UIComponent component,
                                            ResponseWriter writer,
                                            String imageFormat,
                                            String servletName) throws IOException {
        boolean isSaved = false;
        boolean isPreview = false;
        String urlStr = null;

        boolean isSVG = false;
        if (imageFormat.equals("SVG")) {
            isSVG = true;
        }
        isPreview = false; //isJDevDesignView(context);
        Object session = context.getExternalContext().getSession(true);

        int svgMode = -1;
        if (isSVG) {
            UIImageView uiImgView = (UIImageView)component;
            svgMode =
                ImageViewRendererUtils.getViewSVGMode(svgMode, cmnView, uiImgView);

            String parentFormName = uiImgView.getSvgParentFormClientId();
            if (parentFormName == null) {
                parentFormName =
                    JsfUtils.getParentFormClientId(uiImgView, context);
            }
            SVGWriterProvider provider = cmnView.getSVGWriterProvider();
            ShapeAttributesSet shapeAttrsSet =
                (ShapeAttributesSet)uiImgView.getFacesBean().getProperty(UIImageView.SHAPE_ATTRIBUTES_SET_KEY);
        }
    }
}

```

```

ShapeAttributesCallback shapeAttrsCallback =
    uiImgView.getShapeAttributesCallback();
if ((provider != null) &&
    ((provider instanceof ImageViewSVGWriterProviderImpl))) {
    ((ImageViewSVGWriterProviderImpl)provider).init(context,
        parentFormName,
        shapeAttrsSet,
        shapeAttrsCallback,
        uiImgView.isPartialSubmit());
}

}

String id = ImageViewRendererUtils.constructID(component);

Writer svgWriter = null;
OutputStream svgOutputStream = null;
if ((component instanceof UIImageView)) {
    UIImageView uiImgView = (UIImageView)component;
    svgWriter = uiImgView.getSvgWriter();
    svgOutputStream = uiImgView.getSvgOutputStream();
}

if ((isSVG) && (UIImageViewWrapper.getDisposition(component) == 1) &&
    ((svgWriter != null) || (svgOutputStream != null))) {
    try {
        if (svgWriter != null)
            cmnView.exportToSVGWithException(svgWriter, svgMode, null);
        else if (svgOutputStream != null)
            cmnView.exportToSVGWithException(svgOutputStream, svgMode,
                null);
    } catch (IOException e1) {
        throw e1;
    } catch (Exception e2) {
        // _LOG.severe("Could not export image to given stream or writer", e2);
    }
} else if (isPreview) {
    String server_dir =
        FileUtils.getImageServerPath(context, component);

    File dir = new File(server_dir);
    boolean is_dir = dir.isDirectory();
    if ((!is_dir) && (!dir.exists()))
        is_dir = dir.mkdirs();
    if (is_dir) {
        String prefix =
            new StringBuilder().append("g").append(String.valueOf(System.currentTimeMillis())).toString();
        String suffix = ".png";
        if (isSVG)
            suffix = ".svg";
        File image = File.createTempFile(prefix, suffix, dir);
        if (image != null) {
            BufferedOutputStream out_stream =
                new BufferedOutputStream(new FileOutputStream(image));
            try {
                if (isSVG)
                    cmnView.exportToSVGWithException(out_stream,
                        svgMode, null);
                else
                    cmnView.exportToPNGWithException(out_stream);
                out_stream.close();
                FileUtils.registerForDeletion(context, component,
                    image, id, isPreview);
            } catch (IOException e1) {
                if (isSVG)
                    writer.writeAttribute("src", url.toString(), null);
                else
                    urlStr = url.toString();
                isSaved = true;
            } catch (Exception e2) {
                // _LOG.severe("Could not export image to be saved on disk", e2);
            }
        }
    }
    if (!isSaved) {
        // _LOG.warning("saving Graph image to disk failed");
    }
} else if (session != null) {
    ByteArrayOutputStream stream =
        new SerializableByteArrayStream(10240);
    try {
        if (isSVG)
            cmnView.exportToSVGWithException(stream, svgMode, null);
    }
}

```

```

        else
            cmnView.exportToPNGWithException(stream);
        stream.close();

        String random = component.getClientId(context);
        random =
            new StringBuilder().append(random).append((int)(10000000.0D *
                Math.random())).toString();

        Map sessionMap = context.getExternalContext().getSessionMap();
        if (!isSVG)
            sessionMap.put(new StringBuilder().append(id).append(random).toString(),
                           stream);
        else {
            sessionMap.put(id, stream);
        }
        StringBuilder dest = new StringBuilder(100);

        dest.append(new StringBuilder().append("/servlet/").append(servletName).append("?").toString());
        dest.append("id");
        dest.append('=');
        dest.append(id);
        //dest.append("&random=");
        dest.append(random);

        String url =
            CoreRenderer.toResourceUri(context, dest.toString());
        dest =
new StringBuilder(context.getExternalContext().encodeResourceURL(url));

        if (isSVG) {
            writer.writeAttribute("src", dest.toString(), null);
        } else
            urlStr = dest.toString();
        isSaved = true;
    } catch (IOException e1) {
        throw e1;
    } catch (Exception e2) {
        //_LOG.severe("Could not export image to be saved on session", e2);
    }
}

} else {
    //_LOG.warning("saving Graph image to session failed");
}
return urlStr;
}

private void executeWithParam(String operationBinding, String paramName,
                             Object param) {
    OperationBinding op =
        setOperationParam(operationBinding, paramName, param);
    op.execute();
}

private OperationBinding setOperationParam(String operationBinding,
                                         String paramName,
                                         Object param) {
    BindingContainer dc = ADFUtils.getDCBindingContainer();
    OperationBinding op = dc.getOperationBinding(operationBinding);
    op.getParamsMap().put(paramName, param);
    return op;
}

//helper class for saveAndRenderImage

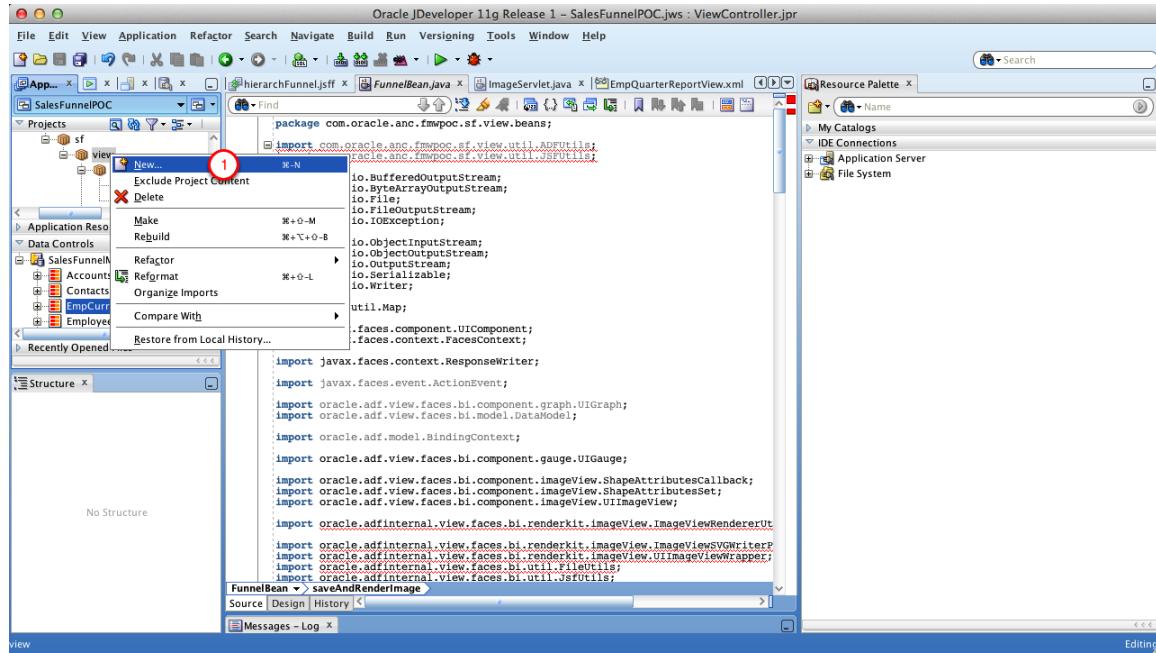
public static class SerializableByteArrayStream extends ByteArrayOutputStream implements Serializable {
    SerializableByteArrayStream(int size) {
        super();
    }

    private void writeObject(ObjectOutputStream o) throws IOException {
        o.writeInt(size());
        writeTo(o);
    }

    private void readObject(ObjectInputStream i) throws IOException,
                                                ClassNotFoundException {
        int size = i.readInt();
        byte[] data = new byte[size];
        i.readFully(data);
        write(data);
    }
}
}

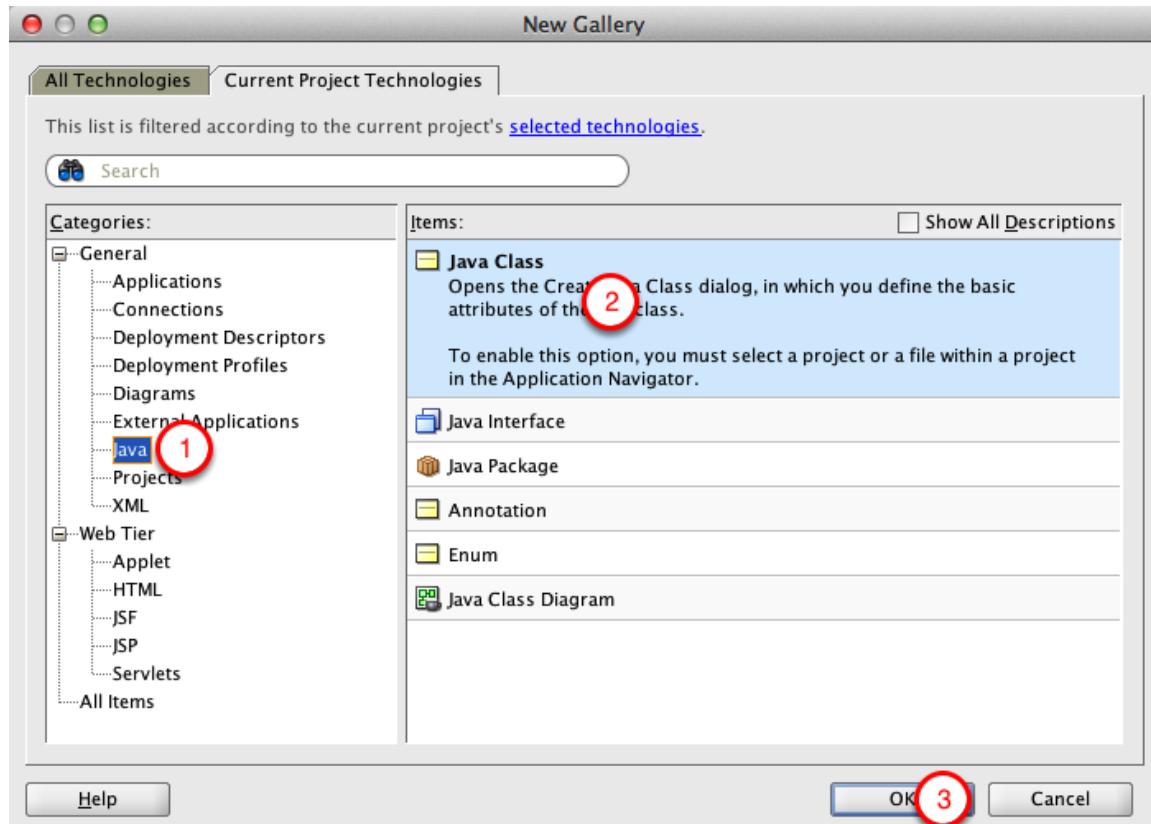
```

Create JSFUtils



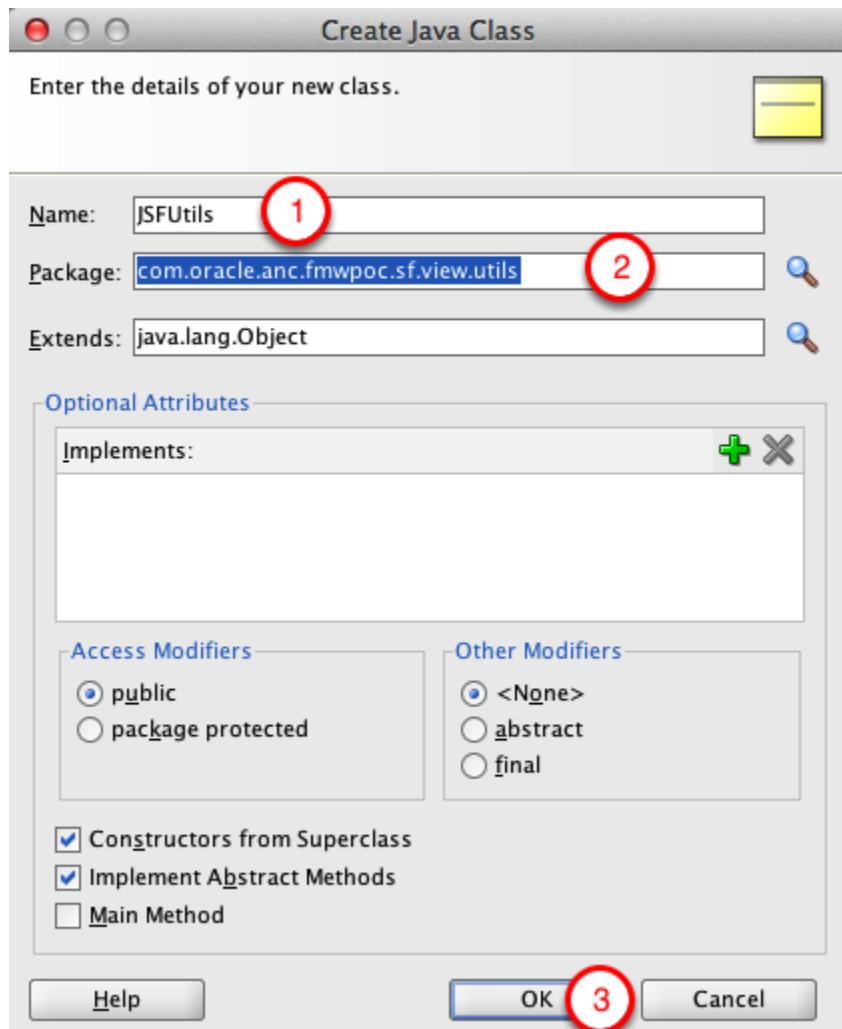
1. Right-click com.oracle.anc.fmwpoc.sf.view package, select new

New Gallery



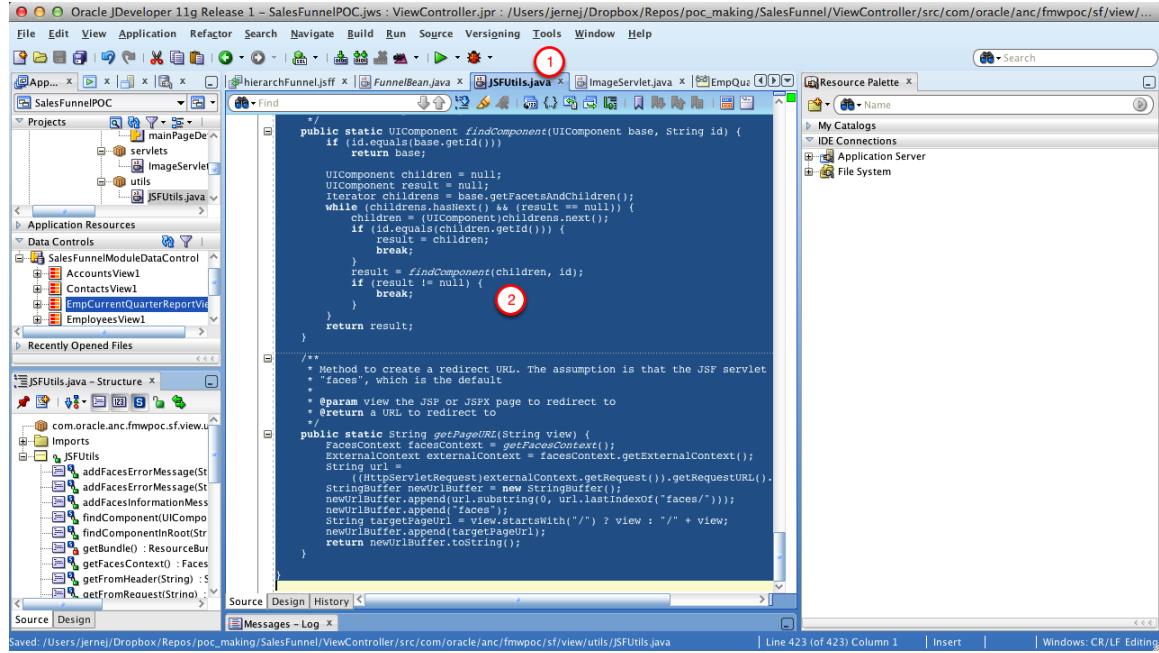
1. Select Java
2. Select Java Class
3. Click OK

Create Java Class



1. Name the class JSFUtils
2. Set Package name to com.oracle.fmw poc.sf.view.utils
3. Click OK

Paste JSFUtils Implementation



1. Open JSFUtils.java
2. Paste the implementation

```

package com.oracle.anc.fmwpoc.sf.view.utils;

import java.util.Iterator;
import java.util.Locale;
import java.util.Map;
import java.util.MissingResourceException;
import java.util.ResourceBundle;

import javax.el.ELContext;
import javax.el.ExpressionFactory;

import javax.el.MethodExpression;
import javax.el.ValueExpression;

import javax.faces.application.Application;
import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.component.UIViewRoot;
import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;

import javax.servlet.http.HttpServletRequest;

/**
 * General useful static utilities for working with JSF.
 * NOTE: Updated to use JSF 1.2 ExpressionFactory.
 *
 * @author Duncan Mills
 * @author Steve Muench
 *
 * $Id: JSFUtils.java 1885 2007-06-26 00:47:41Z ralsmith $
 */
public class JSFUtils {

    private static final String NO_RESOURCE_FOUND = "Missing resource: ";

    /**
     * Method for taking a reference to a JSF binding expression and returning
     * the matching object (or creating it).
     * @param expression EL expression
     * @return Managed object
     */
}

```

```

public static Object resolveExpression(String expression) {
    FacesContext facesContext = getFacesContext();
    Application app = facesContext.getApplication();
    ExpressionFactory elFactory = app.getExpressionFactory();
    ELContext elContext = facesContext.getELContext();
    ValueExpression valueExp =
        elFactory.createValueExpression(elContext, expression,
            Object.class);
    return valueExp.getValue(elContext);
}

/**
 * @return
 */
public static String resolveRemoteUser() {
    FacesContext facesContext = getFacesContext();
    ExternalContext ectx = facesContext.getExternalContext();
    return ectx.getRemoteUser();
}

/**
 * @return
 */
public static String resolveUserPrincipal() {
    FacesContext facesContext = getFacesContext();
    ExternalContext ectx = facesContext.getExternalContext();
    HttpServletRequest request = (HttpServletRequest)ectx.getRequest();
    return request.getUserPrincipal().getName();
}

/**
 * @param expression
 * @param returnType
 * @param argTypes
 * @param argValues
 * @return
 */
public static Object resolveMethodExpression(String expression,
                                             Class returnType,
                                             Class[] argTypes,
                                             Object[] argValues) {
    FacesContext facesContext = getFacesContext();
    Application app = facesContext.getApplication();
    ExpressionFactory elFactory = app.getExpressionFactory();
    ELContext elContext = facesContext.getELContext();
    MethodExpression methodExpression =
        elFactory.createMethodExpression(elContext, expression, returnType,
            argTypes);
    return methodExpression.invoke(elContext, argValues);
}

/**
 * Method for taking a reference to a JSF binding expression and returning
 * the matching Boolean.
 * @param expression EL expression
 * @return Managed object
 */
public static Boolean resolveExpressionAsBoolean(String expression) {
    return (Boolean)resolveExpression(expression);
}

/**
 * Method for taking a reference to a JSF binding expression and returning
 * the matching String (or creating it).
 * @param expression EL expression
 * @return Managed object
 */
public static String resolveExpressionAsString(String expression) {
    return (String)resolveExpression(expression);
}

/**
 * Convenience method for resolving a reference to a managed bean by name
 * rather than by expression.
 * @param beanName name of managed bean
 * @return Managed object
 */
public static Object getManagedBeanValue(String beanName) {
    StringBuffer buff = new StringBuffer("#{");
    buff.append(beanName);
    buff.append("}");
    return resolveExpression(buff.toString());
}

/**
 * Method for setting a new object into a JSF managed bean
 * Note: will fail silently if the supplied object does
 * not match the type of the managed bean.
 * @param expression EL expression
 * @param newValue new value to set
 */
public static void setExpressionValue(String expression, Object newValue) {
    FacesContext facesContext = getFacesContext();
}

```

```

Application app = facesContext.getApplication();
ExpressionFactory elFactory = app.getExpressionFactory();
ELContext elContext = facesContext.getELContext();
ValueExpression valueExp =
    elFactory.createValueExpression(elContext, expression,
        Object.class);

//Check that the input newValue can be cast to the property type
//expected by the managed bean.
//If the managed Bean expects a primitive we rely on Auto-Unboxing
Class bindClass = valueExp.getType(elContext);
if (bindClass.isPrimitive() || bindClass.getInstance(newValue)) {
    valueExp.setValue(elContext, newValue);
}
}

/**
 * Convenience method for setting the value of a managed bean by name
 * rather than by expression.
 * @param beanName name of managed bean
 * @param newValue new value to set
 */
public static void setManagedBeanValue(String beanName, Object newValue) {
    StringBuffer buff = new StringBuffer("#{");
    buff.append(beanName);
    buff.append("}");
    setExpressionValue(buff.toString(), newValue);
}

/**
 * Convenience method for setting Session variables.
 * @param key object key
 * @param object value to store
 */
public static
void storeOnSession(String key, Object object) {
    FacesContext ctx = getFacesContext();
    Map sessionState = ctx.getExternalContext().getSessionMap();
    sessionState.put(key, object);
}

/**
 * Convenience method for getting Session variables.
 * @param key object key
 * @return session object for key
 */
public static Object getFromSession(String key) {
    FacesContext ctx = getFacesContext();
    Map sessionState = ctx.getExternalContext().getSessionMap();
    return sessionState.get(key);
}

/**
 * @param key
 * @return
 */
public static String getFromHeader(String key) {
    FacesContext ctx = getFacesContext();
    ExternalContext ectx = ctx.getExternalContext();
    return ectx.getRequestHeaderMap().get(key);
}

/**
 * Convenience method for getting Request variables.
 * @param key object key
 * @return session object for key
 */
public static Object getFromRequest(String key) {
    FacesContext ctx = getFacesContext();
    Map sessionState = ctx.getExternalContext().getRequestMap();
    return sessionState.get(key);
}

/**
 * Pulls a String resource from the property bundle that
 * is defined under the application <message-bundle> element in
 * the faces config. Respects Locale
 * @param key string message key
 * @return Resource value or placeholder error String
 */
public static String getStringFromBundle(String key) {
    ResourceBundle bundle = getBundle();
    return getStringSafely(bundle, key, null);
}

/**
 * Convenience method to construct a <code>FacesMessage</code>
 * from a defined error key and severity
 * This assumes that the error keys follow the convention of
 * using <b>_detail</b> for the detailed part of the

```

```

        * message, otherwise the main message is returned for the
        * detail as well.
        * @param key for the error message in the resource bundle
        * @param severity severity of message
        * @return Faces Message object
    */
    public static FacesMessage getMessageFromBundle(String key,
                                                    FacesMessage.Severity severity) {
        ResourceBundle bundle = getBundle();
        String summary = getStringSafely(bundle, key, null);
        String detail = getStringSafely(bundle, key + "_detail", summary);
        FacesMessage message = new FacesMessage(summary, detail);
        message.setSeverity(severity);
        return message;
    }

    /**
     * Add JSF info message.
     * @param msg info message string
     */
    public static void addFacesInformationMessage(String msg) {
        FacesContext ctx = getFacesContext();
        FacesMessage fm =
            new FacesMessage(FacesMessage.SEVERITY_INFO, msg, "");
        ctx.addMessage(getRootViewComponentId(), fm);
    }

    /**
     * Add JSF error message.
     * @param msg error message string
     */
    public static void addFacesErrorMessage(String msg) {
        FacesContext ctx = getFacesContext();
        FacesMessage fm =
            new FacesMessage(FacesMessage.SEVERITY_ERROR, msg, "");
        ctx.addMessage(getRootViewComponentId(), fm);
    }

    /**
     * Add JSF error message for a specific attribute.
     * @param attrName name of attribute
     * @param msg error message string
     */
    public static void addFacesErrorMessage(String attrName, String msg) {
        FacesContext ctx = getFacesContext();
        FacesMessage fm =
            new FacesMessage(FacesMessage.SEVERITY_ERROR, attrName, msg);
        ctx.addMessage(getRootViewComponentId(), fm);
    }

    // Informational getters

    /**
     * Get view id of the view root.
     * @return view id of the view root
     */
    public static String getRootViewId() {
        return getFacesContext().getViewRoot().getViewId();
    }

    /**
     * Get component id of the view root.
     * @return component id of the view root
     */
    public static String getRootViewComponentId() {
        return getFacesContext().getViewRoot().getId();
    }

    /**
     * Get FacesContext.
     * @return FacesContext
     */
    public static FacesContext getFacesContext() {
        return FacesContext.getCurrentInstance();
    }

    /**
     * Internal method to pull out the correct local
     * message bundle
     */
    private static ResourceBundle getBundle() {
        FacesContext ctx = getFacesContext();
        UIViewRoot uiRoot = ctx.getViewRoot();
        Locale locale = uiRoot.getLocale();
        ClassLoader ldr = Thread.currentThread().getContextClassLoader();
        return ResourceBundle.getBundle(ctx.getApplication().getMessageBundle(),
                                        locale, ldr);
    }

    /**
     * Get an HTTP Request attribute.
     * @param name attribute name
     * @return attribute value
     */
}

```

```

/*
public static Object getRequestAttribute(String name) {
    return getFacesContext().getExternalContext().getRequestMap().get(name);
}

/**
 * Set an HTTP Request attribute.
 * @param name attribute name
 * @param value attribute value
 */
public static void setRequestAttribute(String name, Object value) {
    getFacesContext().getExternalContext().getRequestMap().put(name,
                                                               value);
}

/*
 * Internal method to proxy for resource keys that don't exist
 */

private static String getStringSafely(ResourceBundle bundle, String key,
                                      String defaultValue) {
    String resource = null;
    try {
        resource = bundle.getString(key);
    } catch (MissingResourceException mrex) {
        if (defaultValue != null) {
            resource = defaultValue;
        } else {
            resource = NO_RESOURCE_FOUND + key;
        }
    }
    return resource;
}

/**
 * Locate an UIComponent in view root with its component id. Use a recursive way to achieve this.
 * @param id UIComponent id
 * @return UIComponent object
 */
public static UIComponent findComponentInRoot(String id) {
    UIComponent component = null;
    FacesContext facesContext = FacesContext.getCurrentInstance();
    if (facesContext != null) {
        UIComponent root = facesContext.getViewRoot();
        component = findComponent(root, id);
    }
    return component;
}

/**
 * Locate an UIComponent from its root component.
 * Taken from http://www.jroller.com/page/mert?entry=how_to_find_a_uicomponent
 * @param base root Component (parent)
 * @param id UIComponent id
 * @return UIComponent object
 */
public static UIComponent findComponent(UIComponent base, String id) {
    if (id.equals(base.getId()))
        return base;

    UIComponent children = null;
    UIComponent result = null;
    Iterator childrens = base.getFacetsAndChildren();
    while (childrens.hasNext() && (result == null)) {
        children = (UIComponent)childrens.next();
        if (id.equals(children.getId())) {
            result = children;
            break;
        }
        result = findComponent(children, id);
        if (result != null)
            break;
    }
    return result;
}

/**
 * Method to create a redirect URL. The assumption is that the JSF servlet mapping is
 * "faces", which is the default
 *
 * @param view the JSP or JSRX page to redirect to
 * @return a URL to redirect to
 */
public static String getPageURL(String view) {
    FacesContext facesContext = getFacesContext();
    ExternalContext externalContext = facesContext.getExternalContext();
    String url =
        ((HttpServletRequest)externalContext.getRequest()).getRequestURL().toString();
    StringBuffer newUrlBuffer = new StringBuffer();
    newUrlBuffer.append(url.substring(0, url.lastIndexOf("faces/")));
    newUrlBuffer.append("faces");
    String targetPageUrl = view.startsWith("/") ? view : "/" + view;
}

```

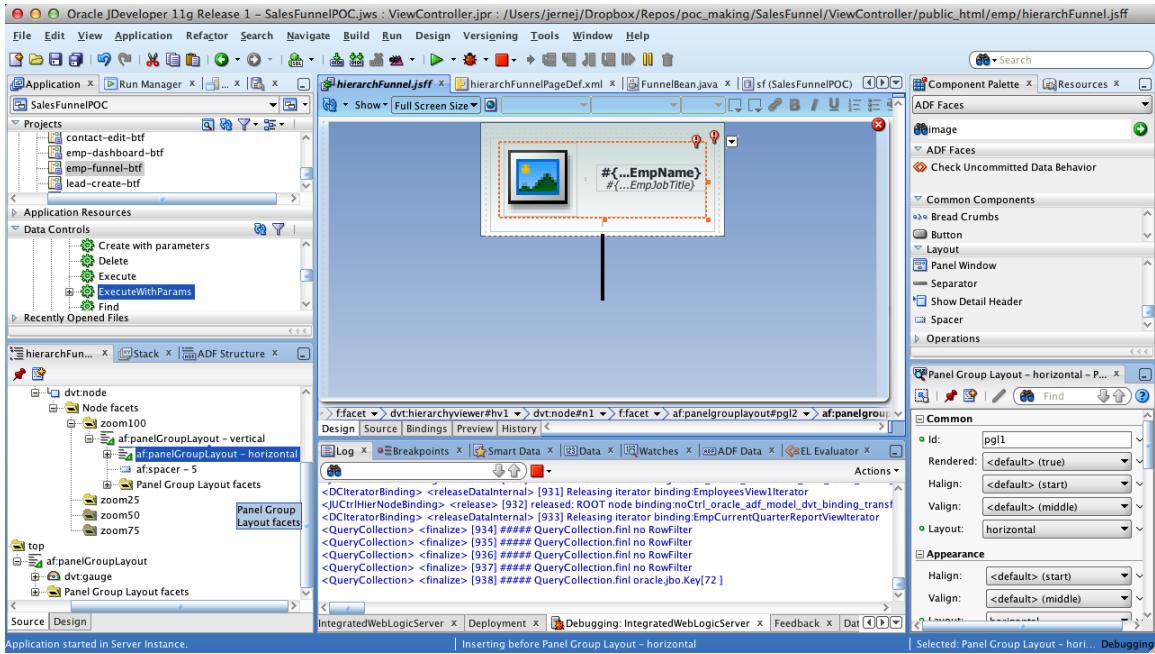
```

        newUrlBuffer.append(targetPageUrl);
        return newUrlBuffer.toString();
    }

}

```

Exercise



Repeat the steps to create a class ADFUtils in the same package.
 Paste the below implementation in the file.

```

package com.oracle.anc.fmwpoc.sf.view.utils;

import java.util.ArrayList;
import java.util.List;

import javax.faces.model.SelectItem;

import oracle.adf.model.BindingContext;
import oracle.adf.model.binding.DCBindingContainer;
import oracle.adf.model.binding.DCIteratorBinding;
import oracle.adf.model.binding.DCParameter;

import oracle.adf.share.logging.ADFLogger;

import oracle.binding.AttributeBinding;
import oracle.binding.BindingContainer;

import oracle.binding.ControlBinding;

import oracle.binding.OperationBinding;

import oracle.jbo.ApplicationModule;
import oracle.jbo.Key;
import oracle.jbo.Row;
import oracle.jbo.uicli.binding.JUCtrlValueBinding;

/**
 * A series of convenience functions for dealing with ADF Bindings.
 * Note: Updated for JDeveloper 11
 *
 * @author Duncan Mills
 * @author Steve Muench
 *
 * $Id: ADFUtils.java 2513 2007-09-20 20:39:13Z ralsmith $

```

```

/*
public class ADFUtils {

    public static final ADFLogger LOGGER = ADFLogger.createADFLogger(ADFUtils.class);

    /**
     * Get application module for an application module data control by name.
     * @param name application module data control name
     * @return ApplicationModule
     */
    public static ApplicationModule getApplicationModuleForDataControl(String name) {
        return (ApplicationModule)JSFUtils.resolveExpression("#{data." + name +
            ".dataProvider}");
    }

    /**
     * A convenience method for getting the value of a bound attribute in the
     * current page context programatically.
     * @param attributeName of the bound value in the pageDef
     * @return value of the attribute
     */
    public static Object getBoundAttributeValue(String attributeName) {
        return findControlBinding(attributeName).getInputValue();
    }

    /**
     * A convenience method for setting the value of a bound attribute in the
     * context of the current page.
     * @param attributeName of the bound value in the pageDef
     * @param value to set
     */
    public static void setBoundAttributeValue(String attributeName,
                                             Object value) {
        findControlBinding(attributeName).setInputValue(value);
    }

    /**
     * Returns the evaluated value of a pageDef parameter.
     * @param pageDefName reference to the page definiton file of the page with the parameter
     * @param parameterName name of the pagedef parameter
     * @return evaluated value of the parameter as a String
     */
    public static Object getPageDefParameterValue(String pageDefName,
                                                 String parameterName) {
        BindingContainer bindings = findBindingContainer(pageDefName);
        DCParameter param =
            ((DCBindingContainer)bindings).findParameter(parameterName);
        return param.getValue();
    }

    /**
     * Convenience method to find a DCControlBinding as an AttributeBinding
     * to get able to then call getInputValue() or setInputValue() on it.
     * @param bindingContainer binding container
     * @param attributeName name of the attribute binding.
     * @return the control value binding with the name passed in.
     */
    public static AttributeBinding findControlBinding(BindingContainer bindingContainer,
                                                       String attributeName) {
        if (attributeName != null) {
            if (bindingContainer != null) {
                ControlBinding ctrlBinding =
                    bindingContainer.getControlBinding(attributeName);
                if (ctrlbinding instanceof AttributeBinding) {
                    return (AttributeBinding)ctrlBinding;
                }
            }
        }
        return null;
    }

    /**
     * Convenience method to find a DCControlBinding as a JUCtrlValueBinding
     * to get able to then call getInputValue() or setInputValue() on it.
     * @param attributeName name of the attribute binding.
     * @return the control value binding with the name passed in.
     */
    public static AttributeBinding findControlBinding(String attributeName) {
        return findControlBinding(getBindingContainer(), attributeName);
    }

    /**
     * Return the current page's binding container.
     * @return the current page's binding container
     */
    public static BindingContainer getBindingContainer() {
        return (BindingContainer)JSFUtils.resolveExpression("#{bindings}");
    }

    /**
     * Return the Binding Container as a DCBindingContainer.
     */
}

```

```

        * @return current binding container as a DCBindingContainer
    */
    public static DCBindingContainer getDCBindingContainer() {
        return (DCBindingContainer)getBindingContainer();
    }

    /**
     * Get List of ADF Faces SelectItem for an iterator binding.
     *
     * Uses the value of the 'valueAttrName' attribute as the key for
     * the SelectItem key.
     *
     * @param iteratorName ADF iterator binding name
     * @param valueAttrName name of the value attribute to use
     * @param displayAttrName name of the attribute from iterator rows to display
     * @return ADF Faces SelectItem for an iterator binding
     */
    public static List<SelectItem> selectItemsForIterator(String iteratorName,
                                                          String valueAttrName,
                                                          String displayAttrName) {
        return selectItemsForIterator(findIterator(iteratorName),
                                     valueAttrName, displayAttrName);
    }

    /**
     * Get List of ADF Faces SelectItem for an iterator binding with description.
     *
     * Uses the value of the 'valueAttrName' attribute as the key for
     * the Selectitem key.
     *
     * @param iteratorName ADF iterator binding name
     * @param valueAttrName name of the value attribute to use
     * @param displayAttrName name of the attribute from iterator rows to display
     * @param descriptionAttrName name of the attribute to use for description
     * @return ADF Faces SelectItem for an iterator binding with description
     */
    public static List<SelectItem> selectItemsForIterator(String iteratorName,
                                                          String valueAttrName,
                                                          String displayAttrName,
                                                          String descriptionAttrName) {
        return selectItemsForIterator(findIterator(iteratorName),
                                     valueAttrName, displayAttrName,
                                     descriptionAttrName);
    }

    /**
     * Get List of attribute values for an iterator.
     * @param iteratorName ADF iterator binding name
     * @param valueAttrName value attribute to use
     * @return List of attribute values for an iterator
     */
    public static List attributeListForIterator(String iteratorName,
                                              String valueAttrName) {
        return attributeListForIterator(findIterator(iteratorName),
                                       valueAttrName);
    }

    /**
     * Get List of Key objects for rows in an iterator.
     * @param iteratorName iterabot binding name
     * @return List of Key objects for rows
     */
    public static List<Key> keyListForIterator(String iteratorName) {
        return keyListForIterator(findIterator(iteratorName));
    }

    /**
     * Get List of Key objects for rows in an iterator.
     * @param iter iterator binding
     * @return List of Key objects for rows
     */
    public static List<Key> keyListForIterator(DCIteratorBinding iter) {
        List<Key> attributeList = new ArrayList<Key>();
        for (Row r : iter.getAllRowsInRange()) {
            attributeList.add(r.getKey());
        }
        return attributeList;
    }

    /**
     * Get List of Key objects for rows in an iterator using key attribute.
     * @param iteratorName iterator binding name
     * @param keyAttrName name of key attribute to use
     * @return List of Key objects for rows
     */
    public static List<Key> keyAttrListForIterator(String iteratorName,
                                                String keyAttrName) {
        return keyAttrListForIterator(findIterator(iteratorName), keyAttrName);
    }

    /**
     * Get List of Key objects for rows in an iterator using key attribute.
     */

```

```

        * @param iter iterator binding
        * @param keyAttrName name of key attribute to use
        * @return List of Key objects for rows
        */
    public static List<Key> keyAttrListForIterator(DCIteratorBinding iter,
                                                   String keyAttrName) {
        List<Key> attributeList = new ArrayList<Key>();
        for (Row r : iter.getAllRowsInRange()) {
            attributeList.add(new Key(new Object[] { r.getAttribute(keyAttrName) }));
        }
        return attributeList;
    }

    /**
     * Get a List of attribute values for an iterator.
     *
     * @param iter iterator binding
     * @param valueAttrName name of value attribute to use
     * @return List of attribute values
     */
    public static List attributeListForIterator(DCIteratorBinding iter,
                                                String valueAttrName) {
        List attributeList = new ArrayList();
        for (Row r : iter.getAllRowsInRange()) {
            attributeList.add(r.getAttribute(valueAttrName));
        }
        return attributeList;
    }

    /**
     * Find an iterator binding in the current binding container by name.
     *
     * @param name iterator binding name
     * @return iterator binding
     */
    public static DCIteratorBinding findIterator(String name) {
        DCIteratorBinding iter =
            getDCBindingContainer().findIteratorBinding(name);
        if (iter == null) {
            throw new RuntimeException("Iterator '" + name + "' not found");
        }
        return iter;
    }

    /**
     * @param bindingContainer
     * @param iterator
     * @return
     */
    public static DCIteratorBinding findIterator(String bindingContainer, String iterator) {
        DCBindingContainer bindings =
            (DCBindingContainer)JSFUtils.resolveExpression("#{bindings[" + bindingContainer + "]}");
        if (bindings == null) {
            throw new RuntimeException("Binding container '" +
                bindingContainer + "' not found");
        }
        DCIteratorBinding iter = bindings.findIteratorBinding(iterator);
        if (iter == null) {
            throw new RuntimeException("Iterator '" + iterator + "' not found");
        }
        return iter;
    }

    /**
     * @param name
     * @return
     */
    public static JUCtrlValueBinding findCtrlBinding(String name) {
        JUCtrlValueBinding rowBinding =
            (JUCtrlValueBinding)getDCBindingContainer().findCtrlBinding(name);
        if (rowBinding == null) {
            throw new RuntimeException("CtrlBinding " + name + "' not found");
        }
        return rowBinding;
    }

    /**
     * Find an operation binding in the current binding container by name.
     *
     * @param name operation binding name
     * @return operation binding
     */
    public static OperationBinding findOperation(String name) {
        OperationBinding op =
            getDCBindingContainer().getOperationBinding(name);
        if (op == null) {
            throw new RuntimeException("Operation '" + name + "' not found");
        }
        return op;
    }

    /**
     * Find an operation binding in the current binding container by name.
     *

```

```

/*
 * @param bindingContianer binding container name
 * @param opName operation binding name
 * @return operation binding
 */
public static OperationBinding findOperation(String bindingContianer,
                                             String opName) {
    DCBindingContainer bindings =
        (DCBindingContainer)JSFUtils.resolveExpression("#{ " + bindingContianer + " }");
    if (bindings == null) {
        throw new RuntimeException("Binding container '" +
                                   bindingContianer + "' not found");
    }
    OperationBinding op =
        bindings.getOperationBinding(opName);
    if (op == null) {
        throw new RuntimeException("Operation '" + opName + "' not found");
    }
    return op;
}

/**
 * Get List of ADF Faces SelectItem for an iterator binding with description.
 *
 * Uses the value of the 'valueAttrName' attribute as the key for
 * the SelectItem key.
 *
 * @param iter ADF iterator binding
 * @param valueAttrName name of value attribute to use for key
 * @param displayAttrName name of the attribute from iterator rows to display
 * @param descriptionAttrName name of the attribute for description
 * @return ADF Faces SelectItem for an iterator binding with description
 */
public static List<SelectItem> selectItemsForIterator(DCIteratorBinding iter,
                                                       String valueAttrName,
                                                       String displayAttrName,
                                                       String descriptionAttrName) {
    List<SelectItem> selectItems = new ArrayList<SelectItem>();
    for (Row r : iter.getAllRowsInRange()) {
        selectItems.add(new SelectItem(r.getAttribute(valueAttrName),
                                       (String)r.getAttribute(displayAttrName),
                                       (String)r.getAttribute(descriptionAttrName)));
    }
    return selectItems;
}

/**
 * Get List of ADF Faces SelectItem for an iterator binding.
 *
 * Uses the value of the 'valueAttrName' attribute as the key for
 * the SelectItem key.
 *
 * @param iter ADF iterator binding
 * @param valueAttrName name of value attribute to use for key
 * @param displayAttrName name of the attribute from iterator rows to display
 * @return ADF Faces SelectItem for an iterator binding
 */
public static List<SelectItem> selectItemsForIterator(DCIteratorBinding iter,
                                                       String valueAttrName,
                                                       String displayAttrName) {
    List<SelectItem> selectItems = new ArrayList<SelectItem>();
    for (Row r : iter.getAllRowsInRange()) {
        selectItems.add(new SelectItem(r.getAttribute(valueAttrName),
                                       (String)r.getAttribute(displayAttrName)));
    }
    return selectItems;
}

/**
 * Get List of ADF Faces SelectItem for an iterator binding.
 *
 * Uses the rowKey of each row as the SelectItem key.
 *
 * @param iteratorName ADF iterator binding name
 * @param displayAttrName name of the attribute from iterator rows to display
 * @return ADF Faces SelectItem for an iterator binding
 */
public static List<SelectItem> selectItemsByKeyForIterator(String iteratorName,
                                                          String displayAttrName) {
    return selectItemsByKeyForIterator(findIterator(iteratorName),
                                      displayAttrName);
}

/**
 * Get List of ADF Faces SelectItem for an iterator binding with discription.
 *
 * Uses the rowKey of each row as the SelectItem key.
 *
 * @param iteratorName ADF iterator binding name
 * @param displayAttrName name of the attribute from iterator rows to display
 * @param descriptionAttrName name of the attribute for description
 * @return ADF Faces SelectItem for an iterator binding with discription
 */

```

```

public static List<SelectItem> selectItemsByKeyForIterator(String iteratorName,
                                                       String displayAttrName,
                                                       String descriptionAttrName) {
    return selectItemsByKeyForIterator(findIterator(iteratorName),
                                       displayAttrName,
                                       descriptionAttrName);
}

/**
 * Get List of ADF Faces SelectItem for an iterator binding with discription.
 *
 * Uses the rowKey of each row as the SelectItem key.
 *
 * @param iter ADF iterator binding
 * @param displayAttrName name of the attribute from iterator rows to display
 * @param descriptionAttrName name of the attribute for description
 * @return ADF Faces SelectItem for an iterator binding with discription
 */
public static List<SelectItem> selectItemsByKeyForIterator(DCIteratorBinding iter,
                                                       String displayAttrName,
                                                       String descriptionAttrName) {
    List<SelectItem> selectItems = new ArrayList<SelectItem>();
    for (Row r : iter.getAllRowsInRange()) {
        selectItems.add(new SelectItem(r.getKey(),
                                       (String)r.getAttribute(displayAttrName),
                                       (String)r.getAttribute(descriptionAttrName)));
    }
    return selectItems;
}

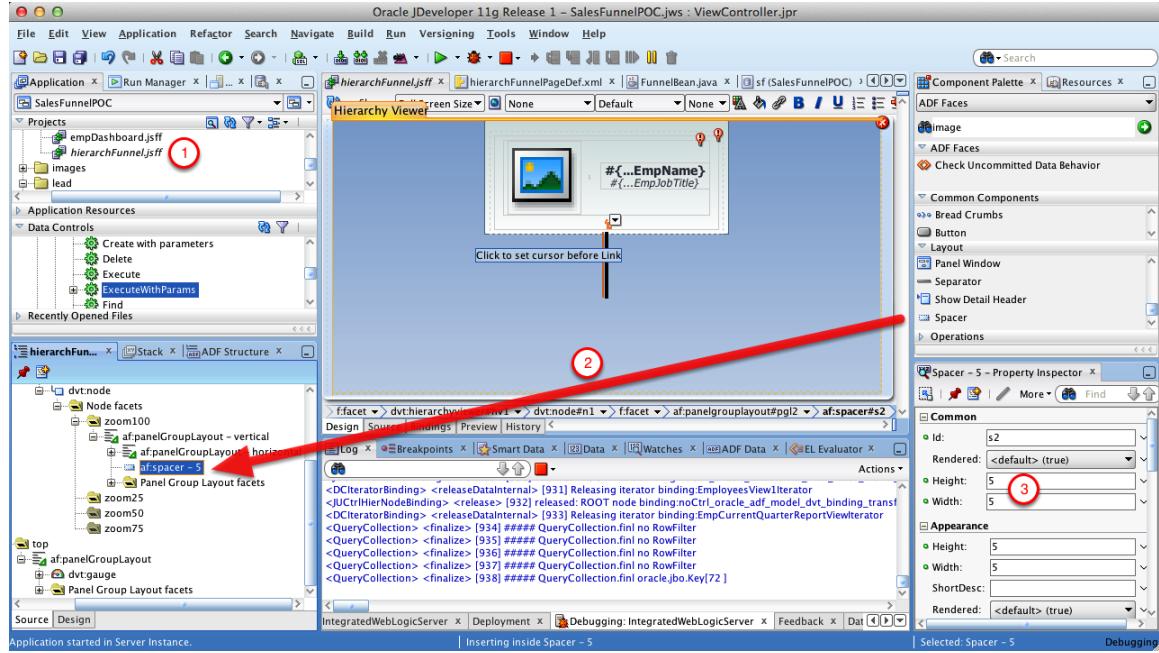
/**
 * Get List of ADF Faces SelectItem for an iterator binding.
 *
 * Uses the rowKey of each row as the SelectItem key.
 *
 * @param iter ADF iterator binding
 * @param displayAttrName name of the attribute from iterator rows to display
 * @return List of ADF Faces SelectItem for an iterator binding
 */
public static List<SelectItem> selectItemsByKeyForIterator(DCIteratorBinding iter,
                                                       String displayAttrName) {
    List<SelectItem> selectItems = new ArrayList<SelectItem>();
    for (Row r : iter.getAllRowsInRange()) {
        selectItems.add(new SelectItem(r.getKey(),
                                       (String)r.getAttribute(displayAttrName)));
    }
    return selectItems;
}

/**
 * Find the BindingContainer for a page definition by name.
 *
 * Typically used to refer eagerly to page definition parameters. It is
 * not best practice to reference or set bindings in binding containers
 * that are not the one for the current page.
 *
 * @param pageDefName name of the page defintion XML file to use
 * @return BindingContainer ref for the named definition
 */
private static BindingContainer findBindingContainer(String pageDefName) {
    BindingContext bctx = getDCBindingContainer().getBindingContext();
    BindingContainer foundContainer =
        bctx.findBindingContainer(pageDefName);
    return foundContainer;
}

/**
 * @param opList
 */
public static void printOperationBindingExceptions(List opList){
    if(opList != null && !opList.isEmpty()){
        for(Object error:opList){
            LOGGER.severe( error.toString() );
        }
    }
}
}

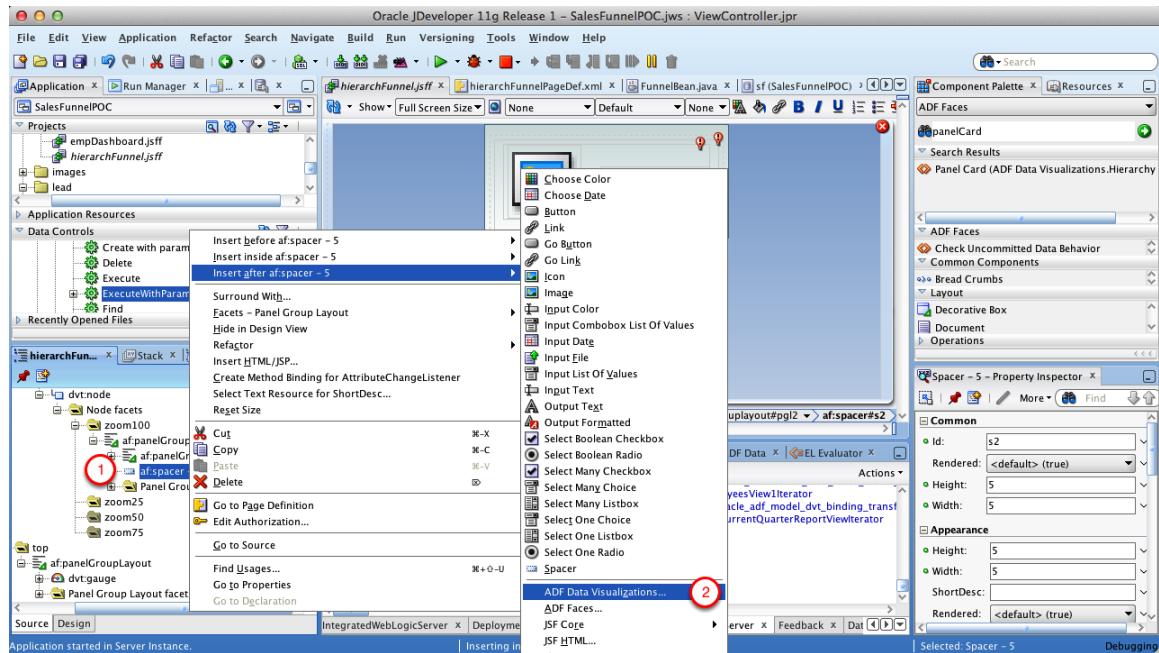
```

Change Hierarchy Node Layout



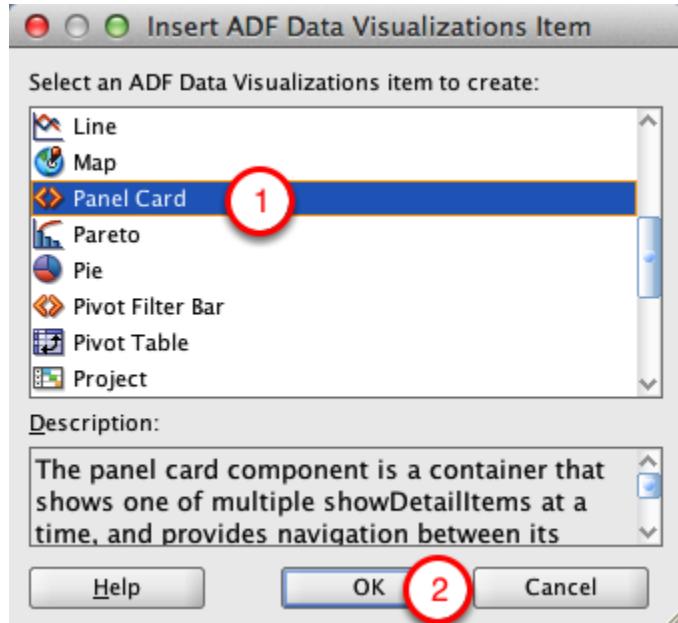
1. Open hierarchFunnel.jsff
2. Drag and drop Spacer inside panel group layout - vertical, after panel group layout - horizontal
3. Set Height and Width to 5

Add Panel Card



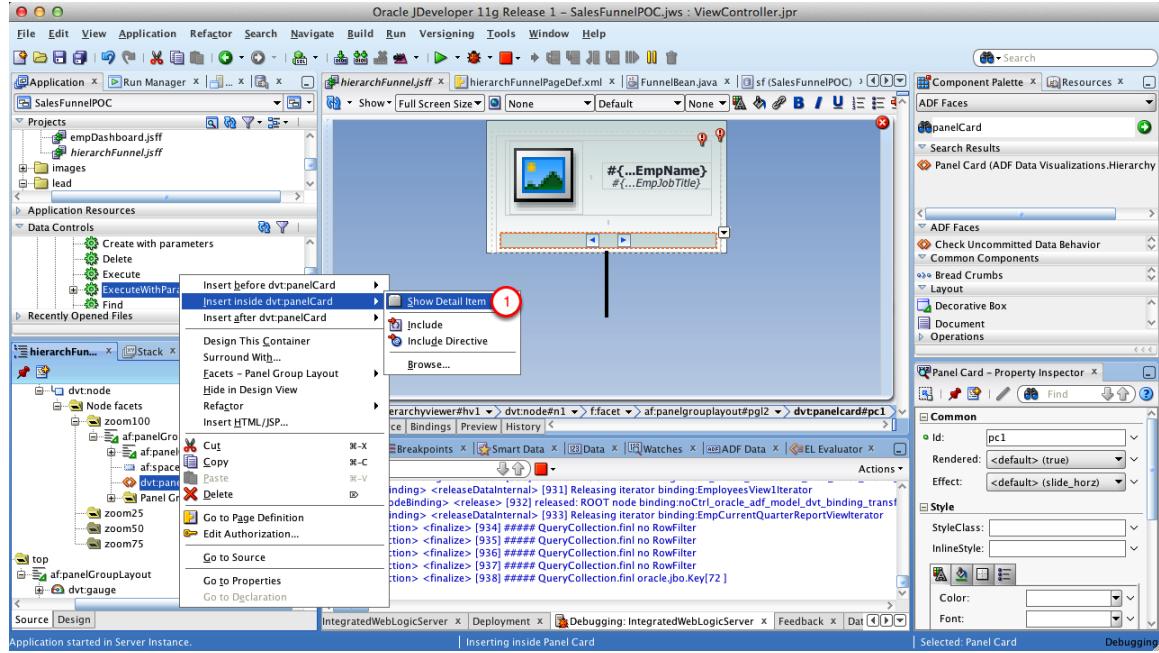
1. Right-click the spacer
2. Select Insert after > ADF Data Visualizations

Insert ADF Data Visualizations Item



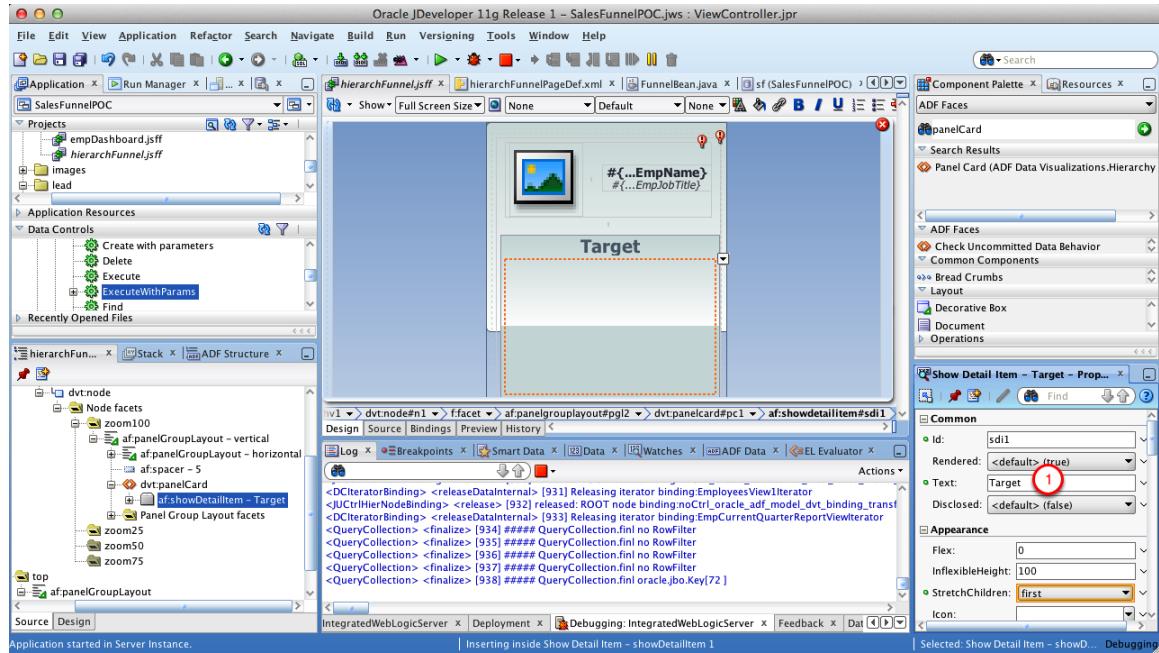
1. Select Panel Card
2. Click OK

Add Card



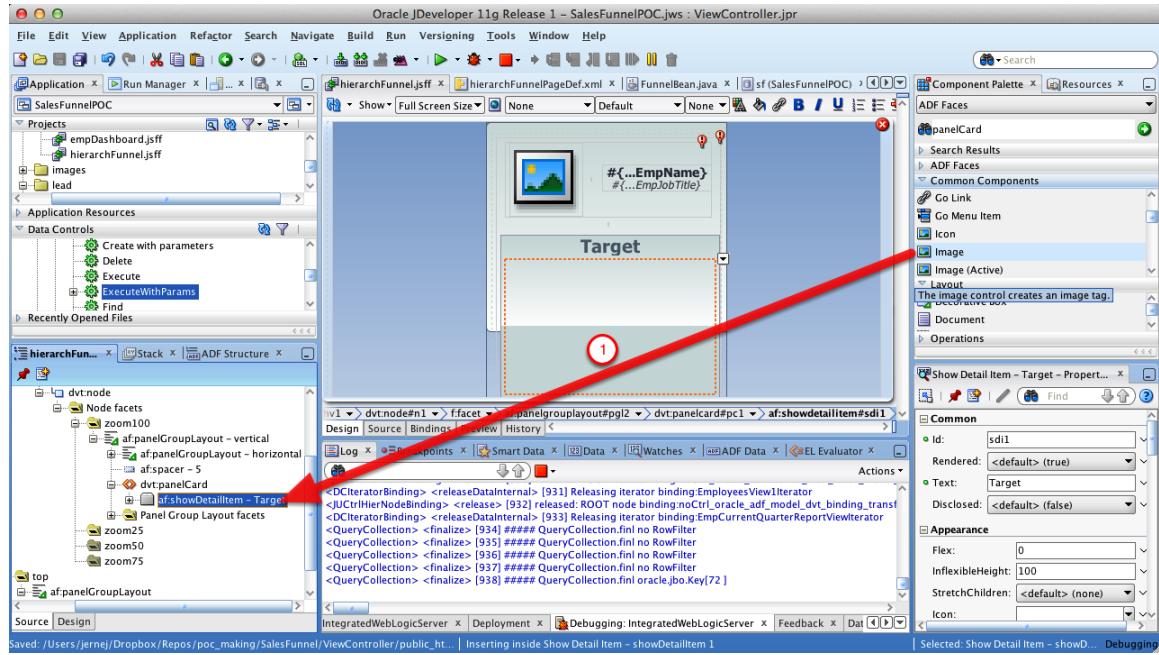
1. Insert showDetailItem inside Panel Card

Set Card Properties



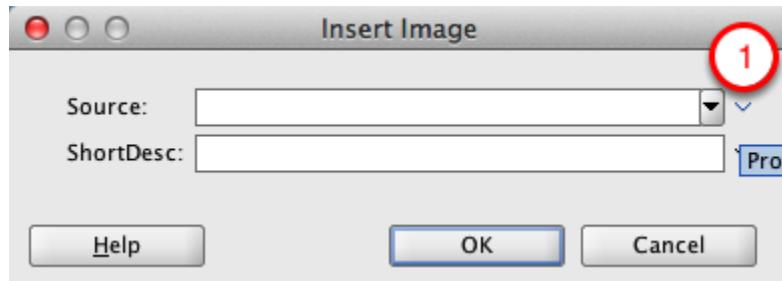
1. Set Text property to Target

Add Image



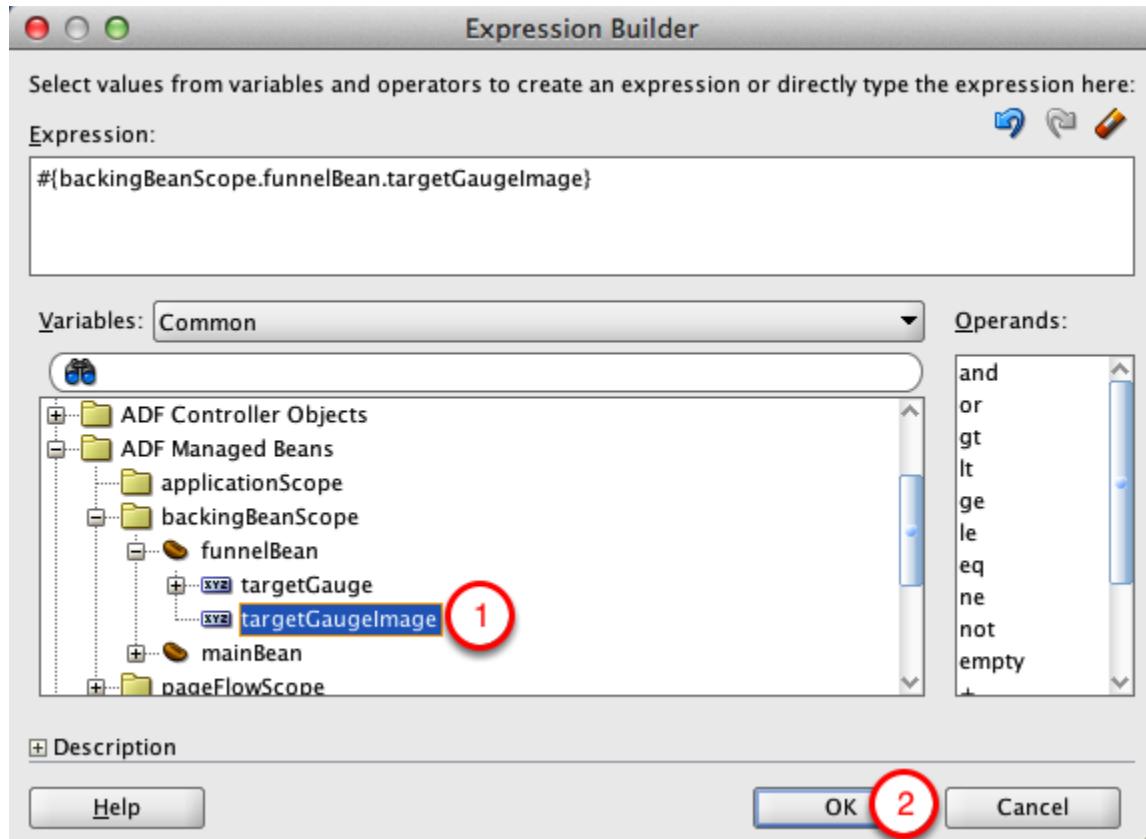
1. Drag and drop Image inside Show Detail Item

Insert Image



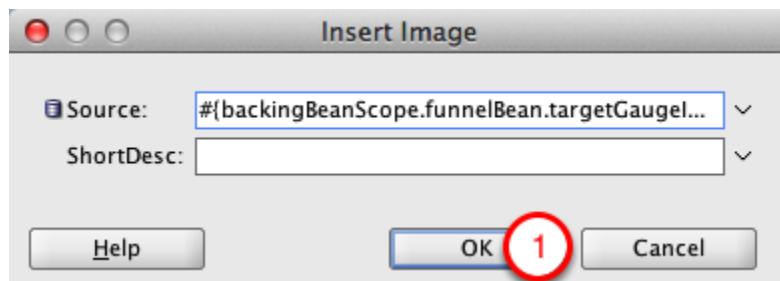
1. Click the small down arrow next to Source property and select Expression Builder

Expression Builder



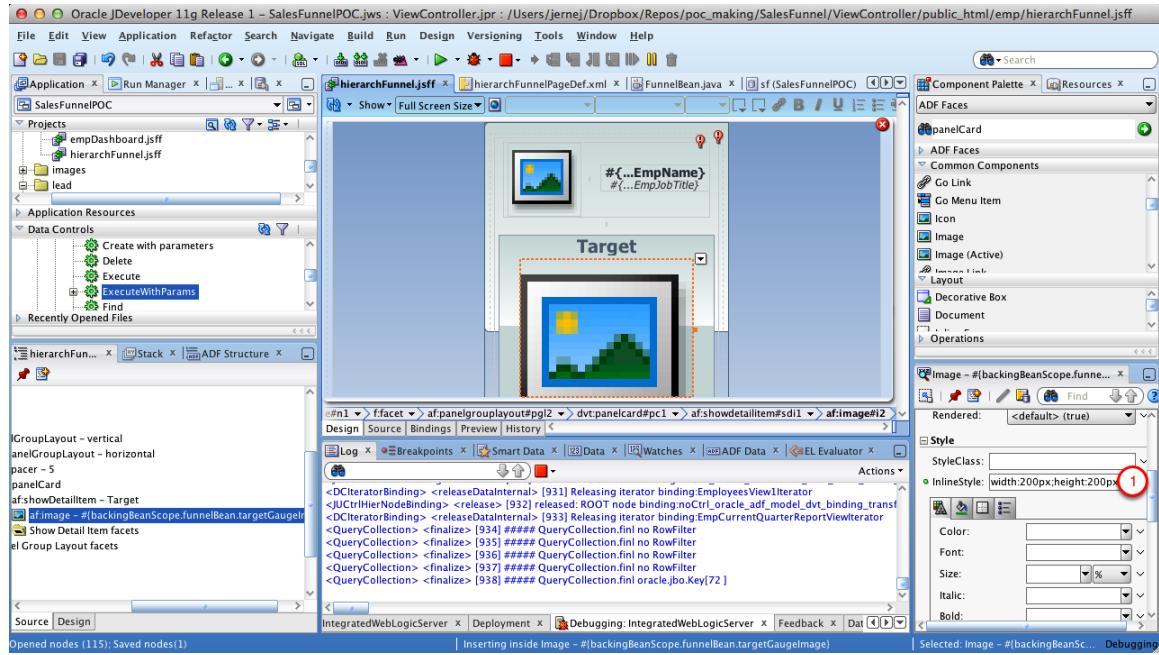
1. Select ADF Managed Beans > backingBeanScope > targetGaugeImage
2. Click OK

Insert Image



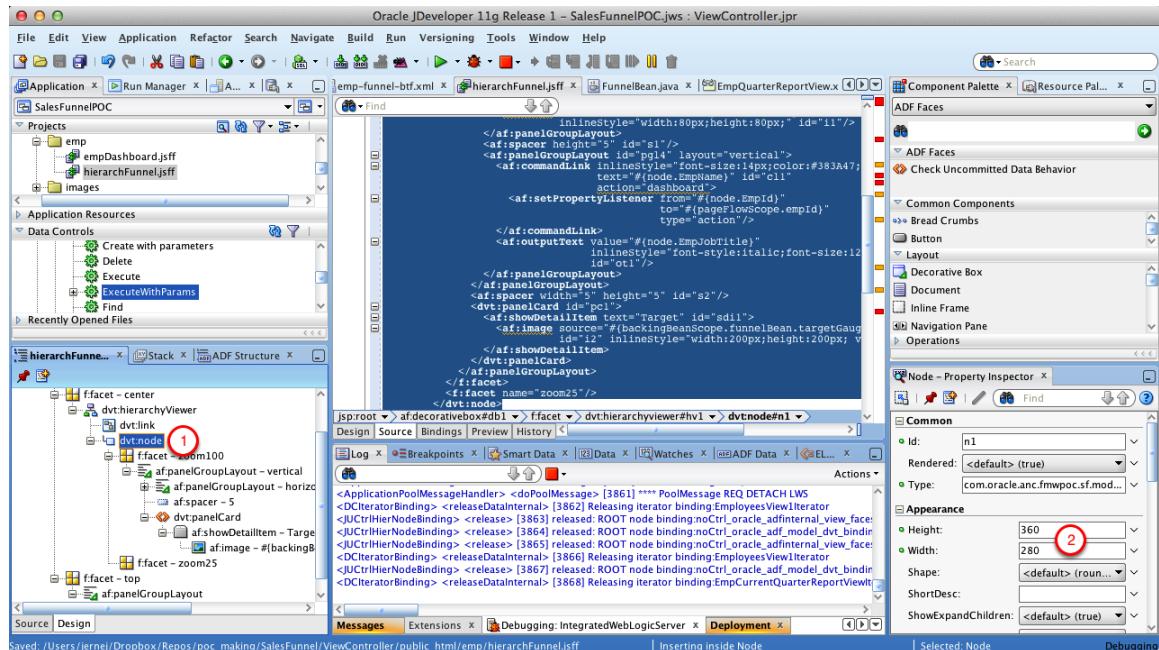
1. Click OK to close the dialog

Set Image Style



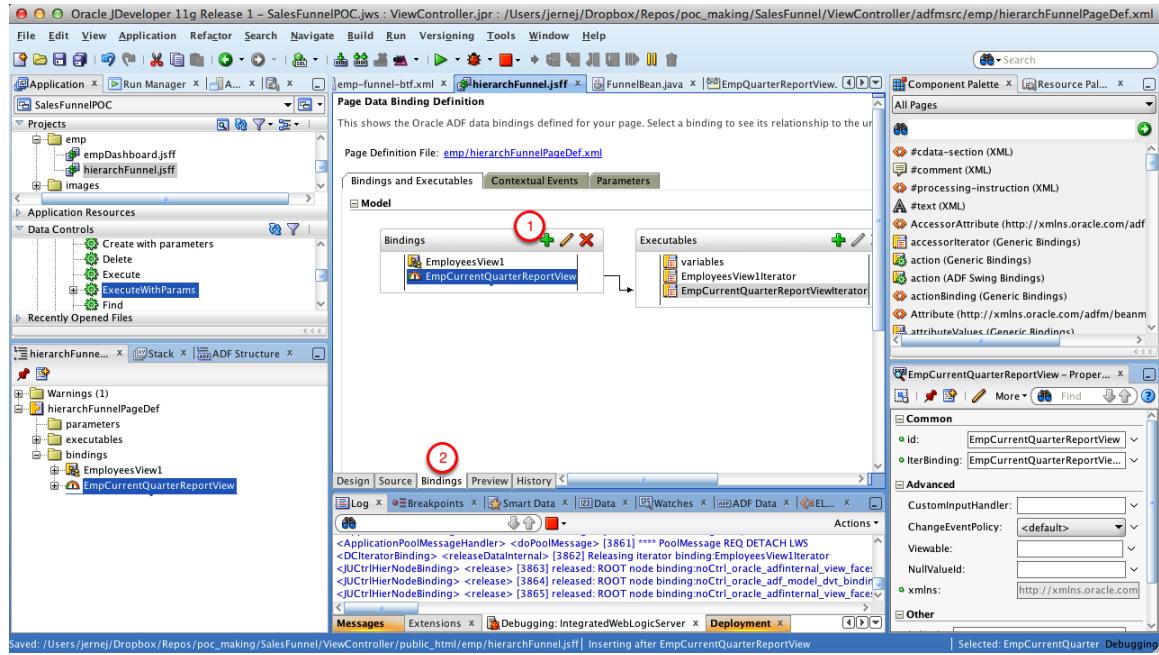
1. Set `InlineStyle` property to "`width:200px;height:200px; vertical-align:middle; text-align:center;`" without quotes

Set Node Style



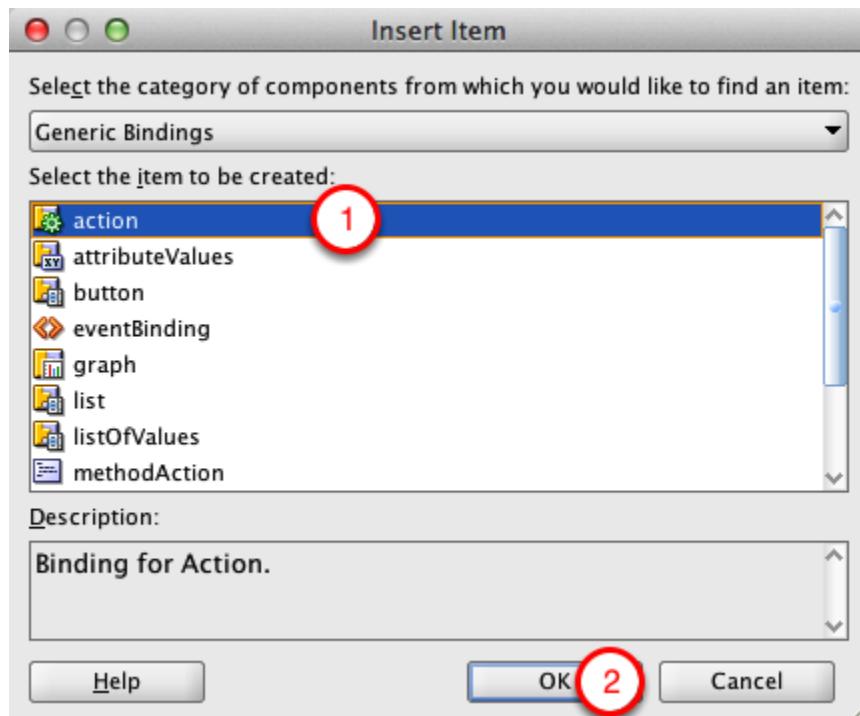
1. Select `dvt:node`
2. Set Height to 360 and Width to 280

Add Action Binding



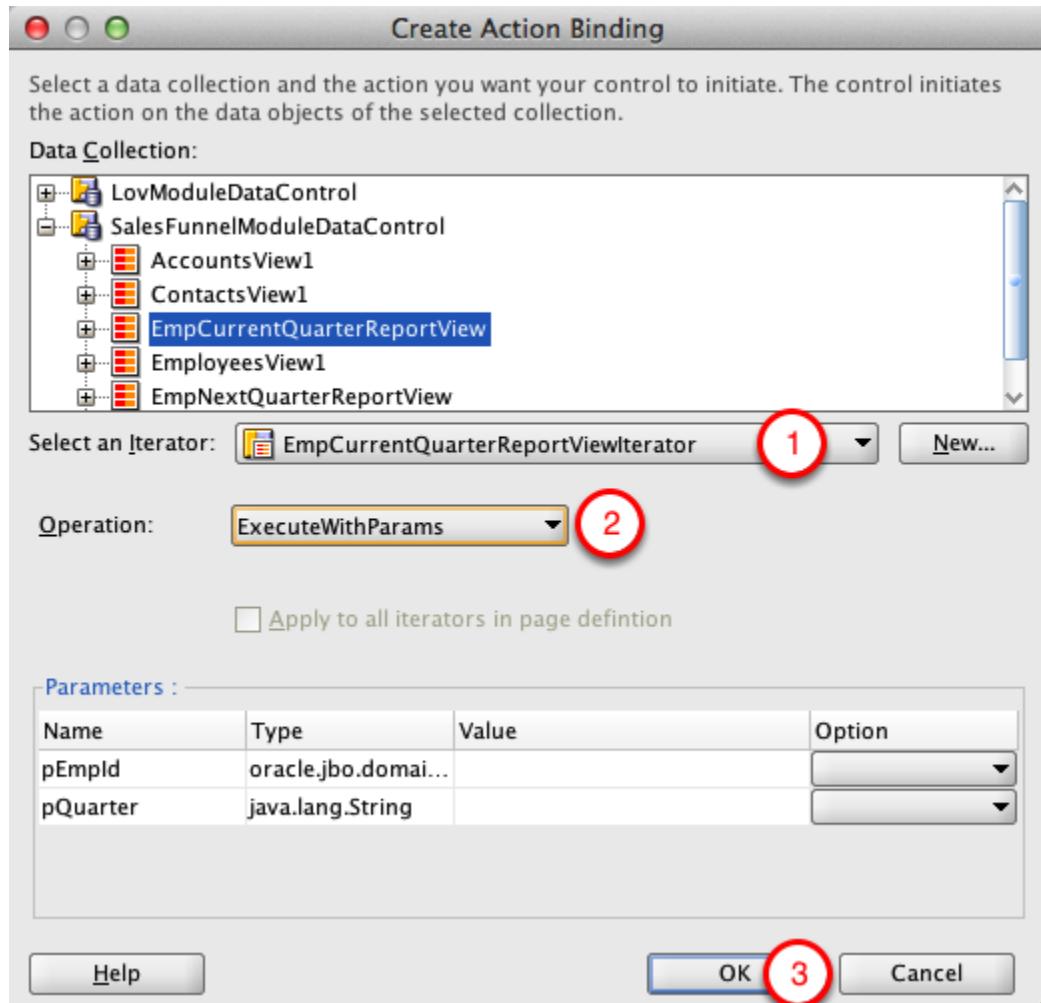
1. Open bindings tab (on hierarchFunnel.jsff)
2. Click plus icon to add new binding

Insert Item



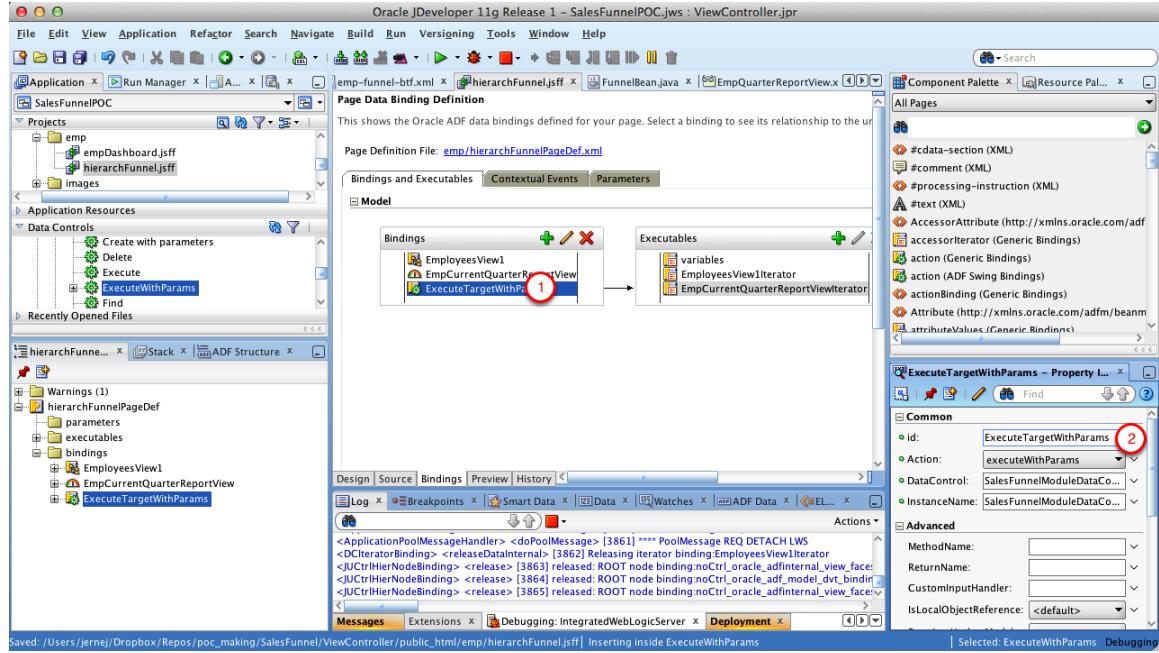
1. Select action
2. Click OK

Create Action Binding



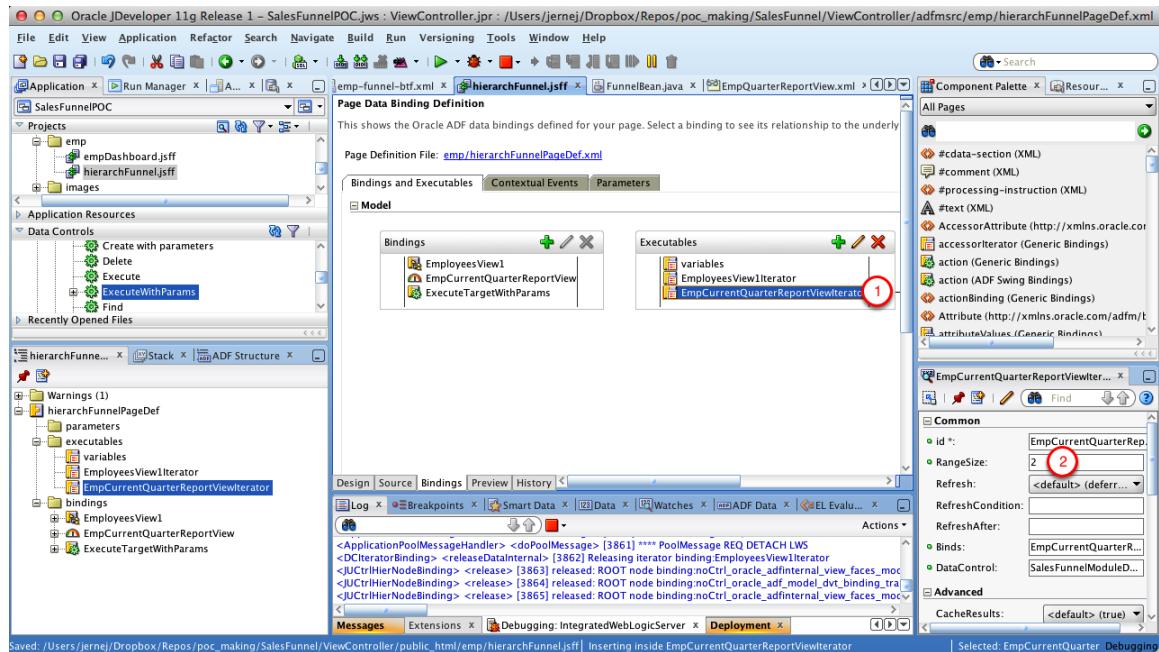
1. Select EmpCurrentQuarterReportViewIterator
2. Select ExecuteWithParams
3. Click OK

Change Action Id



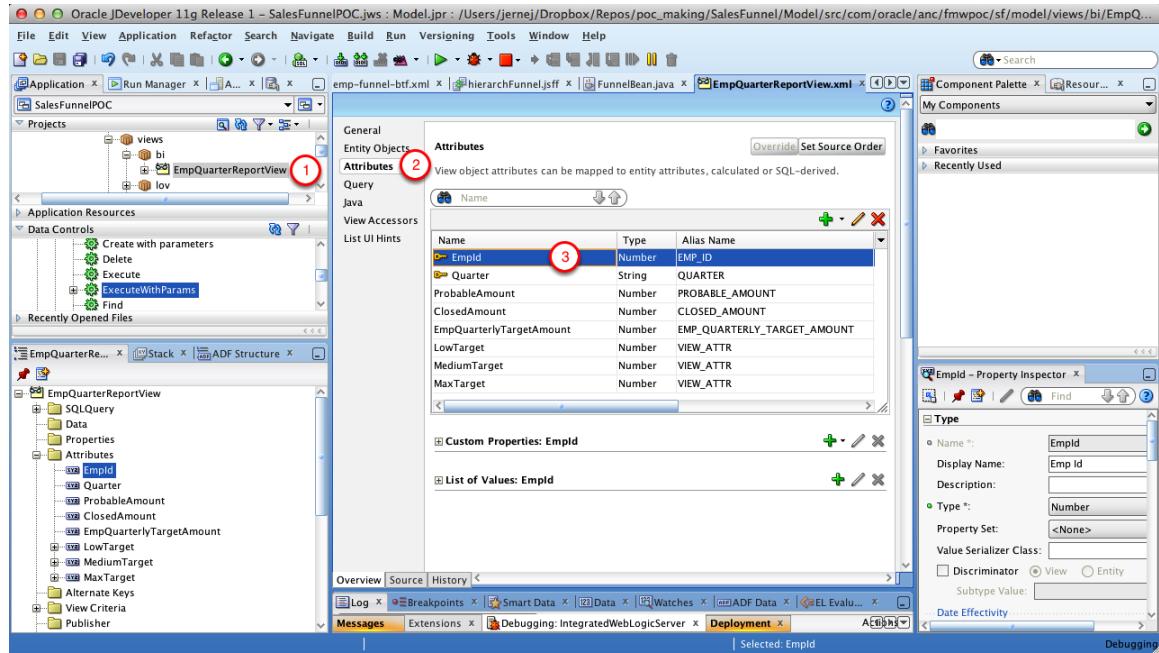
1. Select action binding
2. Set id to ExecuteTargetWithParams

Change Action Range Size



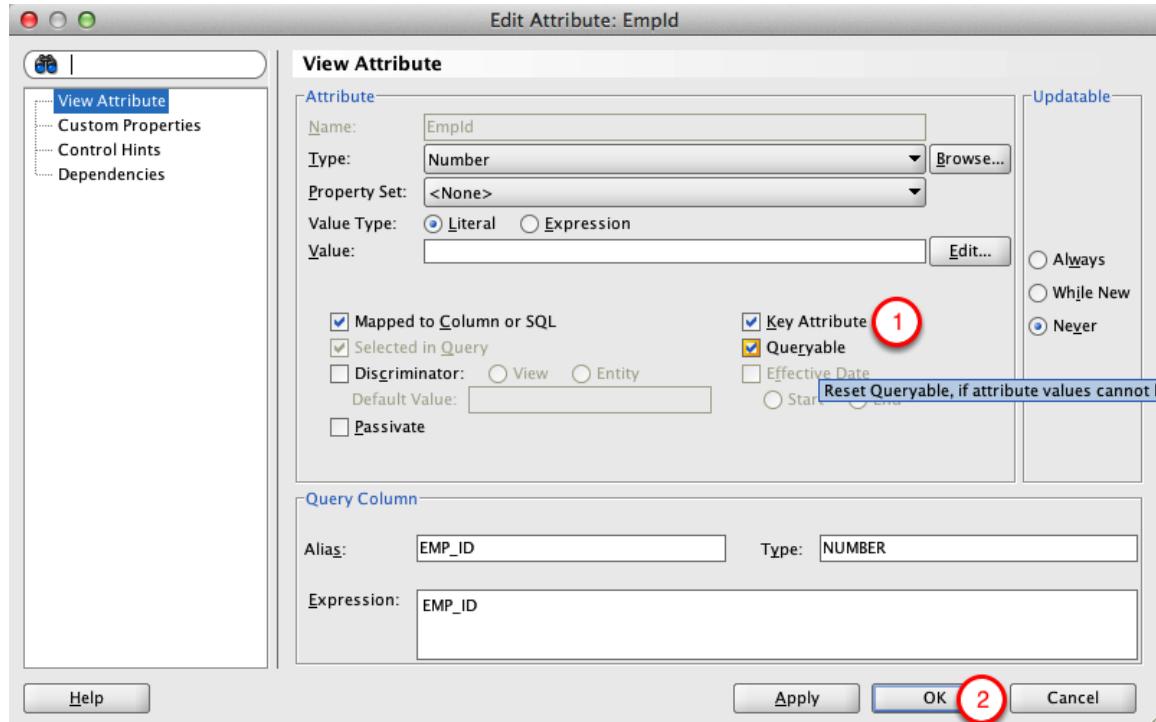
1. Select EmpCurrentQuarterReportViewIterator
2. Set RangeSize to 2

Edit EmpId Attribute



1. Open EmpQuarterReportView
2. Open Attributes tab
3. Double-click EmpId to open dialog

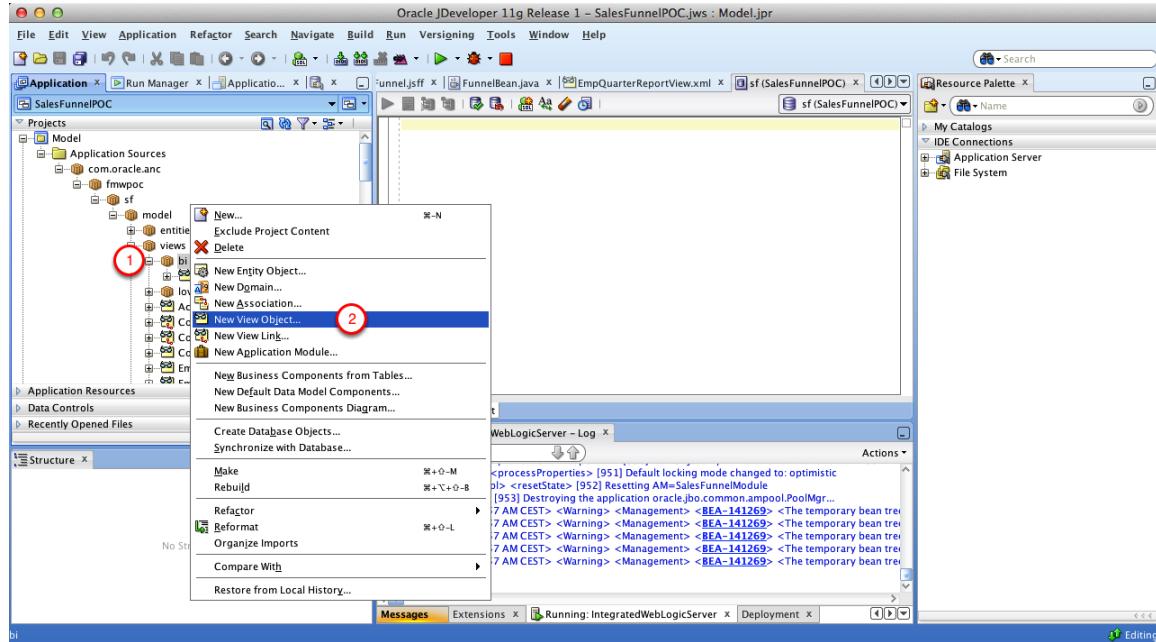
Edit Attribute: Empld



1. Set Key Attribute to checked
2. Click OK

44. Adding funnel graph

Create New View Object



1. Open Model project and right click ..model.views.bi package
2. Select New View Object from the menu

Create View Object - Step 1 of 9

Create View Object – Step 1 of 9

Name

View objects are for joining, filtering, projecting, and sorting your business data for the specific needs of a given application task.

Name	com.oracle.anc.fmw poc.sf.model.views.bi	Browse...
Name:	EmployeeFunnelView	1
Display Name:	Employee Funnel View	
Extends:		Browse...
Property Set:	<None>	

Select the data source type you want to use as the basis for this view object.

Updatable access through entity objects
 Read-only access through SQL query 2
 Rows populated programmatically, not based on a query
 Rows populated at design time (Static List)

Help < Back Next > Finish Cancel

1. Set Name to EmployeeFunnelView
2. Set data source type to "Read-only access through SQL query"

Create View Object - Step 2 of 9

Create View Object – Step 2 of 9

Query

Enter your custom SELECT statement and click Test to check its syntax. Provide the ORDER BY clause separately.

Query Statement:

```
select vw_opp_emp.* from vw_opp_emp  
WHERE  
EMP_ID = :pEmpId
```

(1) A red circle highlights the placeholder ':pEmpId' in the WHERE clause.

Query Clauses:

Order By:

Binding Style:

SQL Mode:

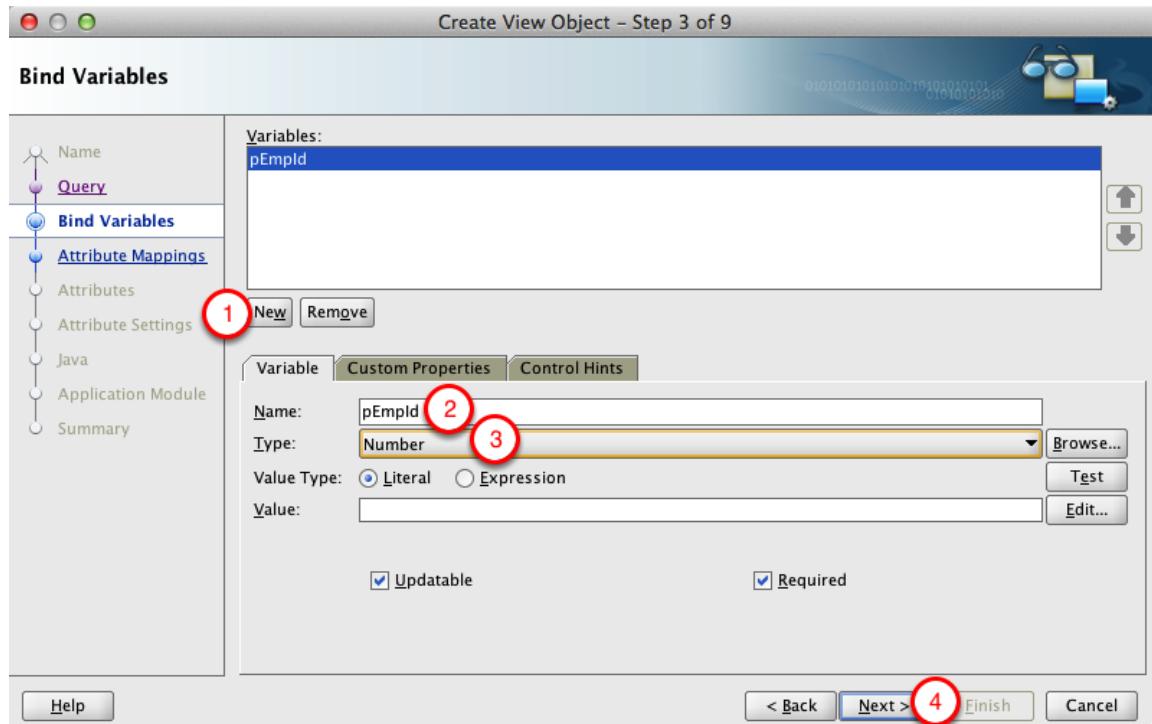
Buttons:

(2) A red circle highlights the 'Next' button.

Paste the following statement:

```
select vw_opp_emp.* from vw_opp_emp  
WHERE  
EMP_ID = :pEmpId
```

Create View Object - Step 3 of 9



1. Create New bind variable
2. Set name to pEmpId
3. Set Type to Number
4. Click Next

Create View Object - Step 4 of 9

Create View Object – Step 4 of 9

Attribute Mappings

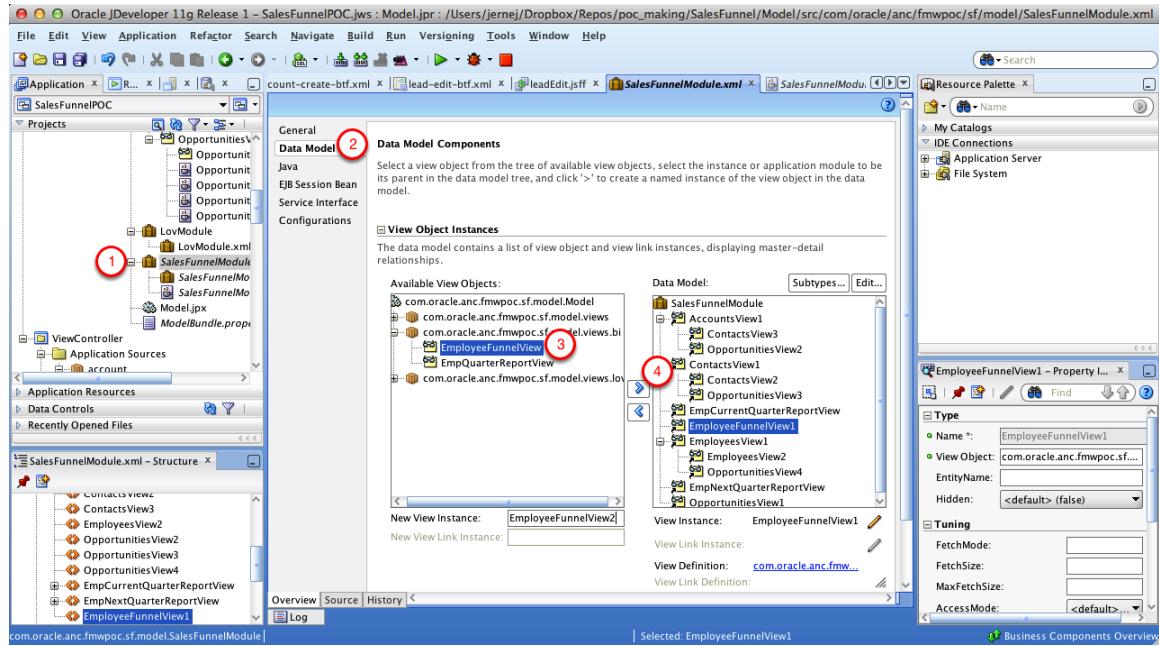
To remap a query column, click on its current view attribute and select a new one from the list. Attribute mapping only applies to Expert Mode.

Query Columns	View Attributes
EMP_ID	sqd EmpId
STAGE	sqd Stage
OPP_ESTIMATED_AMOUNT	sqd OppEstimatedAmount
PROBABLE_AMOUNT	sqd ProbableAmount
STAGE_TARGET	sqd StageTarget

Help < Back **Next >** **Finish** 1 Cancel

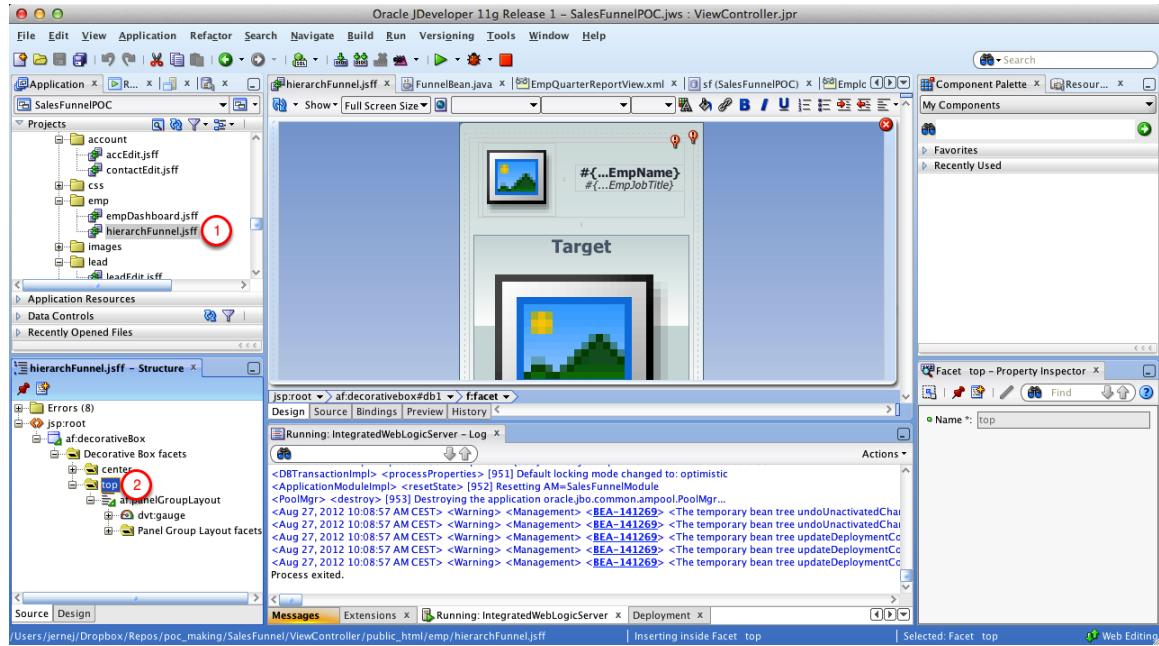
1. Click Finish

Add EmployeeFunnelView to SalesFunnelModule



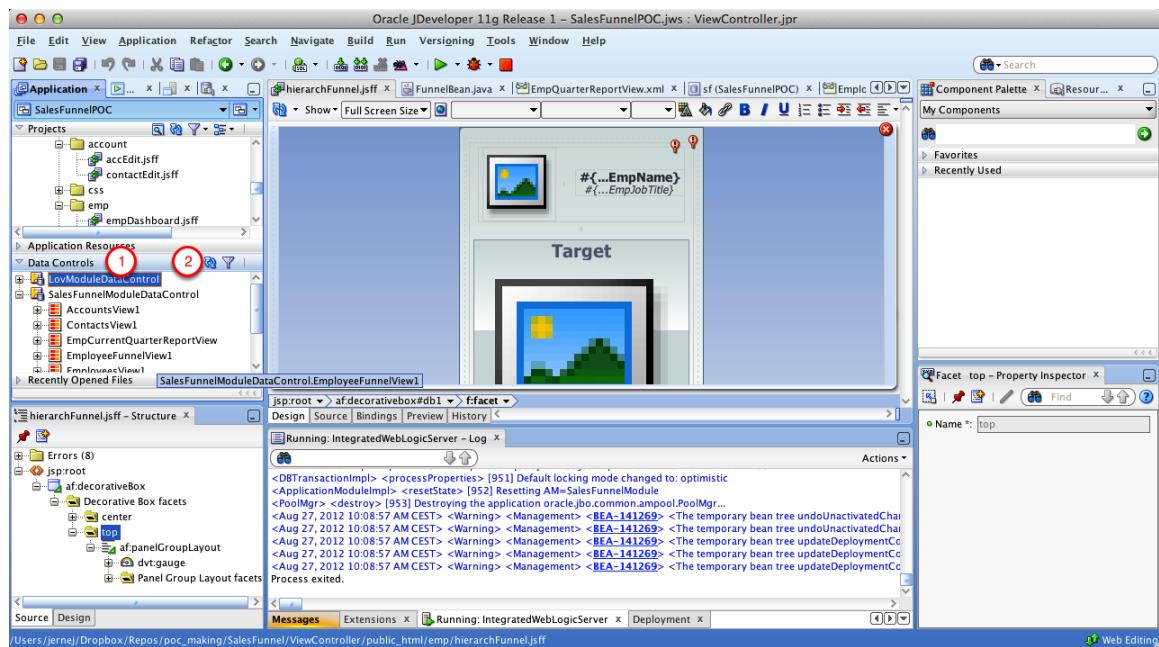
1. Open SalesFunnelModule
2. Open Data Model tab
3. Select EmployeeFunnelView
4. Click on the right Arrow

Open Hierarchy Page



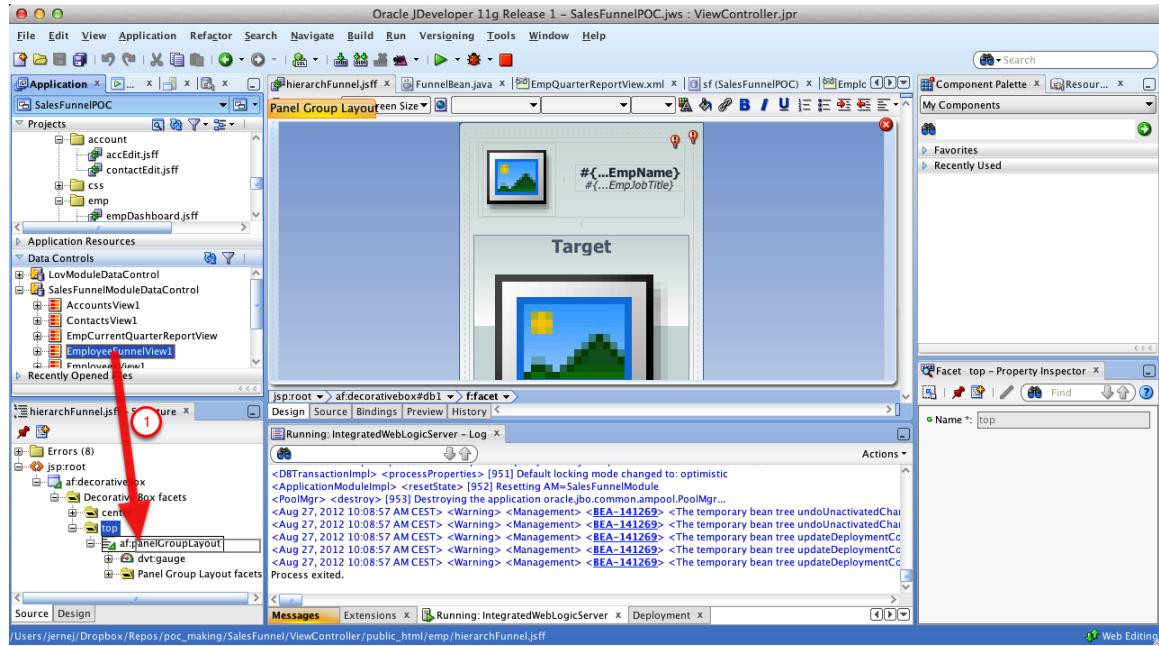
1. Open hierarchFunnel.jsff
2. Expand top facet in the structure window

Refresh Data Controls



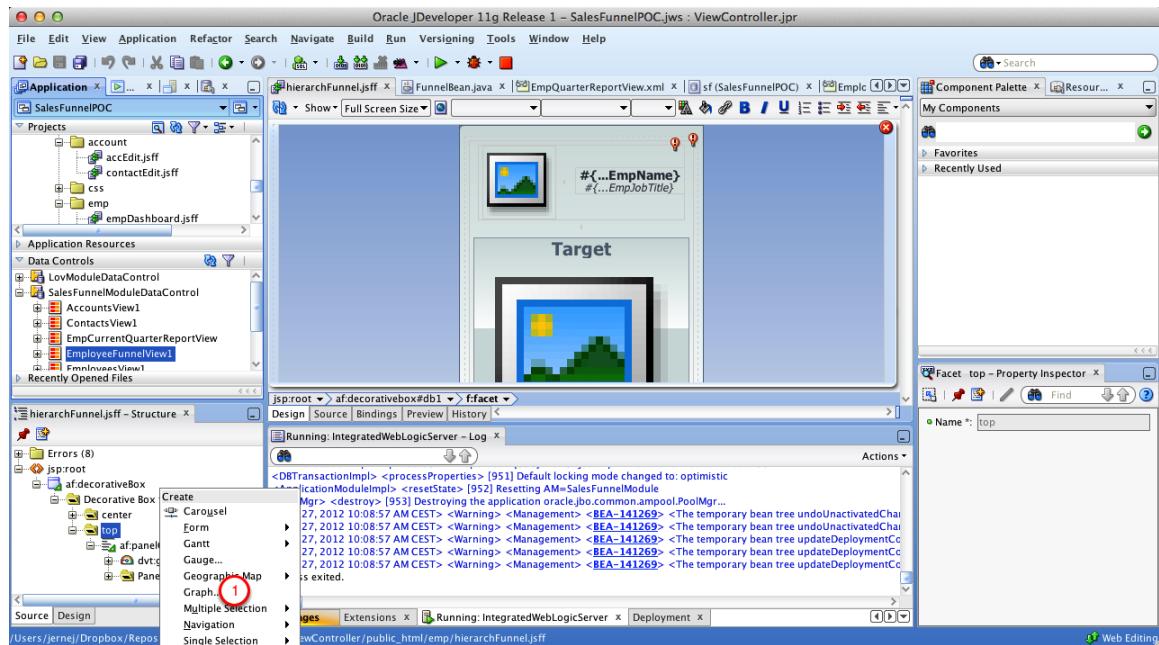
1. Expand Data Controls
2. Click Refresh

Add Funnel Graph



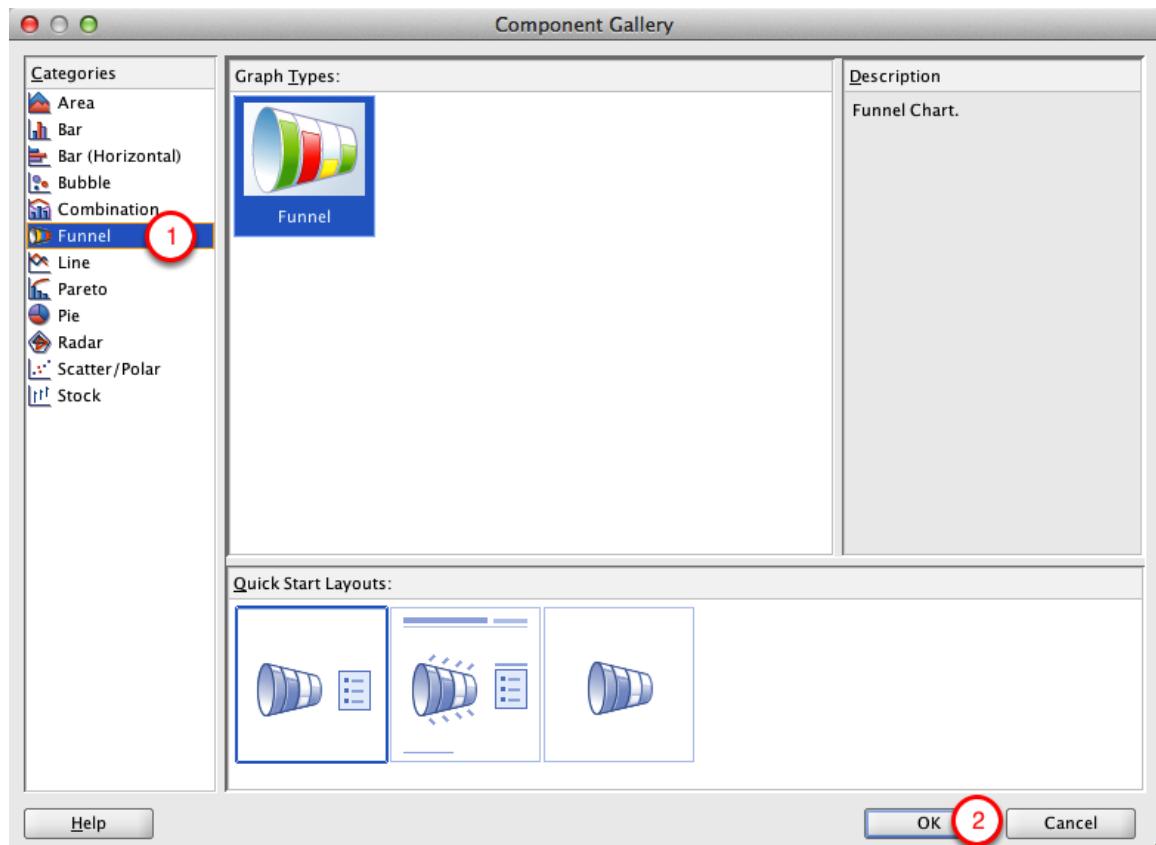
1. Drag and drop EmployeeFunnelView on panelGroupLayout

New Graph



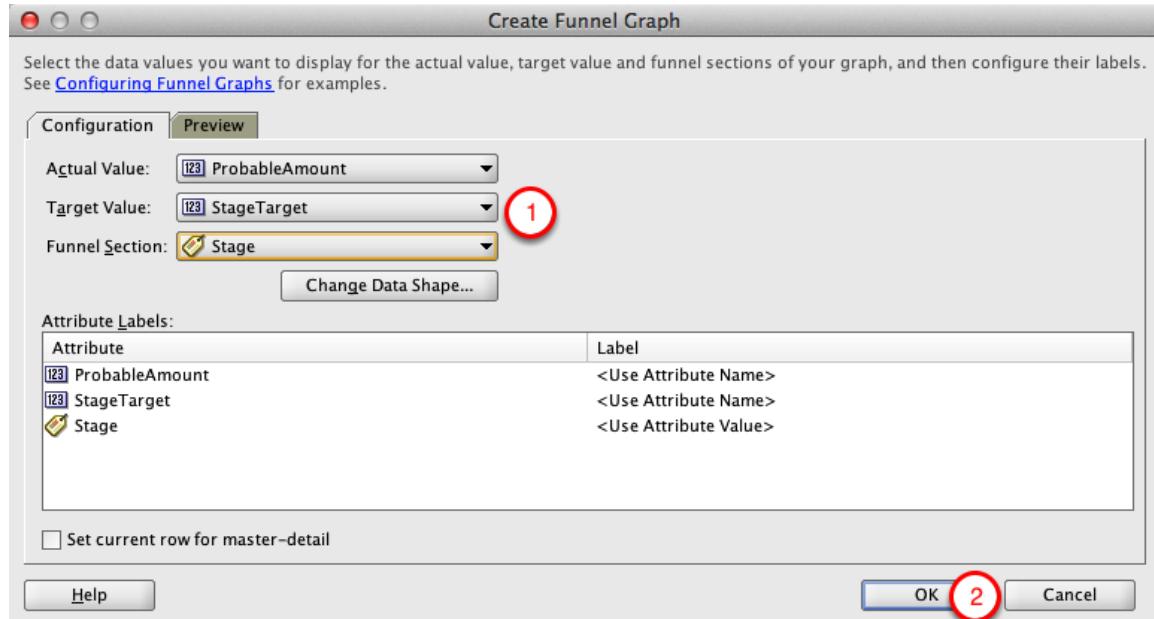
1. Select Graph from the menu

Component Gallery



1. Select Funnel Chart
2. Click OK

Create Funnel Graph

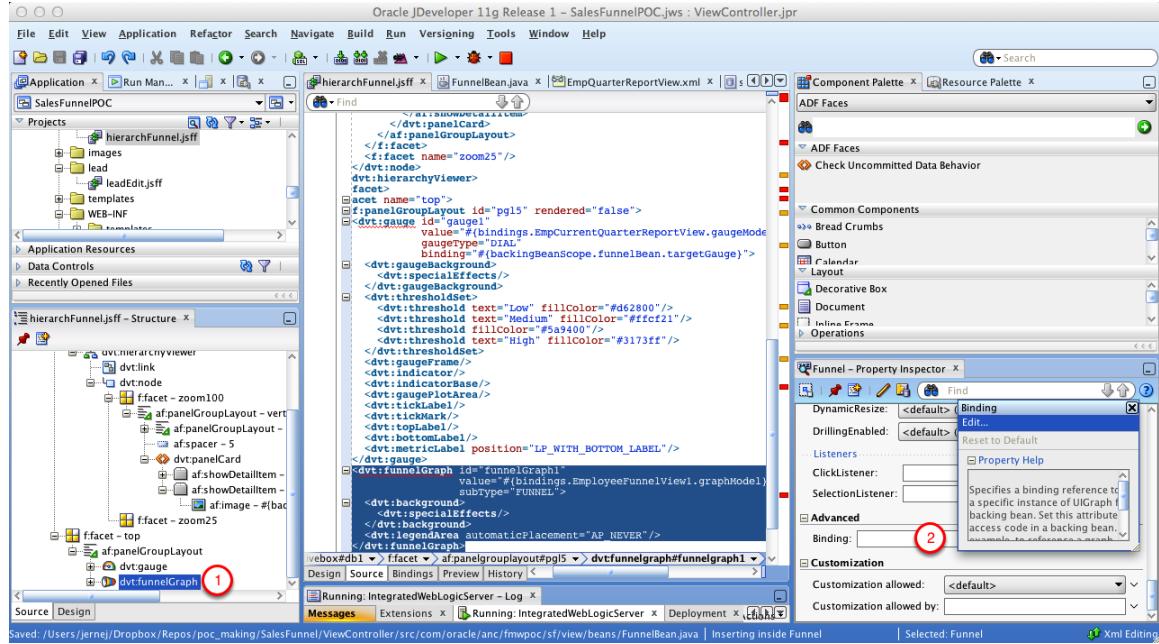


1. Set attributes:

- Actual Value: ProbableAmount
- Target Value: StageTarget
- Funnel Section: Stage

2. Click OK

Add Funnel Binding



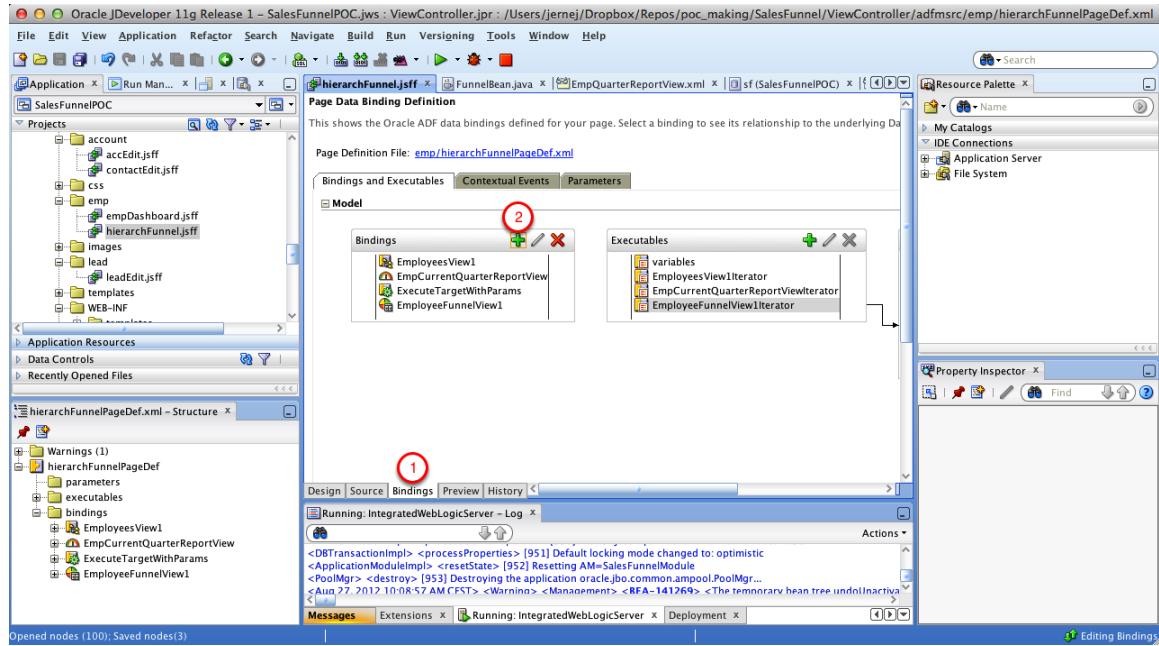
1. Select funnelGraph
2. Click small down arrow next to Binding property in property inspector and click Edit

Edit Property: Binding



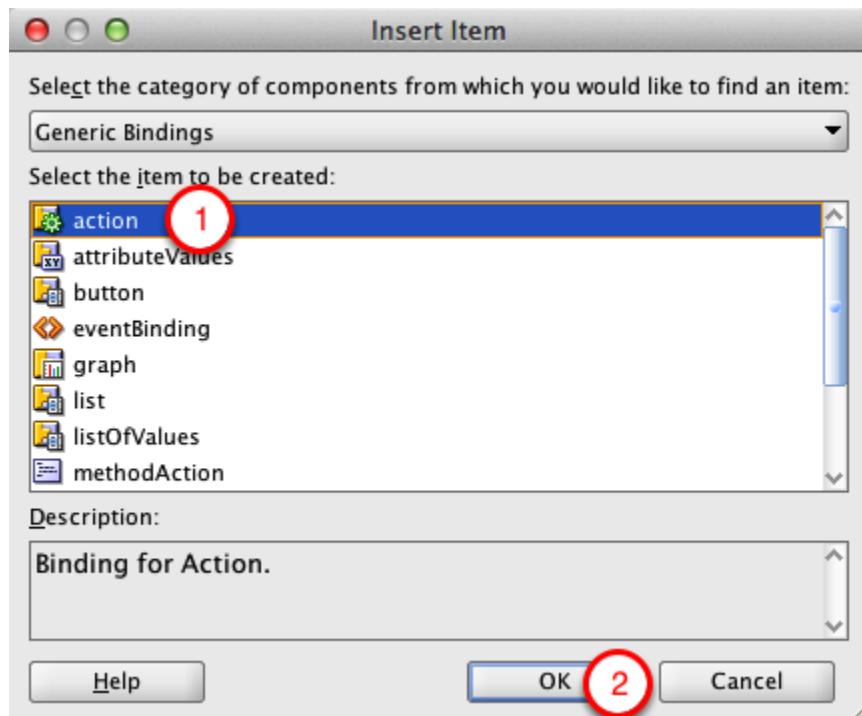
1. Set Managed Bean to funnelBean
2. Click New to create new Property, name it funnelGraph
3. Click OK

Add Action Binding



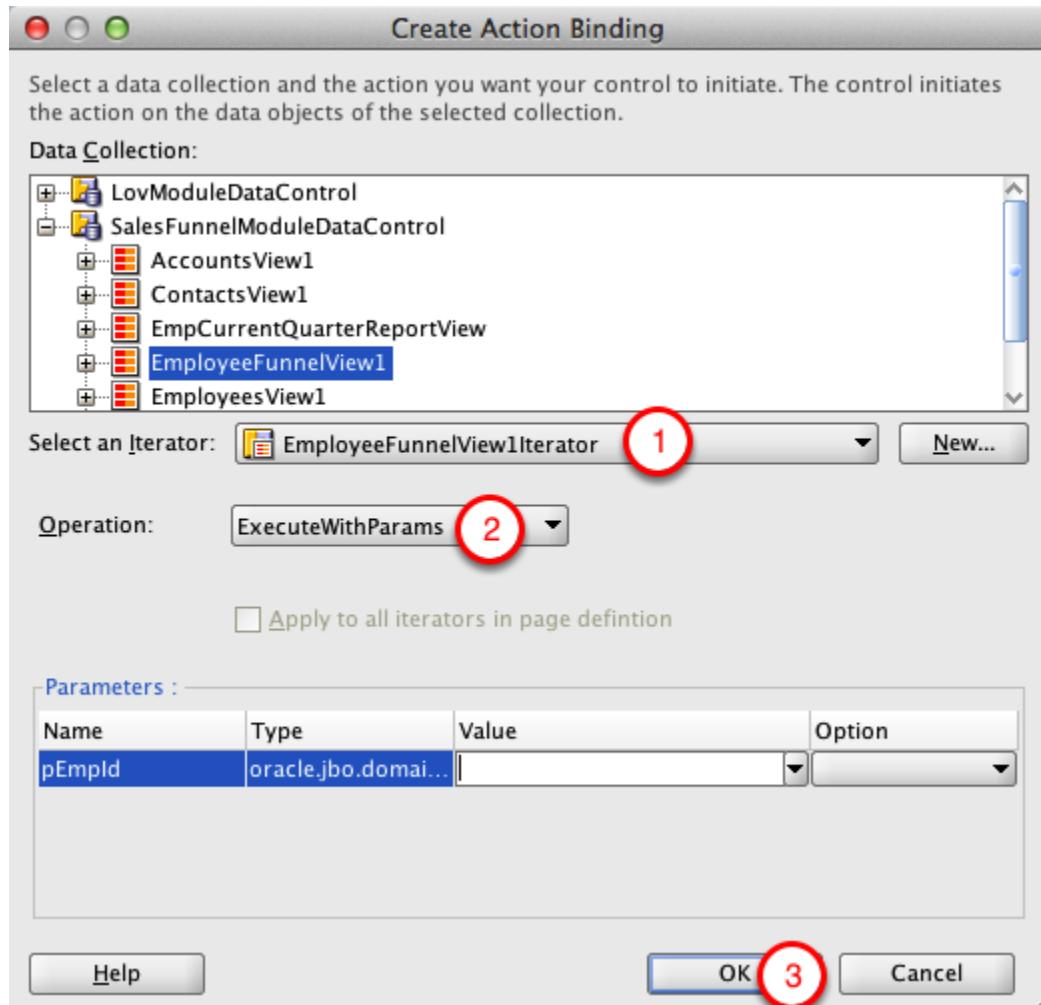
1. Open Bindings Tab
2. Click plus to create new binding

Insert Item



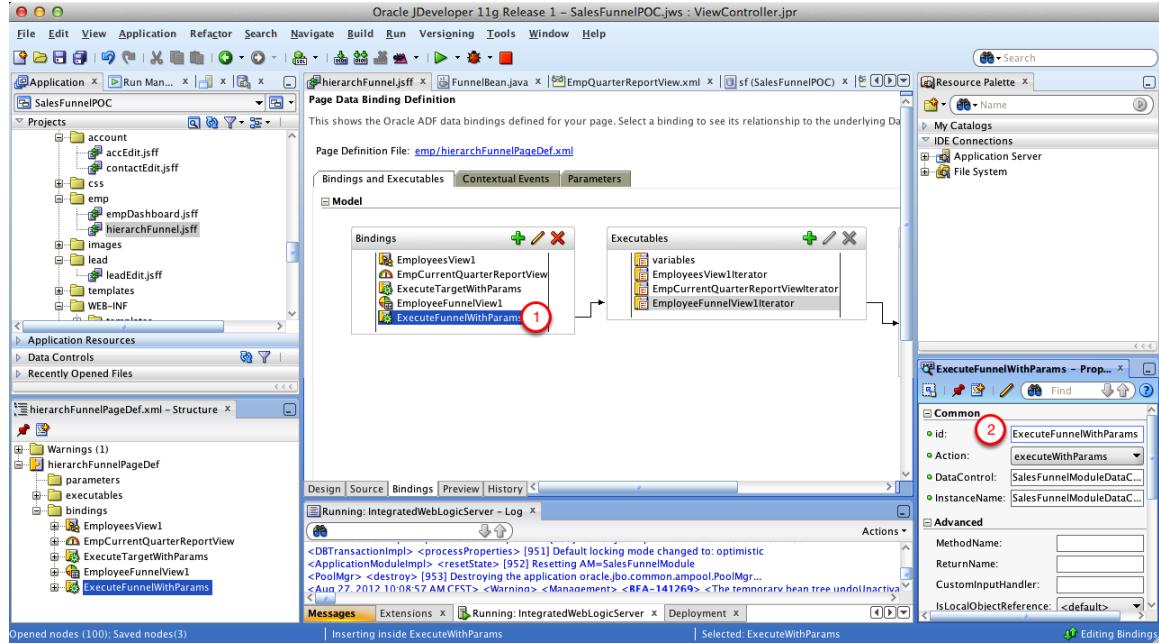
1. Select action
2. Click OK

Create Action Binding



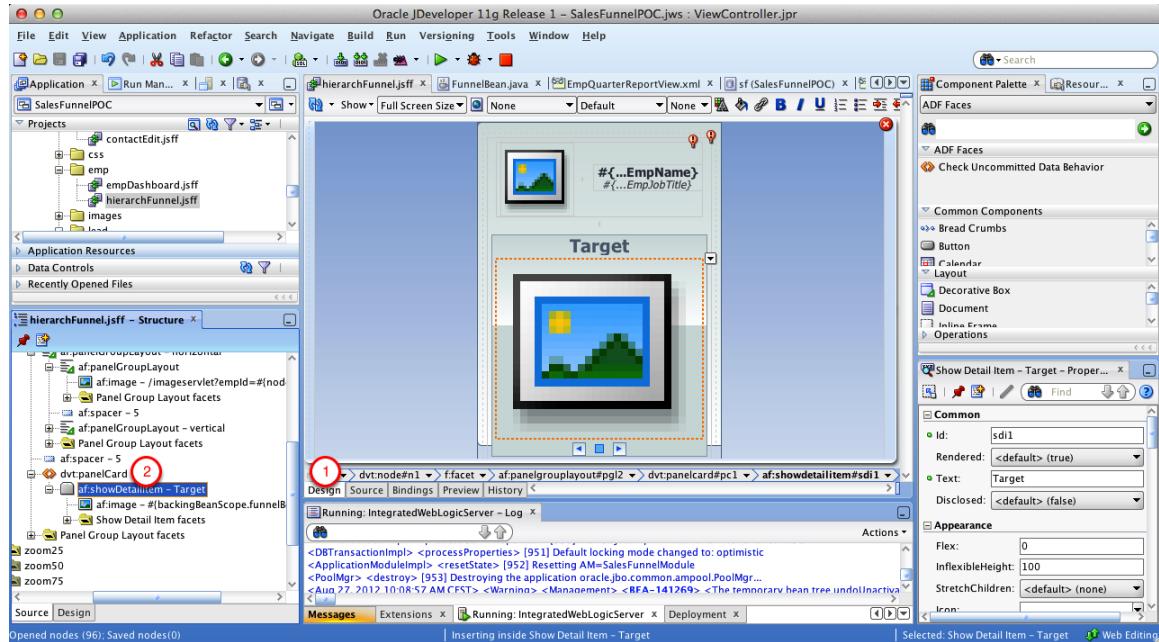
1. Select EmployeeFunnelView1Iterator
2. Select OperationExecuteWithParams
3. Click OK

Change Action Id



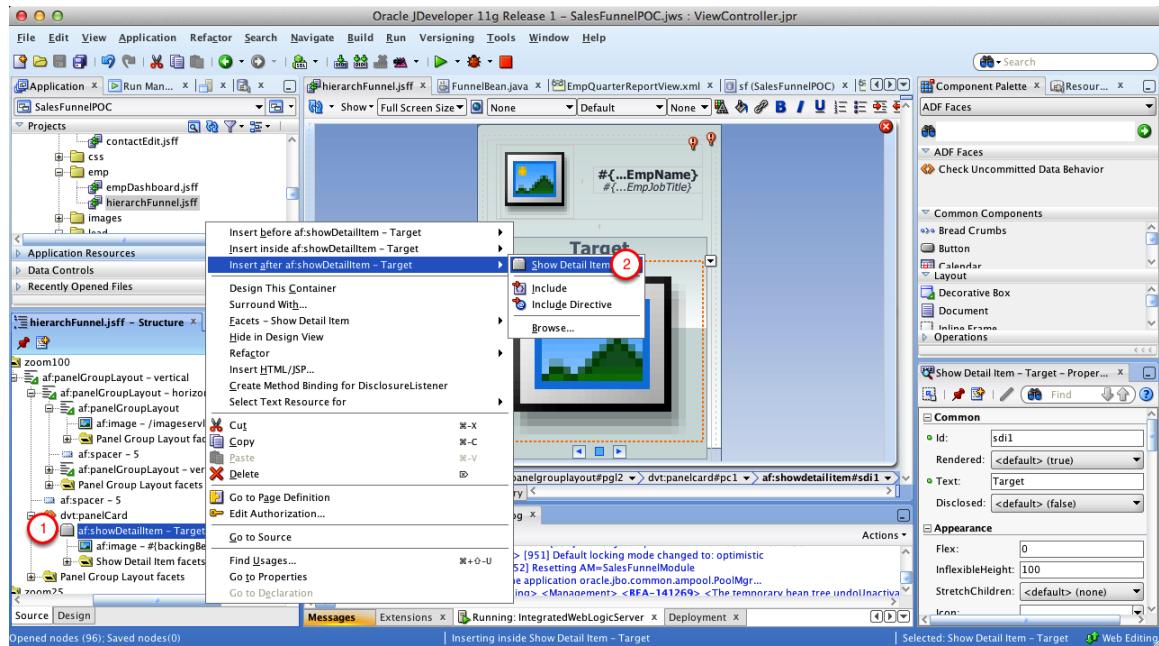
1. Select the action
2. Set id to ExecuteFunnelWithParams

Change Panel Card



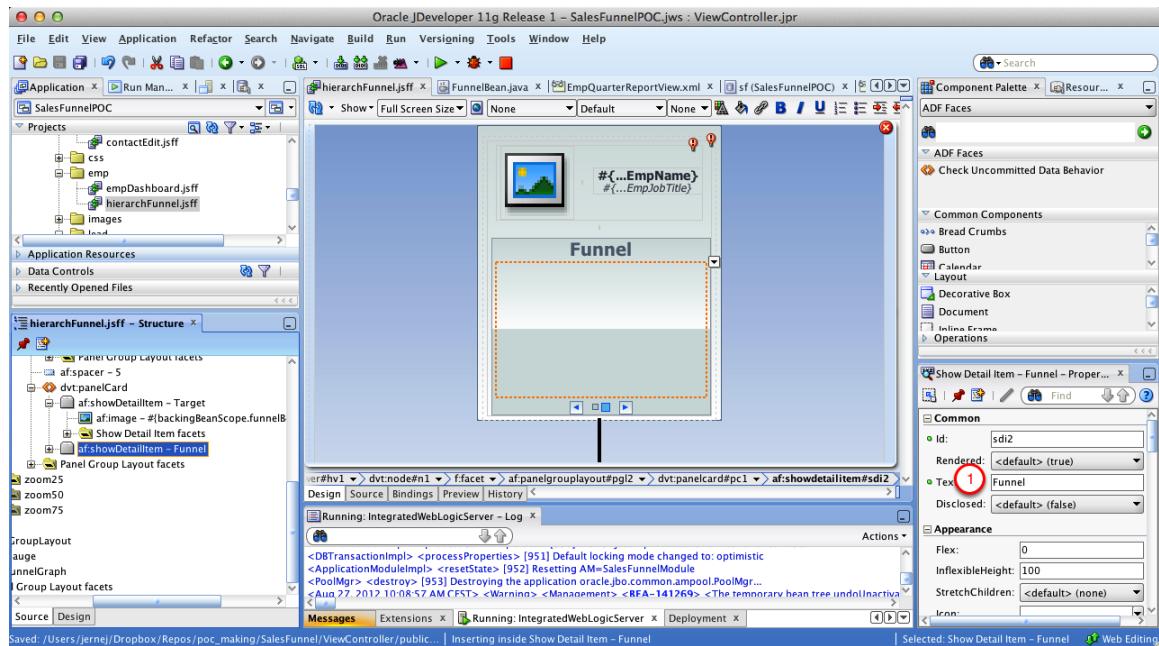
1. Open Design tab
2. Find dvt:panelCard in the structure

Add Card



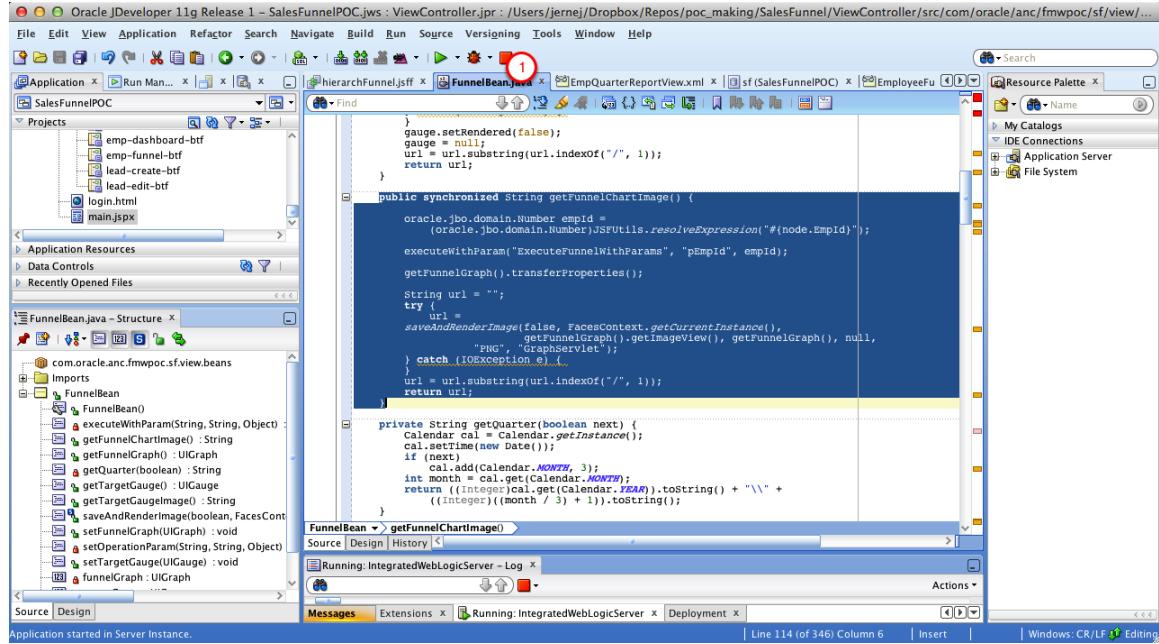
1. Right-click showDetailItem
2. In the popup menu, select "Insert after..." and then click Show Detail Item

Change Card Properties



1. Set Text property to Funnel

Implement getFunnelChartImage



1. Open FunnelBean.java
2. Paste the below code after the getTargetGaugeImage method

```

public synchronized String getFunnelChartImage() {

    oracle.jbo.domain.Number empId =
        (oracle.jbo.domain.Number)JSFUtils.resolveExpression("#{node.EmpId}");

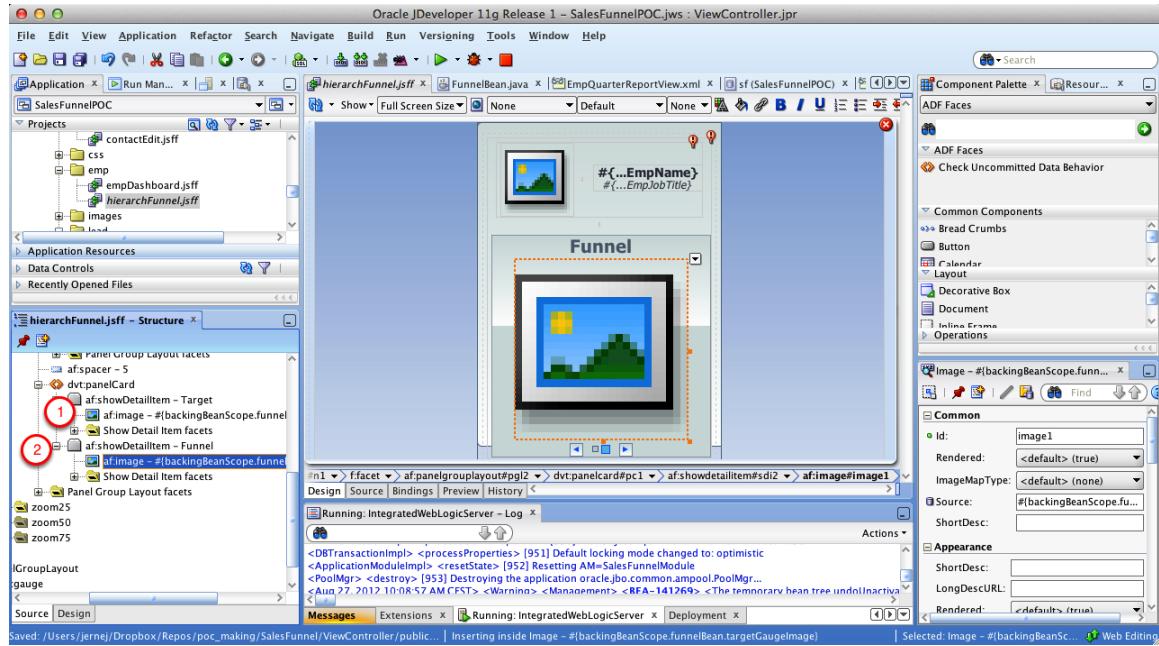
    executeWithParam("ExecuteFunnelWithParams", "pEmpId", empId);

    getFunnelGraph().transferProperties();

    String url = "";
    try {
        url =
            saveAndRenderImage(false, FacesContext.getCurrentInstance(),
                getFunnelGraph().getImageView(), getFunnelGraph(), null,
                "PNG", "GraphServlet");
    } catch (IOException e) {
    }
    url = url.substring(url.indexOf("/", 1));
    return url;
}

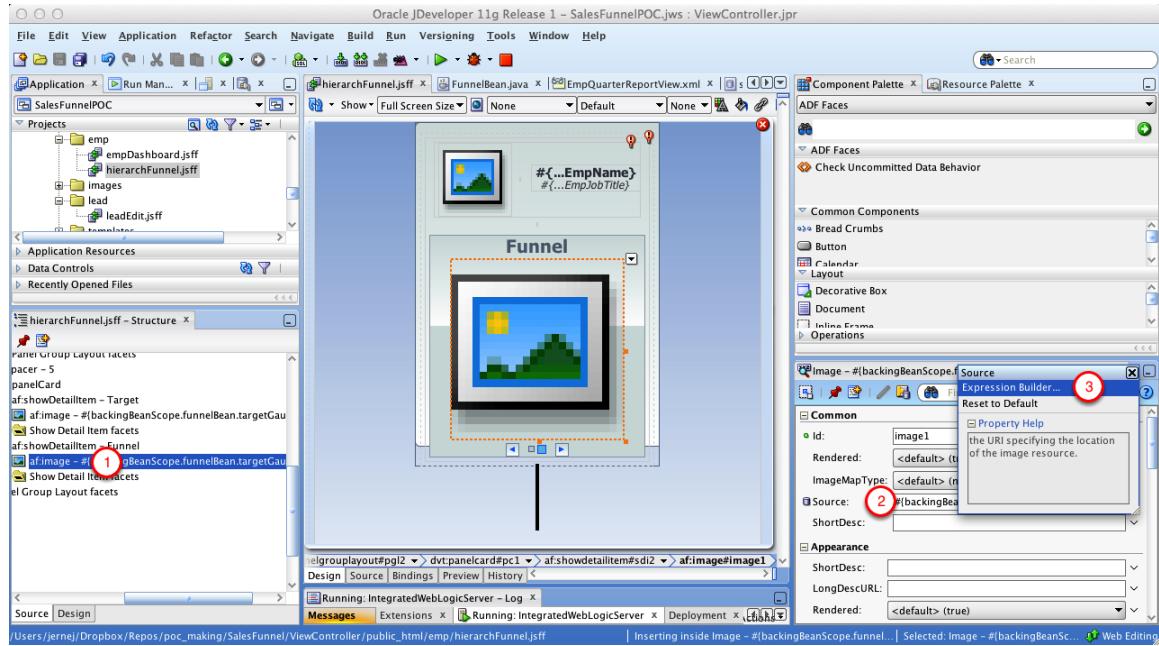
```

Copy Image



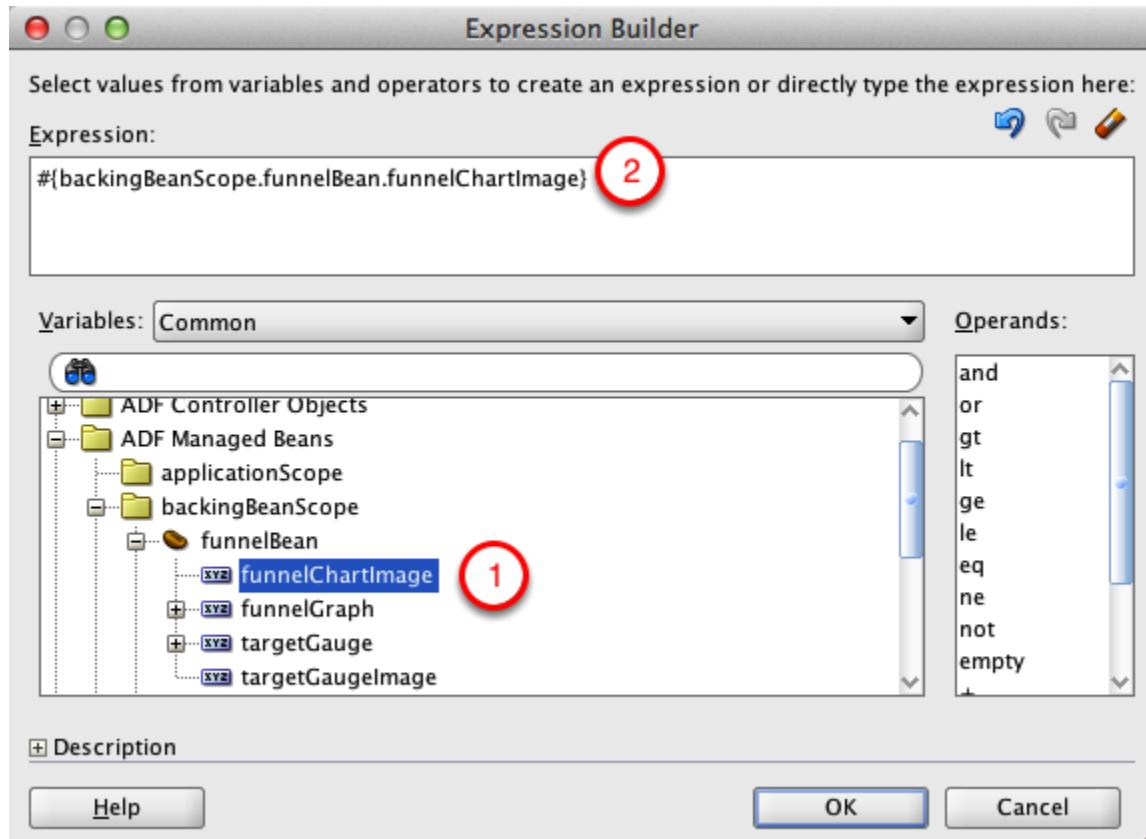
1. Select af:image in the Target showDetailItem and copy it (ctrl-c)
2. Select showDetailItem Funnel and paste the image (ctrl-v)

Change Image Source



1. Select the pasted image
2. Click on the little down arrow next to Source property in property inspector
3. Select Expression Builder

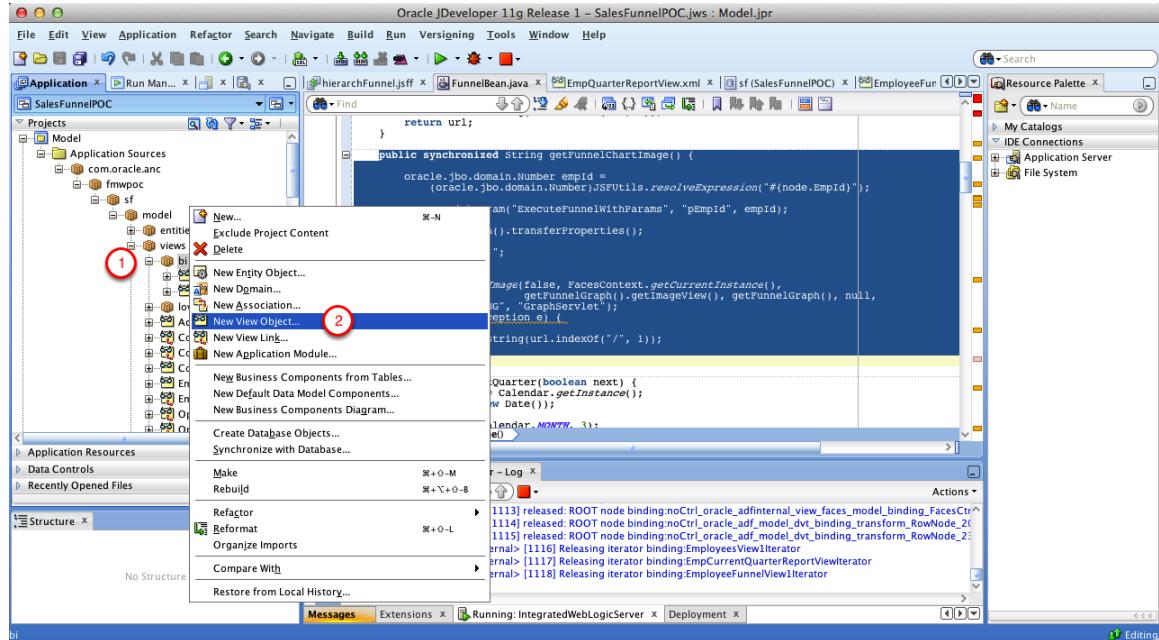
Expression Builder



1. Expand ADF Managed Beans -> backingBeanScope -> funnelBean and select funnelChartImage
2. Make sure the expression is "#{backingBeanScope.funnelBean.funnelChartImage}"

45. Adding report chart

Create New View Object



1. Right-click on ...model.views.bi
2. Select New View Object

Create View Object - Step 1 of 9

Create View Object – Step 1 of 9

Name

View objects are for joining, filtering, projecting, and sorting your business data for the specific needs of a given application task.

Name	<input type="text" value="EmployeeQuarterReportView"/> 1
Entity Objects	<input type="text" value="com.oracle.anc.fmw poc.sf.model.views.bi"/> Browse...
Attributes	
Attribute Settings	
Query	
Bind Variables	
Java	
Application Module	
Summary	

Select the data source type you want to use as the basis for this view object.

Updatable access through entity objects
 Read-only access through SQL query **2**
 Rows populated programmatically, not based on a query
 Rows populated at design time (Static List)

Help **Next > 3** **Finish** **Cancel**

1. Set Name to EmployeeQuarterReportView
2. Set data source type to "Read-only access through SQL query"
3. Click Next

Create View Object - Step 2 of 9

Create View Object – Step 2 of 9

Query

Enter your custom SELECT statement and click Test to check its syntax. Provide the ORDER BY clause separately.

Query Statement:

```
SELECT
    VW_EMP_QUARTER_REPORT.EMP_ID EMP_ID,
    VW_EMP_QUARTER_REPORT.QUARTER QUARTER,
    VW_EMP_QUARTER_REPORT.PROBABLE_AMOUNT PROBABLE_AMOUNT,
    VW_EMP_QUARTER_REPORT.CLOSED_AMOUNT CLOSED_AMOUNT,
    VW_EMP_QUARTER_REPORT.QUARTERLY_TARGET_AMOUNT EMP_QUARTERLY_TARGET_AMOUNT
FROM
    VW_EMP_QUARTER_REPORT
WHERE
    VW_EMP_QUARTER_REPORT.EMP_ID = :pEmpId
```

Query Clauses:

Order By:

Binding Style: Oracle Named

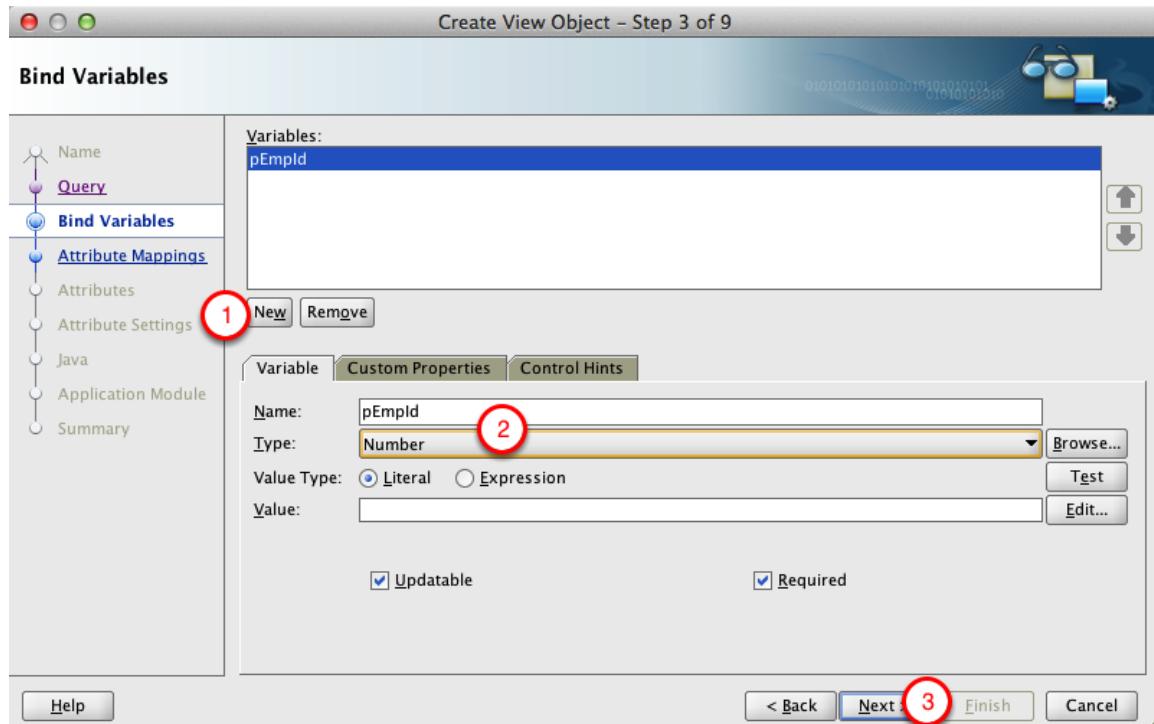
SQL Mode: Expert

Buttons: Help

1. Copy-paste the below SQL into Query Statement
2. Click Next

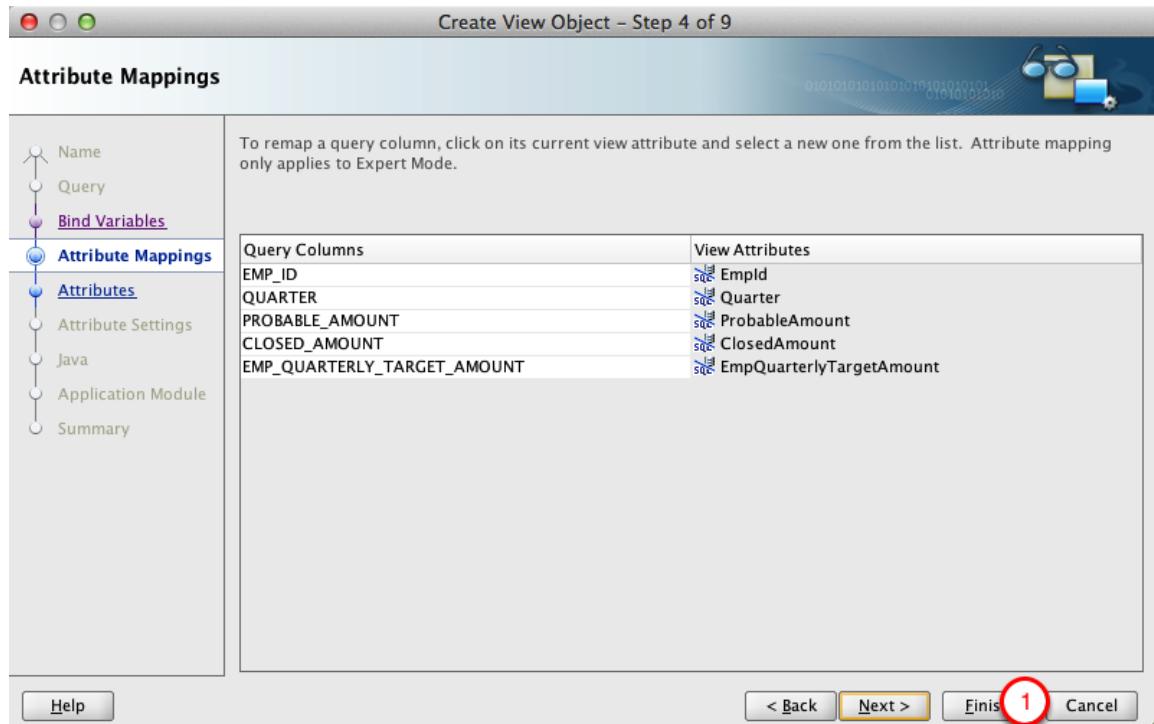
```
SELECT
    VW_EMP_QUARTER_REPORT.EMP_ID EMP_ID,
    VW_EMP_QUARTER_REPORT.QUARTER QUARTER,
    VW_EMP_QUARTER_REPORT.PROBABLE_AMOUNT PROBABLE_AMOUNT,
    VW_EMP_QUARTER_REPORT.CLOSED_AMOUNT CLOSED_AMOUNT,
    VW_EMP_QUARTER_REPORT.QUARTERLY_TARGET_AMOUNT EMP_QUARTERLY_TARGET_AMOUNT
FROM
    VW_EMP_QUARTER_REPORT
WHERE
    VW_EMP_QUARTER_REPORT.EMP_ID = :pEmpId
```

Create View Object - Step 3 of 9



1. Create new bind variable
2. Set Name to pEmpId and Type to Number
3. Click Next

Create View Object - Step 4 of 9

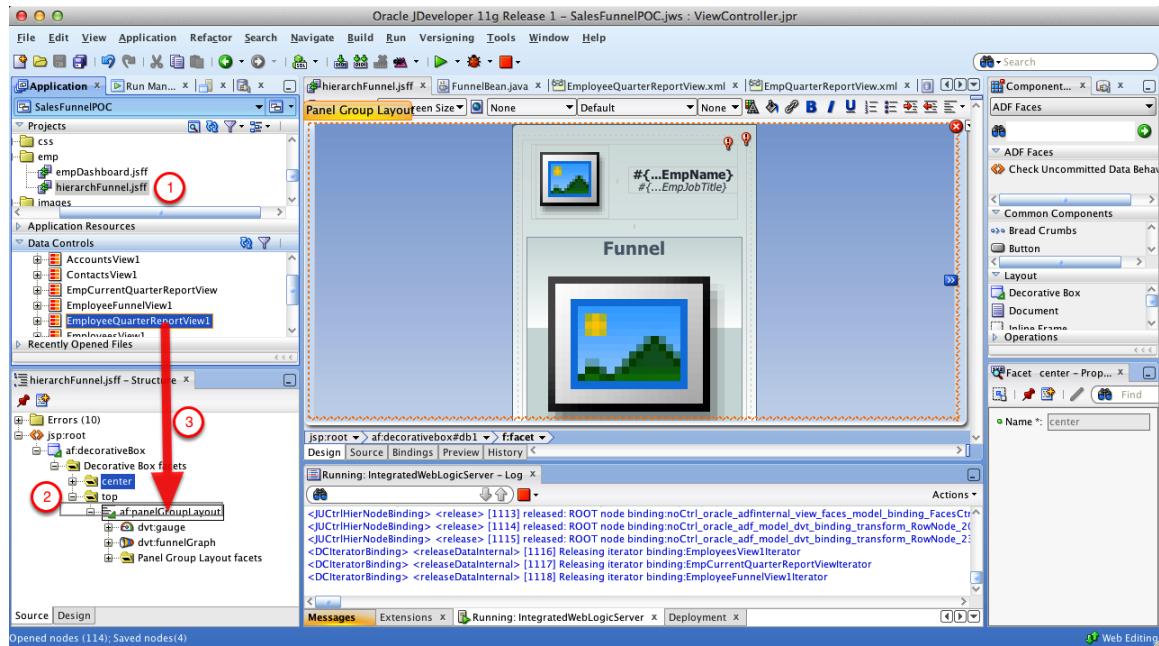


1. Click Finish

Exercise: Add EmployeeQuarterReportView to SalesFunnelModule

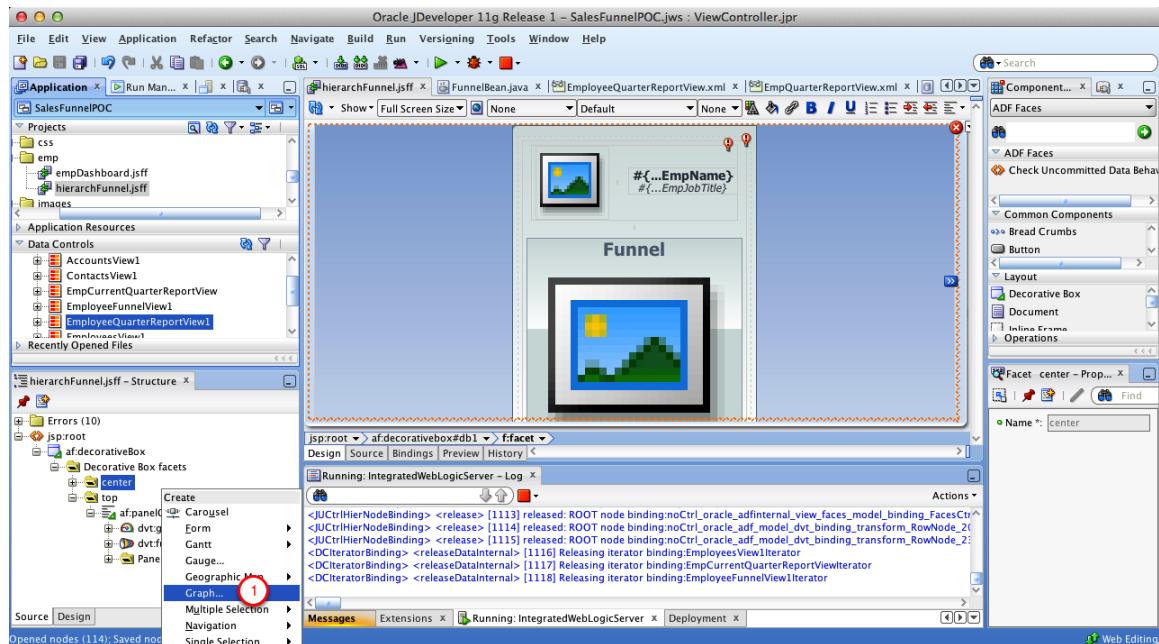
Using previous lessons as a reference, add EmployeeQuarterReportView to SalesFunnelModule

Add Combo Graph



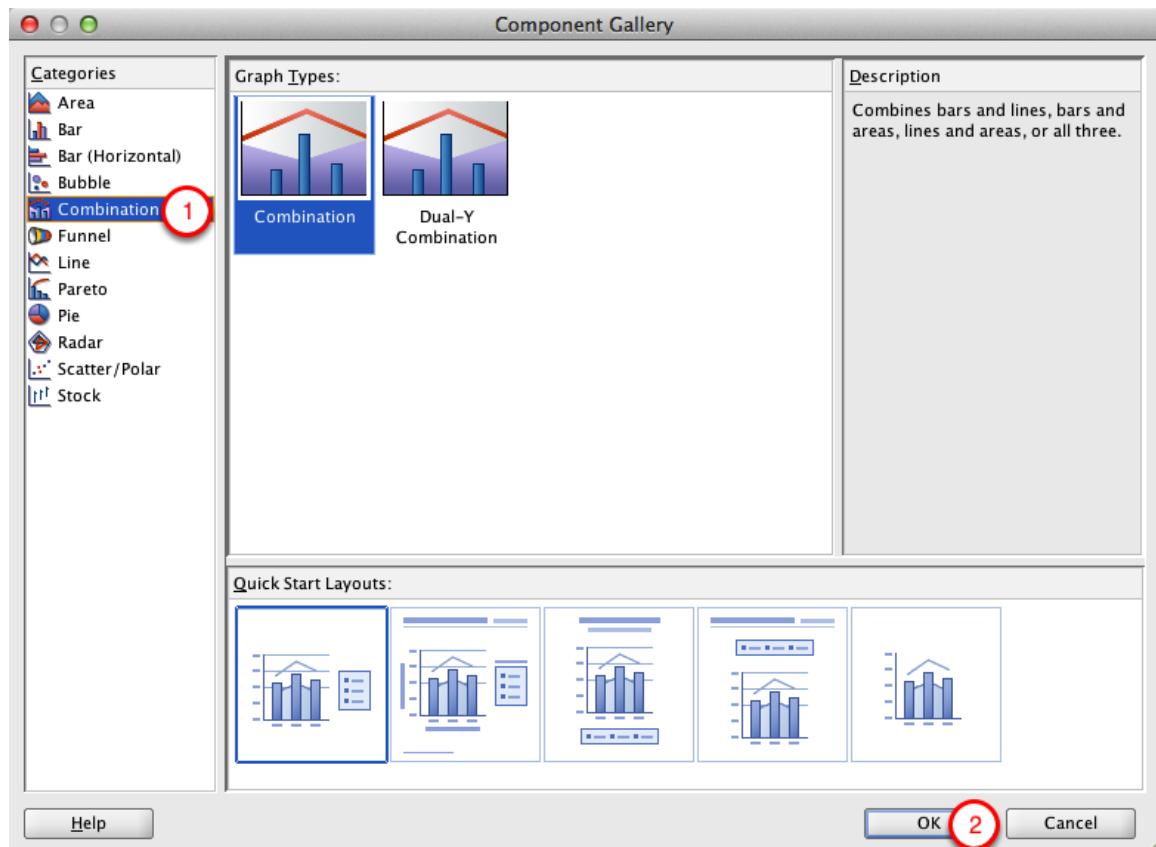
1. Open hierarchyFunnel.jsff
2. Expand top facet
3. Drag and drop EmployeeQuarterReportView1 inside panelGroupLayout

Add Graph



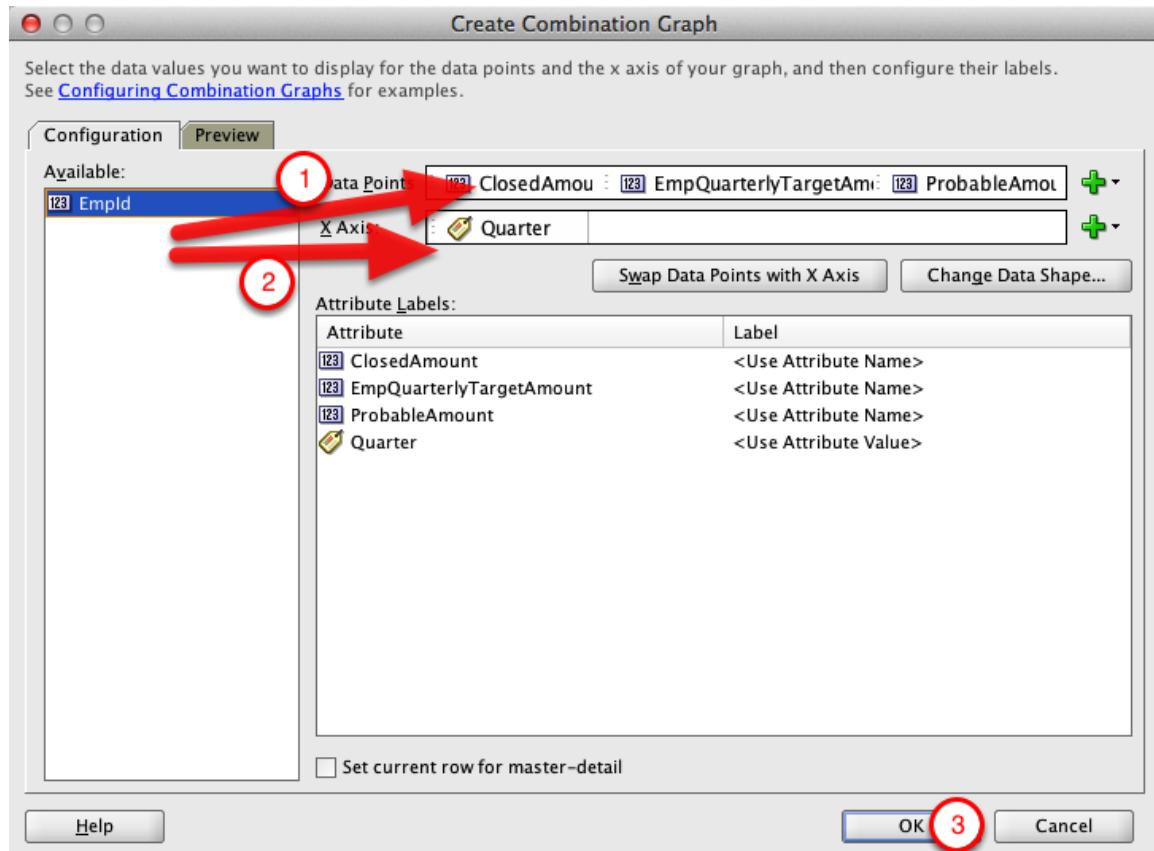
1. Select Graph in the menu

Component Gallery



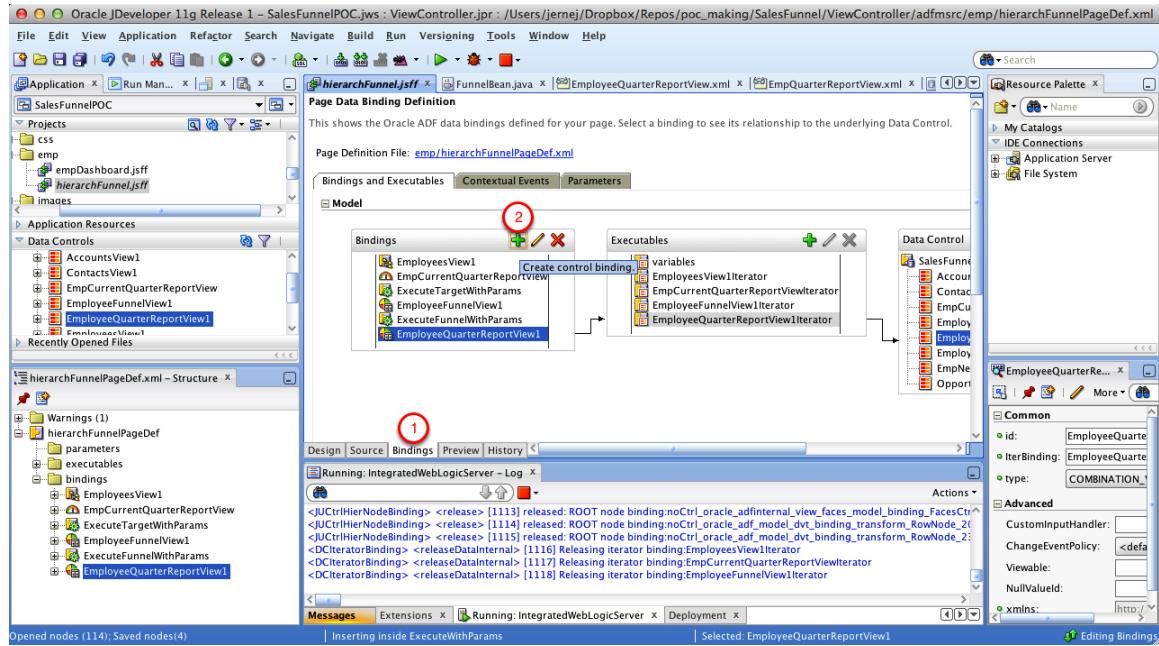
1. Select Combination Graph
2. Click OK

Create Combination Graph



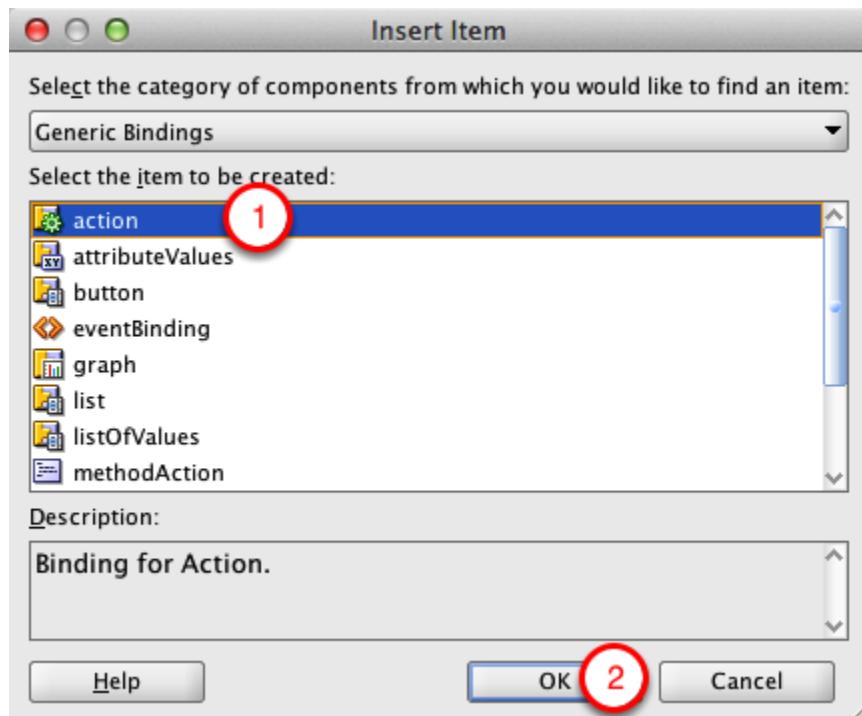
1. Drag and drop ClosedAmount, EmpQuarterlyTargetAmount and ProbableAmount attributes to data point section
2. Drag and drop Quarter attribute to X Axis
3. Click OK

Add Action Binding



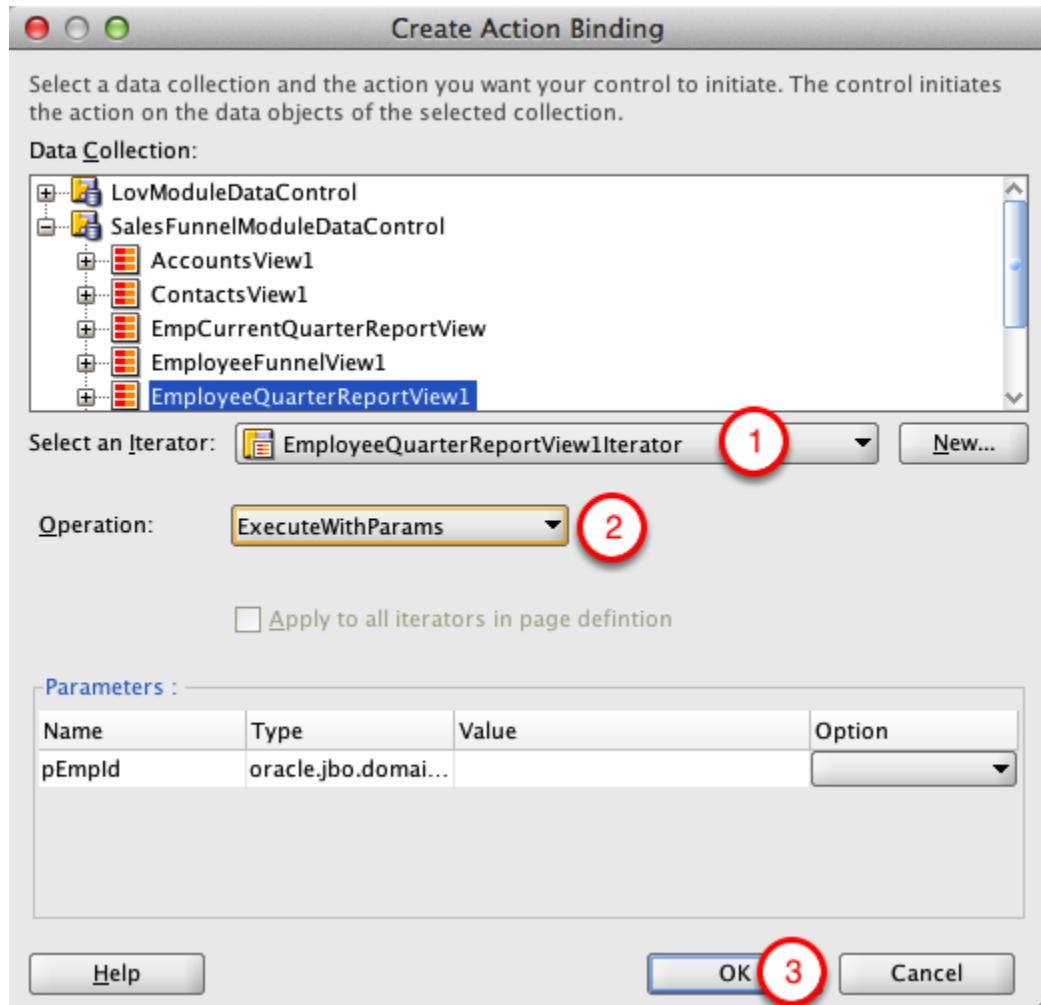
1. Open Bindings
2. Click plus

Insert Item



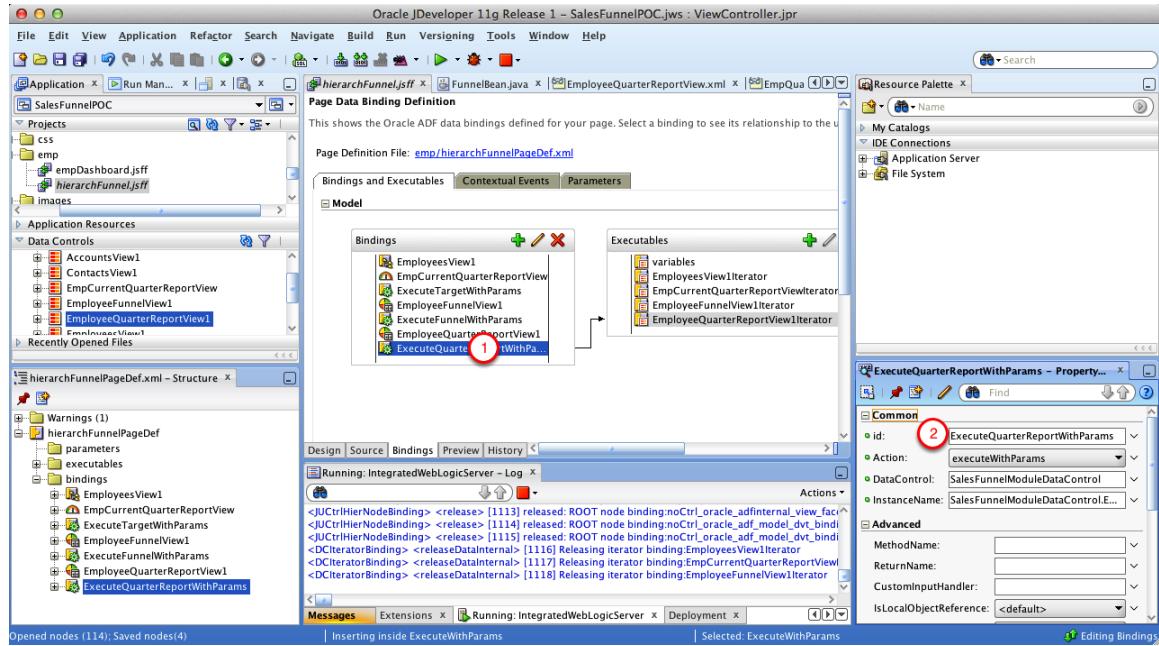
1. Select action
2. Click OK

Create Action Binding



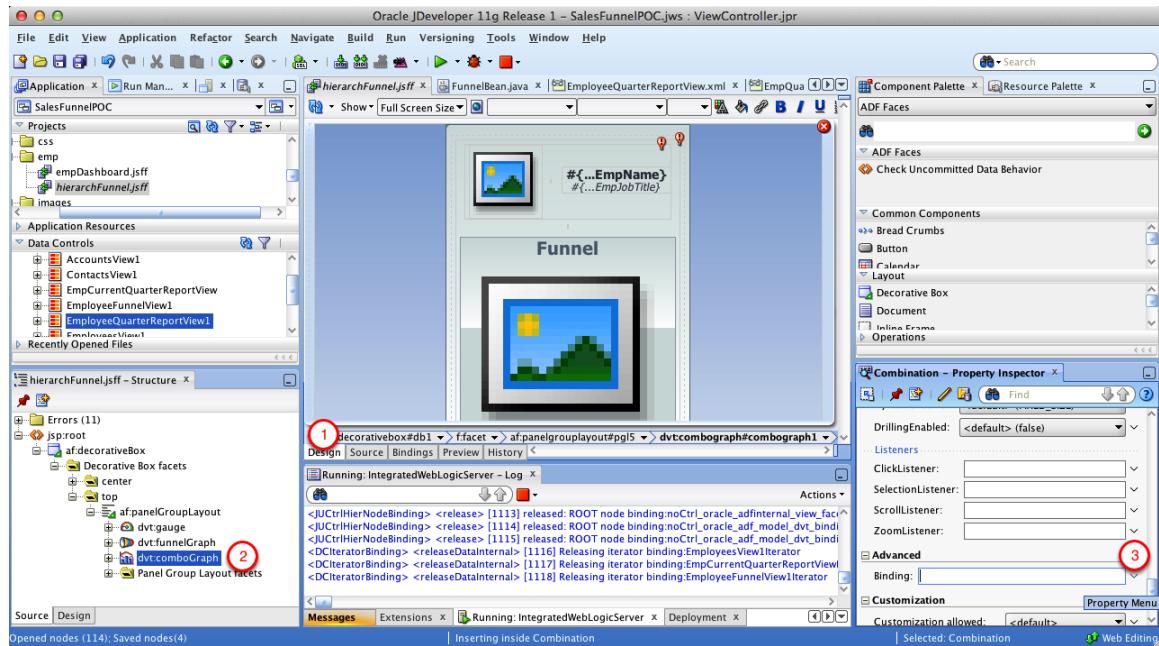
1. Select EmployeeQuarterReportView1Iterator
2. Select ExecuteWithParams Operation
3. Click OK

Change Action Id



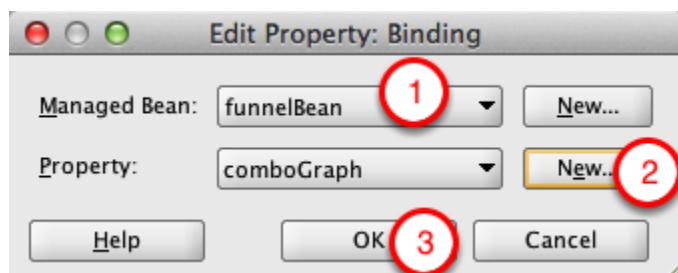
1. Select ExecuteWithParams
2. Change id to ExecuteQuarterReportWithParams

Add Combo Graph Binding



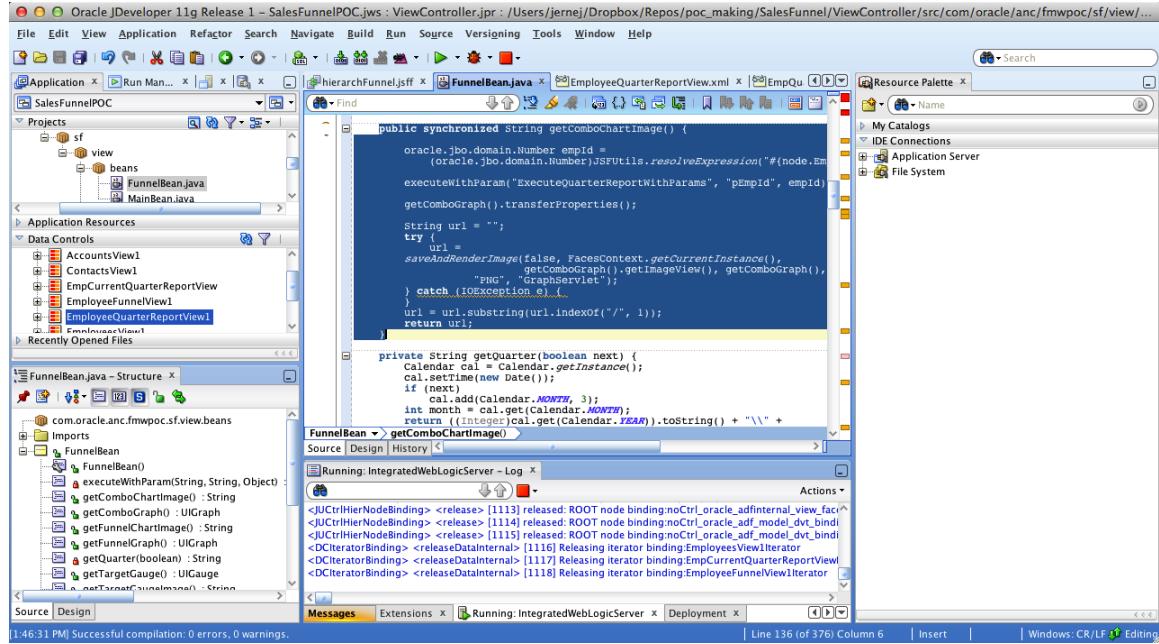
1. Open Design tab
2. Select comboGraph
3. Click on the down arrow next to Binding property and select Edit in the context menu

Edit Property: Binding



1. Select funnelBean Managed Bean
2. Click New next to Property and name it comboGraph
3. Click OK

Implement getComboChartImage



1. Open FunnelBean and paste the below code inside implementation

```

public synchronized String getComboChartImage() {

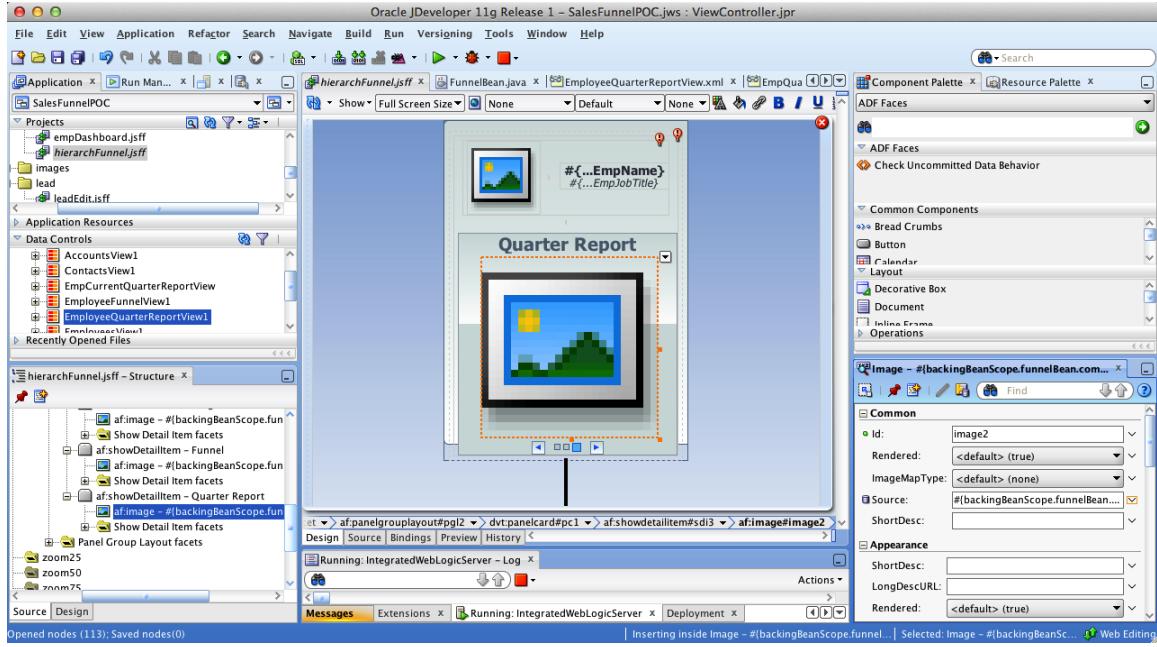
    oracle.jbo.domain.Number empId =
        (oracle.jbo.domain.Number)JSFUtils.resolveExpression("#{node.EmpId}");

    executeWithParam("ExecuteQuarterReportWithParams", "pEmpId", empId);
    getComboGraph().transferProperties();

    String url = "";
    try {
        url =
            saveAndRenderImage(false, FacesContext.getCurrentInstance(),
                getComboGraph().getImageView(), getComboGraph(), null,
                "PNG", "GraphServlet");
    } catch (IOException e) {
    }
    url = url.substring(url.indexOf("/", 1));
    return url;
}

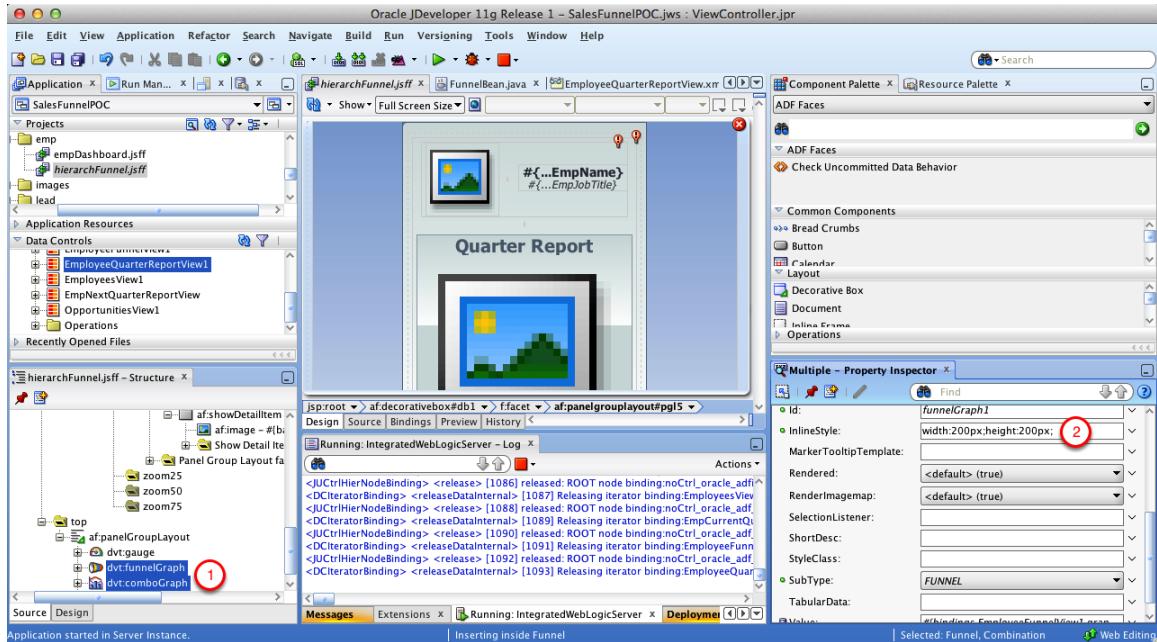
```

Exercise: add Quarter Report View to HView Node



Using similar steps as before, create a new card inside panelCard and render Quarter Report Chart image inside

Set Graph's InlineStyle



inlineStyle=width:200px;height:200px;