ADF Hands-On

Understanding Task Flow Activities

Abstract:

In this hands-on you create a bounded task flows to run as an ADF Region in an ADF Faces page.

Within this hands-on you create and use the following task flow activities: view activity, router activity, method call activity, save point restore activity.

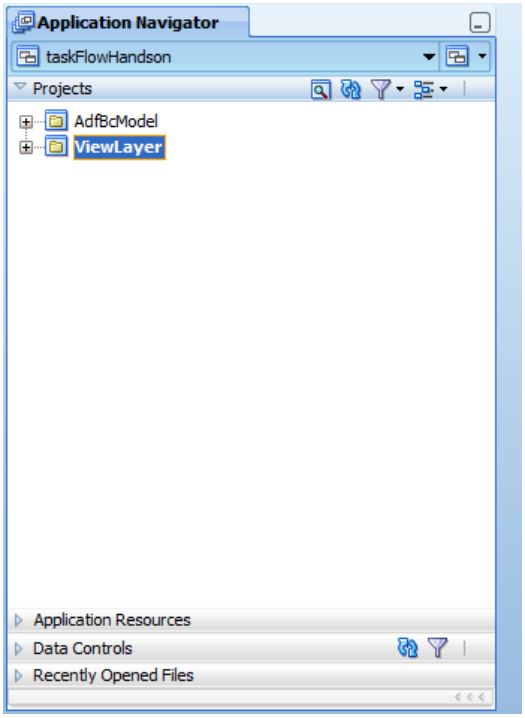
Duration: 60 minutes

Introduction

ADF Task Flow is an extension of the JavaServer Faces controller and provides functionality not available in the current and the next version of JSF. The ADF Controller – the owning technology of Task Flow – is a key success factor for Oracle ADF in the context of JavaServer Faces and Web 2.0 application development. You fid Task Flow not only within the context of custom ADF application development, but also in Oracle product suites like BPEL and BPM.

Prerequisite and Setup

Open the taskFlowHandson(part1).zip file and extract the contained folder containing the starter application. Extract it to a directory on your laptop.



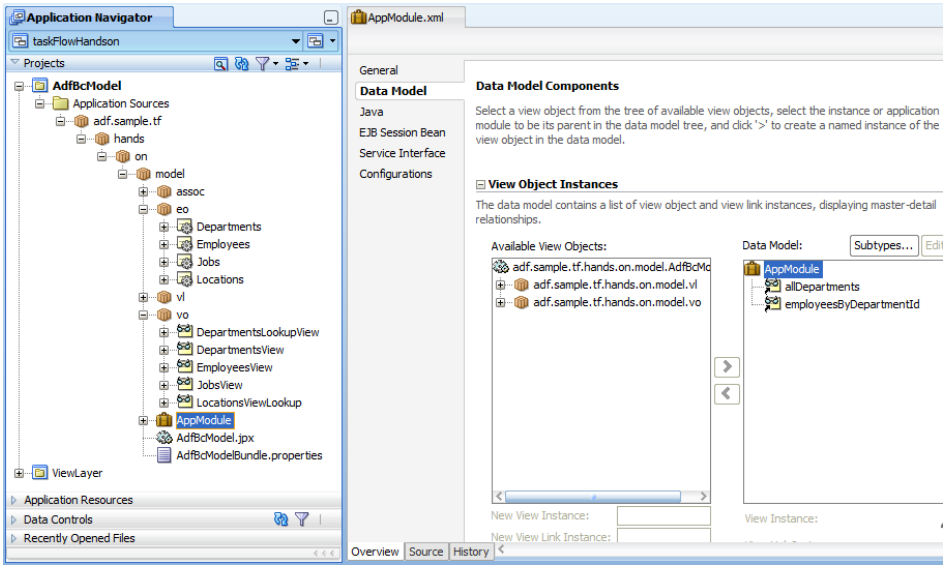
This hands-on requires a database, XE or ORCL, with the HR schema installed and enabled. Both databases by default have the XE schema and no scripts are required to run. Before working on the hands-on, select View | Database Navigator from the Oracle JDeveloper menu. Navigate to the "taskFlowHandson" entry and use the context menu to edit the "hrconn" connection properties.

ADF BC Model

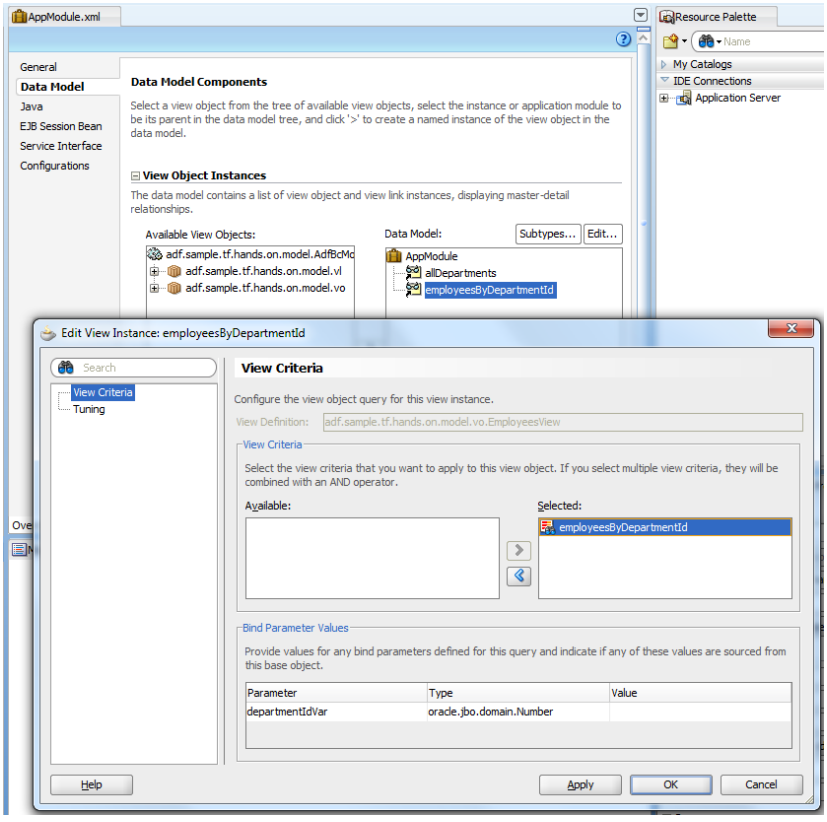
The creation and configuration of ADF Business Component models is not in the focus of this Web 2.0 training. However, to run ADF on real data, a business service is needed – and this one uses ADF Business Components.

ADF Business Components is a Java EE persistence layer built and owned by Oracle. It is used by Oracle Applications for web applications to query and update the database and to server side hold business logic. From a SOA and Web 2.0 perspective, Oracle ADF BC is attractive because it can be accessed from Web

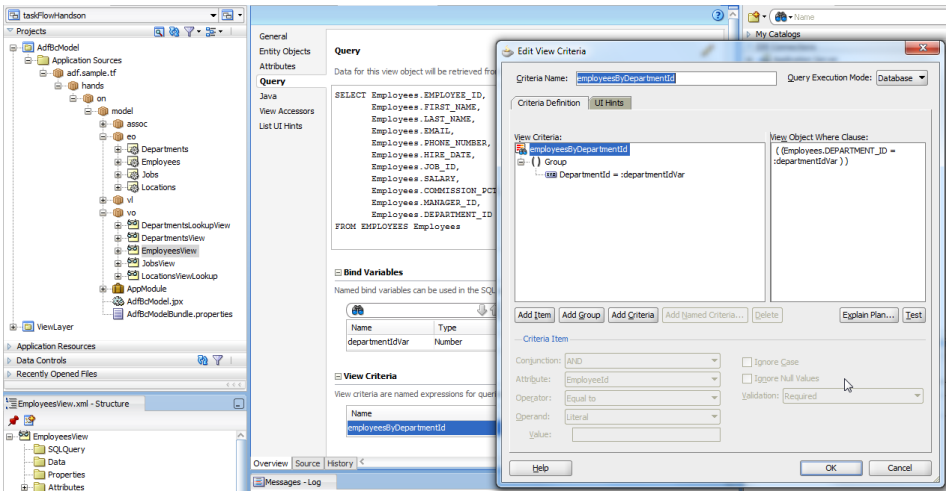
applications using the Oracle ADF BC Data Control and from SOA, exposing View Objects defined on the Application Module data model as SDO objects for remote access. In contrast to other persistence layers and business services, ADF Business Components provides a maximum of declarative developer support in terms of query building and configuration.



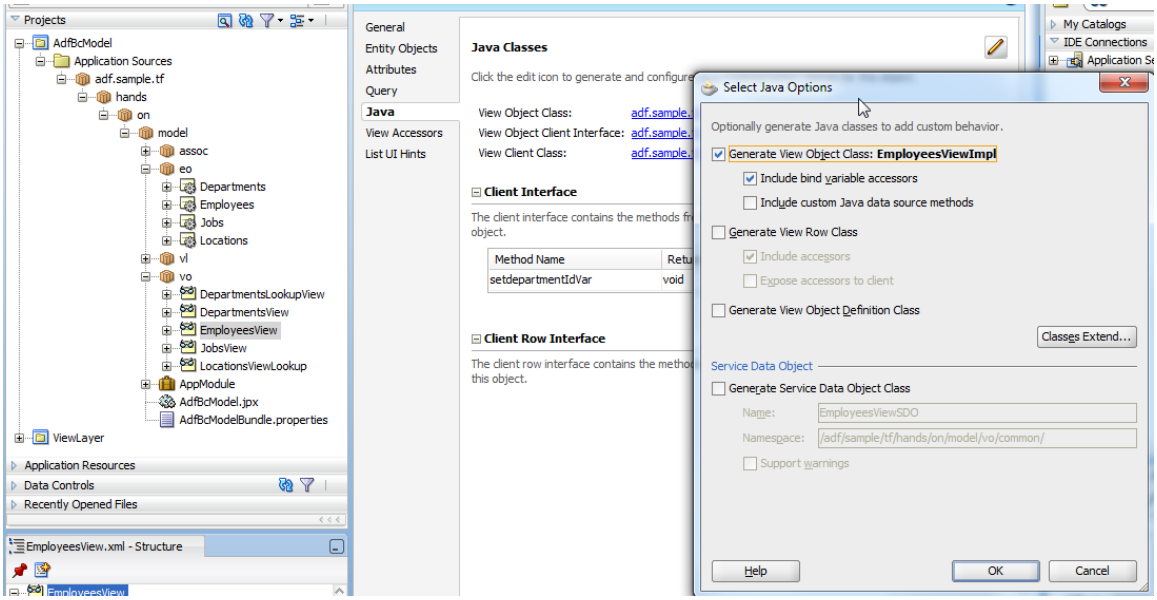
The ADF Business Component model used in this hands-on exposes two View Object instances, "allDepartments" and "employeesByDepartmentId". The View Objects are not dependent (though they could), which is a design decision made for this hands-on to show how to pass parameters from a consuming application to a bounded task flow.



The "employeesByDepartmentId" View Object instance is based on the "EmployeesView" view object. To filter the View Object query to only show employees belonging to a specific department, a View Criteria is used. View Criteria is a named where clause that you define declaratively on the View Object and that you apply dynamically at runtime or statically at design time. In this hands-on, the View Criteria is statically applied to the "employeesByDepartmentId" instance. This way, all access to the View Object instance is filtered by a departmentId that dynamically is passed into the query using a bind variable. The bind variable will be passed as a parameter to the Task Flow.



To expose the bind variable to the ADF model so it can programmatically be set. A View Object Impl class is generated for the EmployeesView object. As shown in the image below, creating the impl class is declaratively done from the View Object editor's Java option.



Last, but not least, the generated setter method of the bind variable is exposed on the client interface of the View Object. Again, the client interface can be created from the View Object editor Java option. As a result a method binding shows in the Data Control Panel.

All of the above is already created for you. All you need to do is to follow the hands-on tasks below to create an application with a bounded task flow in a region.

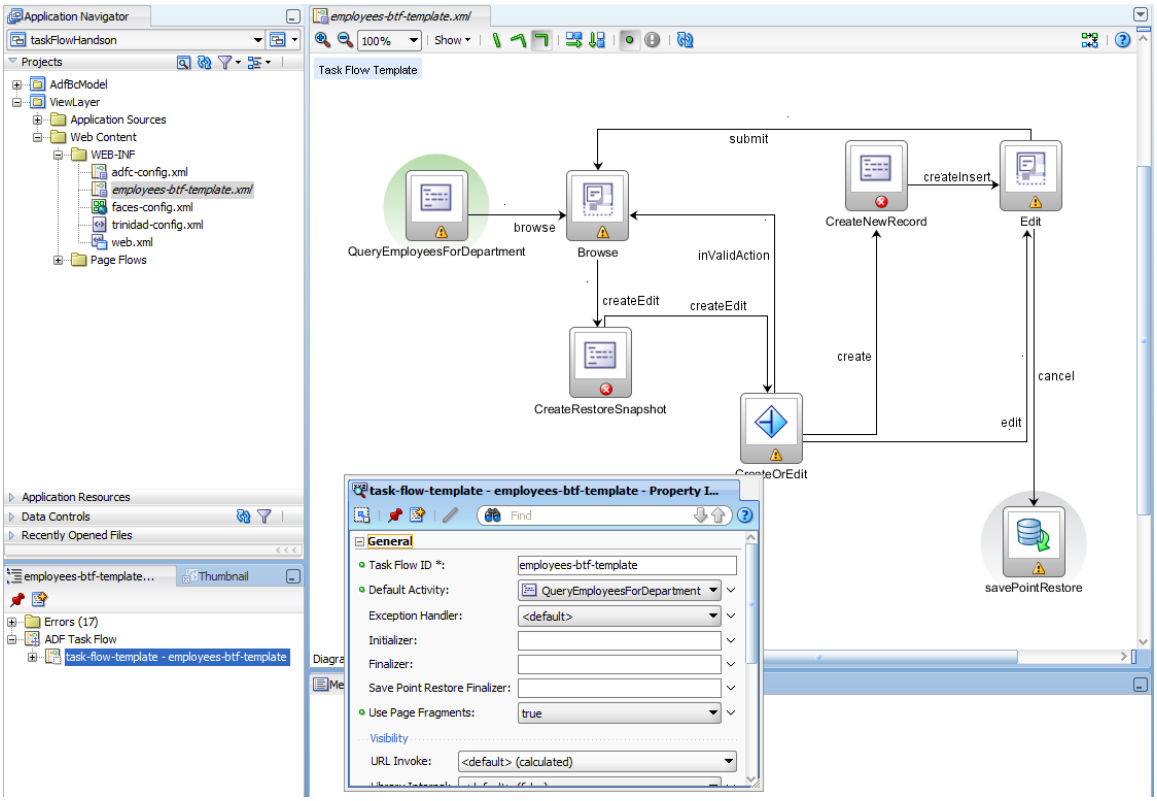
Exploring the Task Flow Starter Template

The hands-on starts from a bounded task flow template, which we use as a blueprint for user communication and implementation road map

1. Open the employees-btf-tenplate.xml file located in the ViewLayer project's WEB-INF directory. You open the task flow template with a double click

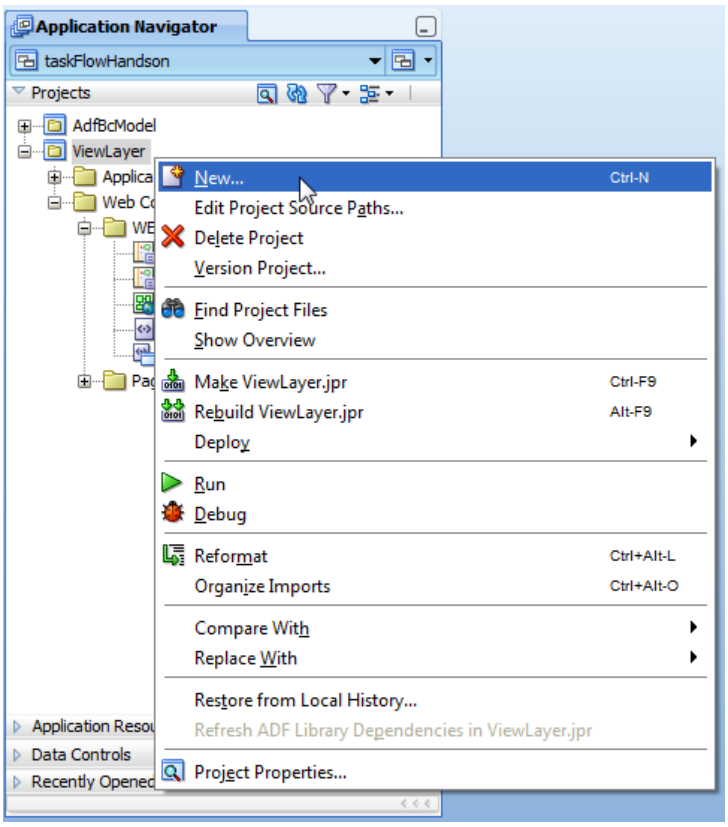
2. Make sure the task flow template is selected in the Application Navigator and open the Structure Window (ctrl+shift+S).

3. In the Structure Window, select the Task Flow template entry and open the Property Inspector (ctrl+shift+I). The Property inspector shows you the task flow configuration, like the Data Control sharing behavior and the task flow reentry allowance policy. The visual editor shows you the task flow control flow. Error icons indicate missing information, which is what the task flow consumer provides when building a new task flow based on this template.

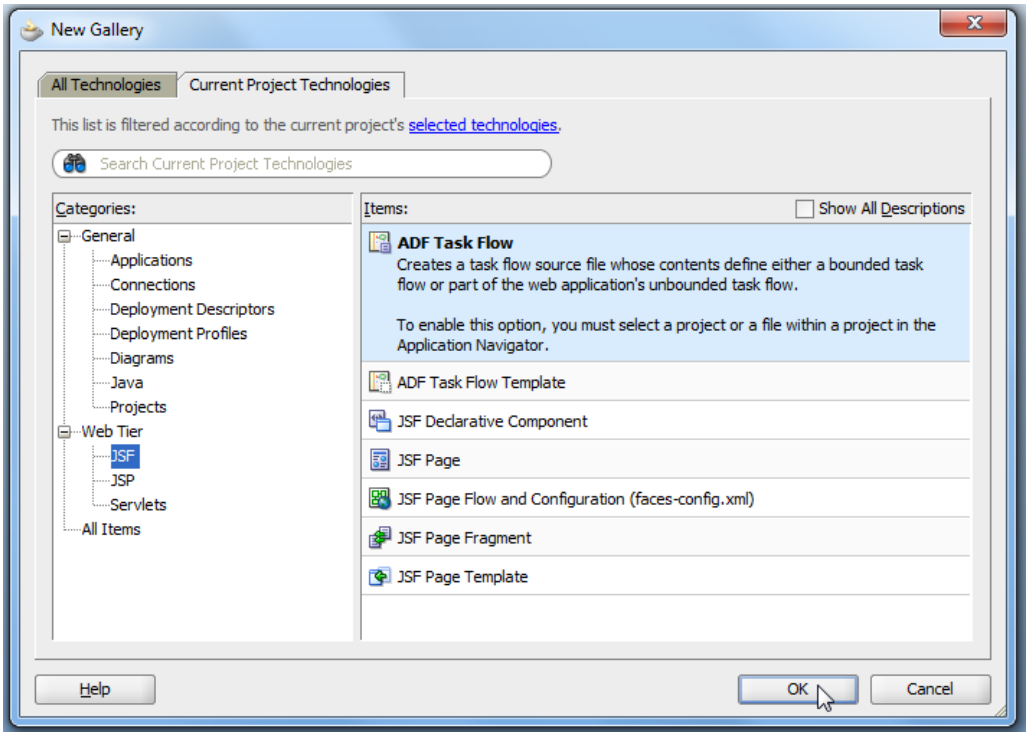


Creating the Bounded Task Flow

1. To build a new task flow, select the View Layer project and choose **New** from the context menu.

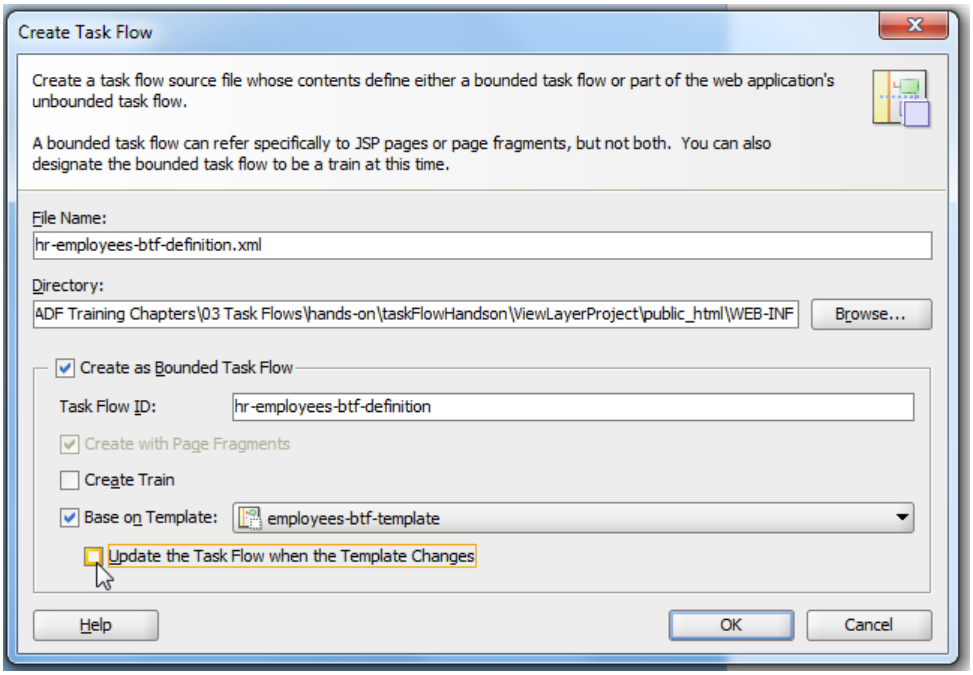


2. In the New Gallery dialog, expand the Web Tier entry and select the JSF node. In the list of Items, select ADF Task Flow



Note in the image above, that ADF Task Flow templates are build from the same dialog. Building a task flow template and building a bounded task flow actually is the same working step. If you know how to build a bounded task flow, you know how to build a template.

3. Ok the dialog



4. In the Create Task Flow dialog, define the file name for the new task flow to be "hr-employees-btf-definition.xml" and make sure

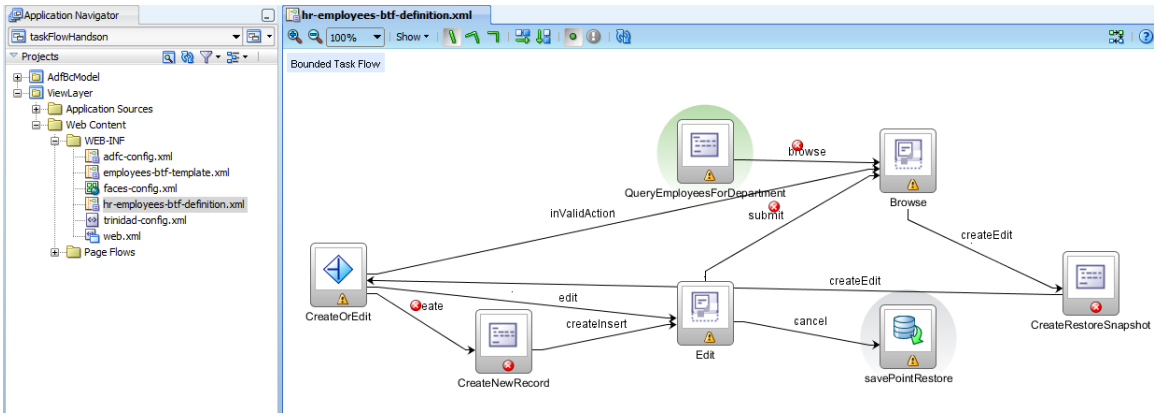
i. The "Create Bounded Task Flow" checkbox is selected

ii. The "Base on Template" checkbox is selected and the template is chosen

iii. The "Update the Task Flow when the Template Changes" check box is **not** selected

Note that this will create a new bounded task flow that copies the content of the task flow template into its source definition. This decouples the task flow from future changes in the template. The other option, to keep the dependency is good to use for task flow templates that only provide configuration settings displayed in the Property Inspector, or control flows you don't need to further edit or direct to (for example, the Exception Handler).

For task flows that you need to further edit the template content for a copy is the option to use.



5. The opened task flow looks a bit messy and not as well aligned as the template you studied before. This is because the task flow diagrammer definition is not part of the template metadata. However, you will fix this quite easily

6. At the bottom of the visual diagrammer, select the "Source" tab to show themetada source code

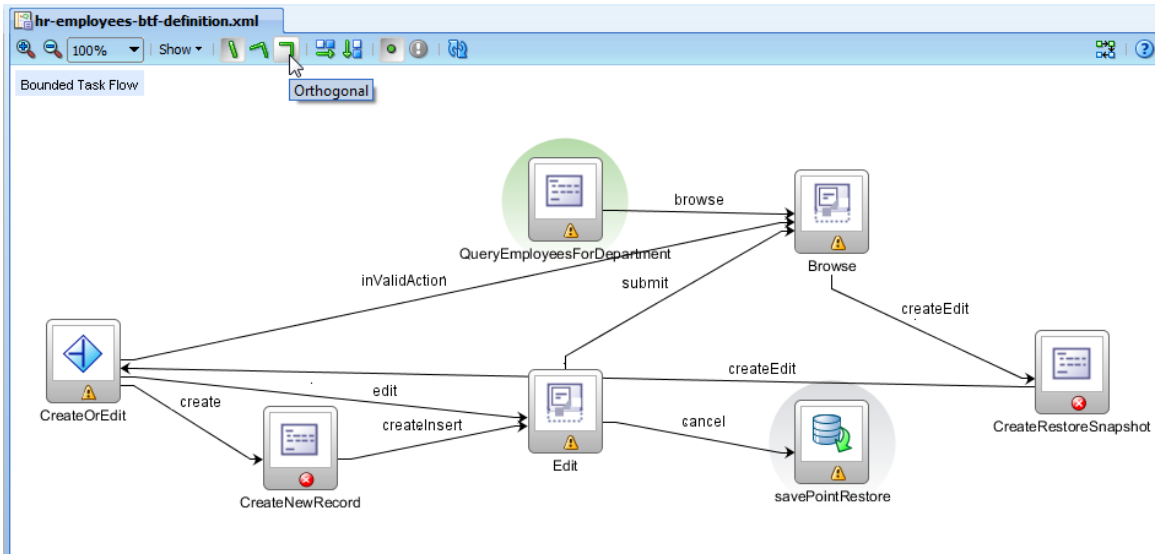
7. Some of the errors shown in this file are for control flow cases "id" values, which are not unique. The values don't need to be unique unless you plan to use MDS for customization. However, even if you don't have plans for using MDS, it is better to fix the id values right away.

8. To create unique id values, juts add a number to the existing id. For example, 20 will be changed to 201 etc.

```
</control-flow-case>
<control-flow-case id="  20">
  <from-outcome id="  19">create</from-outcome>
  <to-activity-id id="  18">CreateNewRecord</to-activity-id>
</control-flow-case>
```

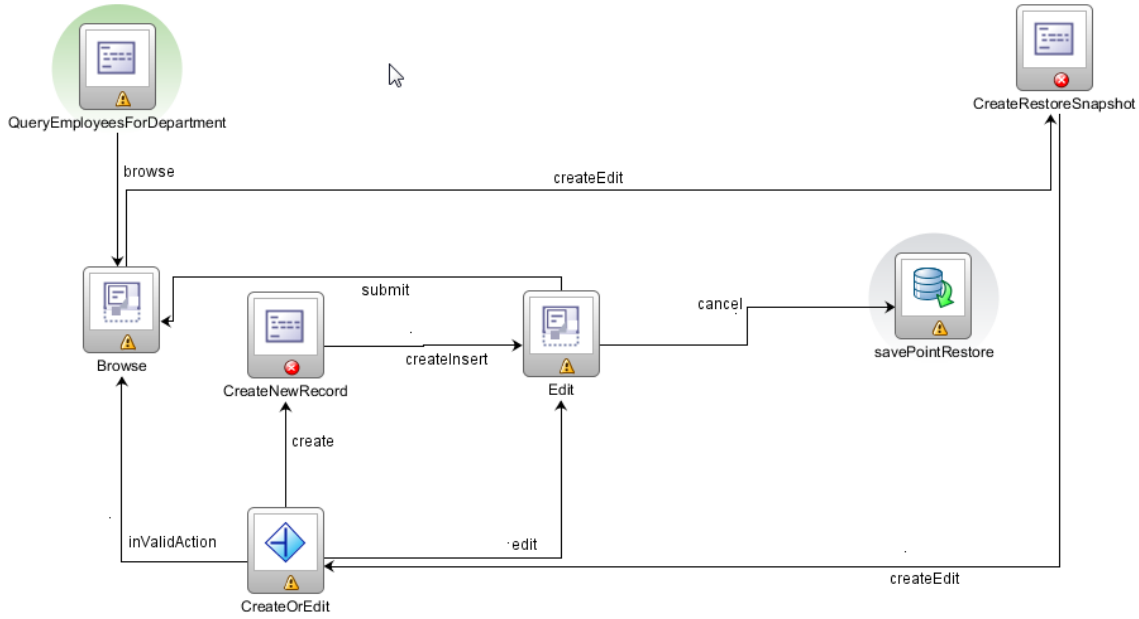
9. Once you fixed all the "id" violations, switch back to the visual design view

10. On the toolbar, click the "Orthogonal" align icon to change the diagram view



11. Move the activities on the diagram so they are in the order of their process flow

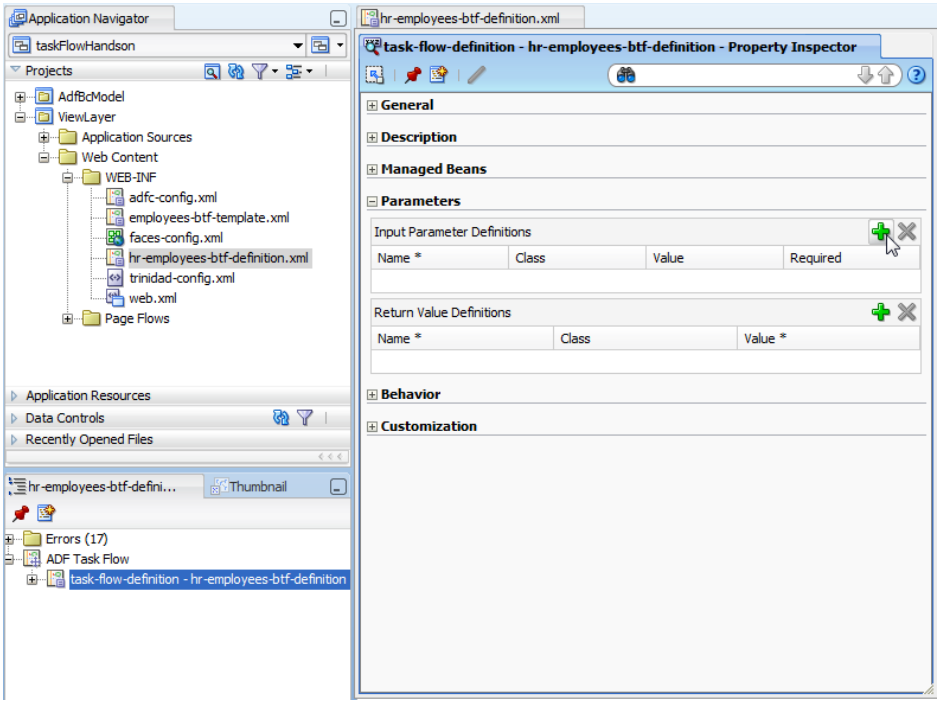
12. At the end your diagram should look like the one below



Implementing the Bounded Task Flow

1. To configure the bounded task flow, select the bounded task flow in the Application Navigator and then select the same task flow entry in the Structure Window

2. Open the Property Inspector

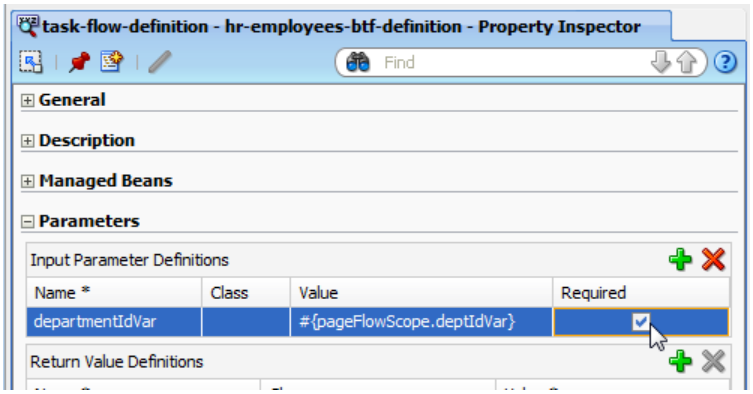


3. In the Property Inspector, navigate to the "Parameters" section and press the green plus icon to create a new input parameter. The input parameter passes the department Id for which to display employees to the task flow

4.Define the parameter name as "departmentIdVar" and the value as #{pageFlowScope.deptVar}. The parameter is required, so make sure the "Required" checkbox is selected too.

Note: The expression in the Task Flow parameter "value" attribute pints to an attribute in the pageFlowScope memory. The attribute however does not contain the value to be passed to the task flow but is the location where the task flow stores the input parameter value. So the department Id becomes available in the task flow from #{pageFlowScope.deptVar}

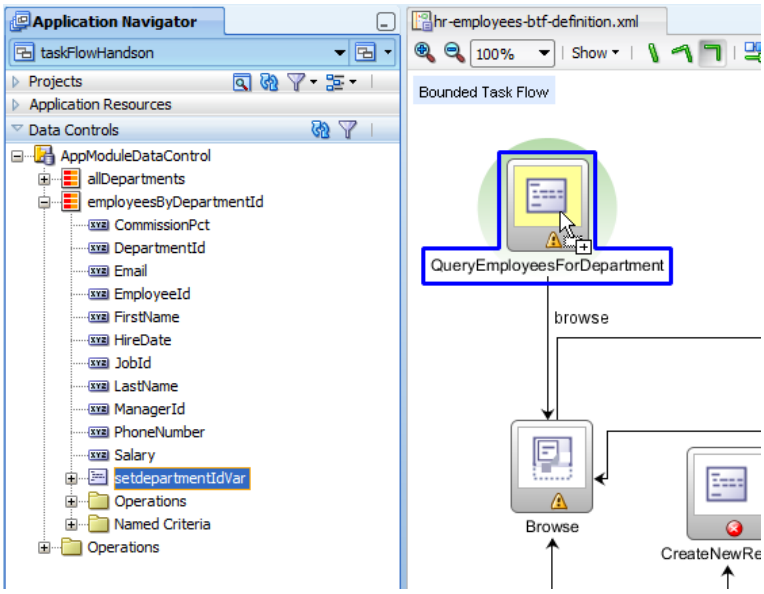
Note: Yes, the input parameter could be a part of the bounded task flow template. However, we did want you to do something at least.



5. Navigate back to the visual task flow diagrammer. If you closed it, start it by a double click onto the task flows XML definition file in the Application Navigator.

6.Expand the Data Controls panel and select the " setDepartmentIdVar" method entry

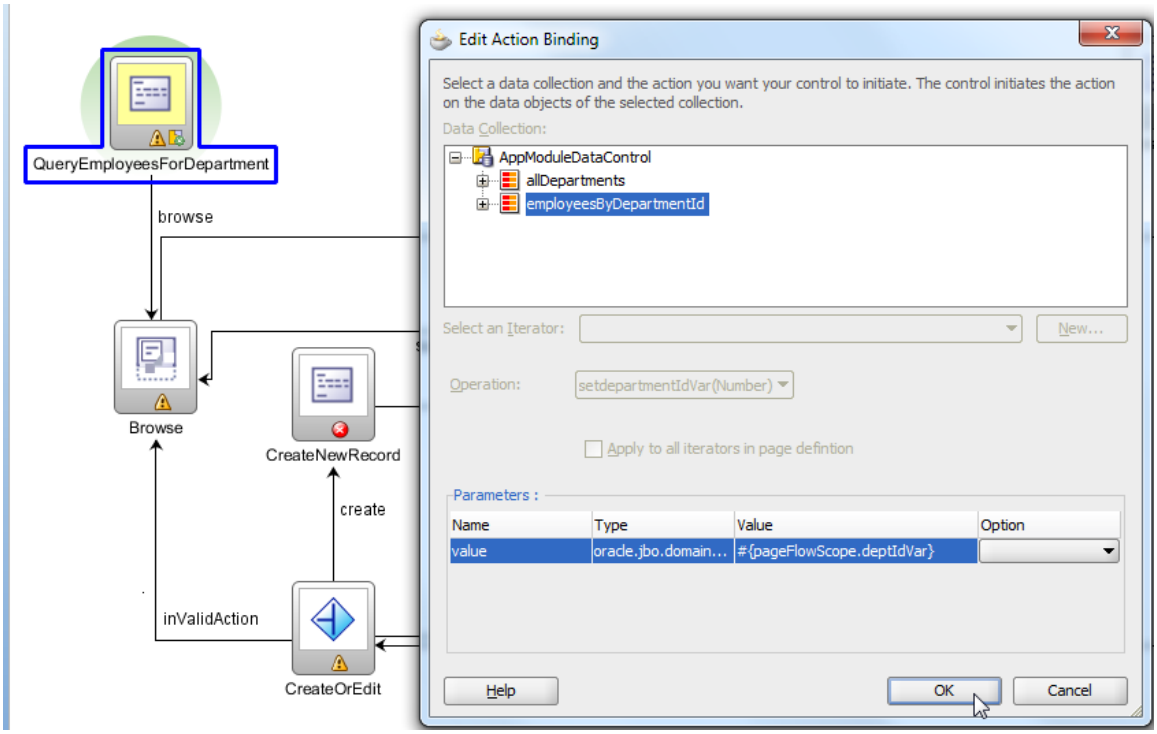
7. Drag the "setDepartmentIdVar" method from the DataControls panel to onto the "QueryEmployeesForDepartment" method call activity.



8. In the opened "Edit Action Binding" dialog, provide the EL reference to the input parameter value

#{pageFlowScope.deptVar}

Note: the #{pageFlowScope.deptVar} cannot be looked up using the Oracle Expression Builder dialog because memory attributes are not available at design time. If you want to design your task flow so that all variables are discoverable at design time, create a managed bean and configure it to the viewScope. Then reference a property (setter/getter pair) in the managed bean to store input and output parameter values. This also allows you to ensure the input argument is typed checked.



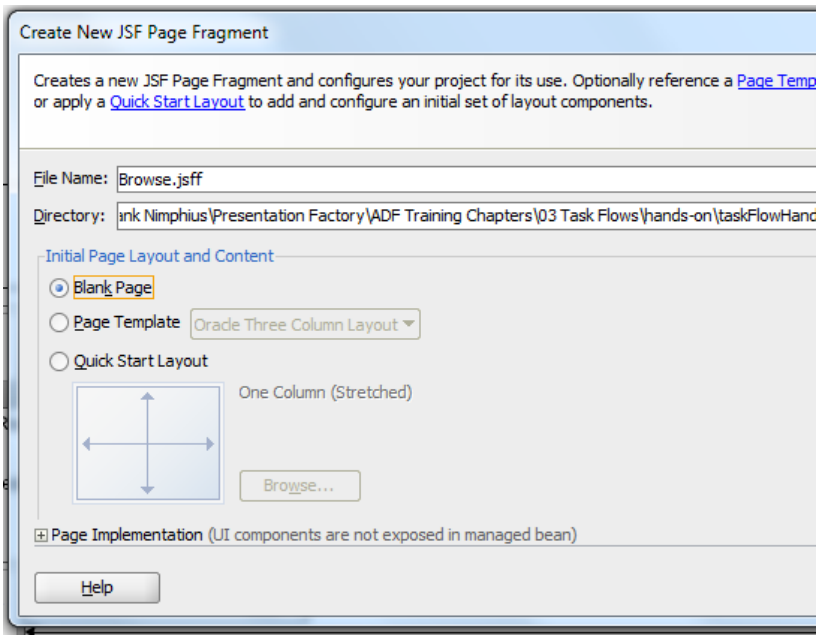
9. Ok the dialog to close it. Oracle JDeveloper creates a PageDef file (a binding file) for the method activity with a binding entry created for the set bind variable method and the method argument.

Note: another use of the method call activity is to configure it for managed bean method access, or – as we will see later - for a method accessed on the ADF framework

10. Double click the "Browse" activity to create the browse page

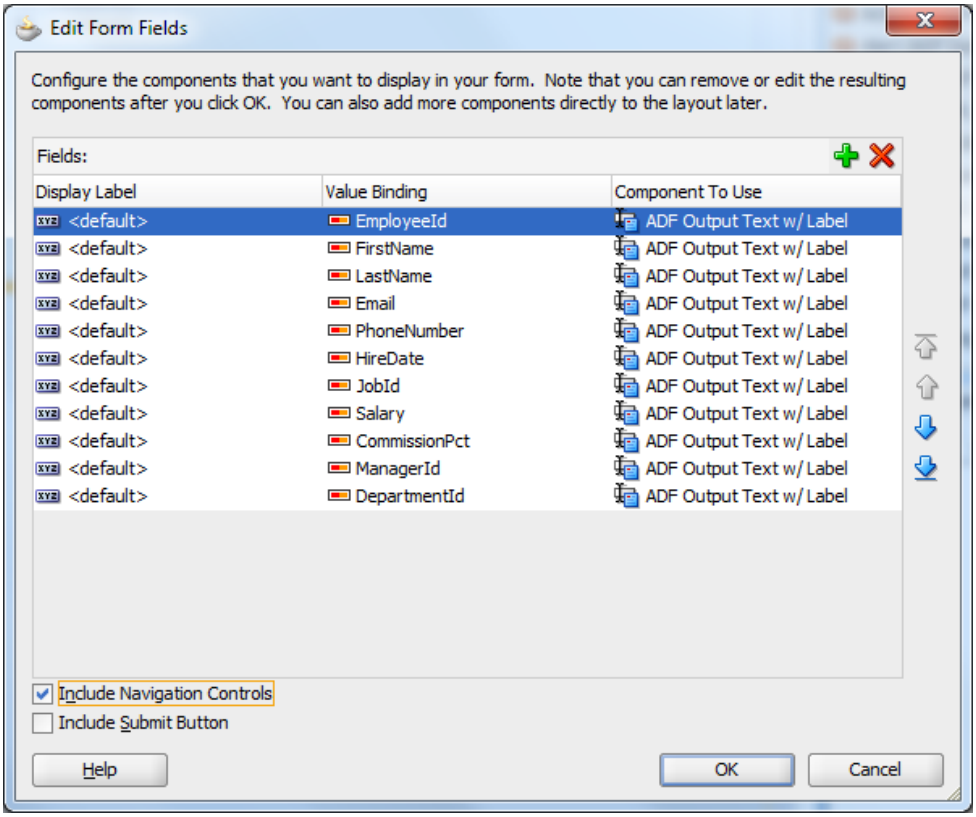
11. Keep the default settings and press Ok.

Note: The JSF view is created as a page fragment, which is a requirement for bounded task flows that run as an ADF Region



12. drag the "employeesByDepartmentId" collection (View Object) entry from the Data Controls panel onto the JSF Browse view

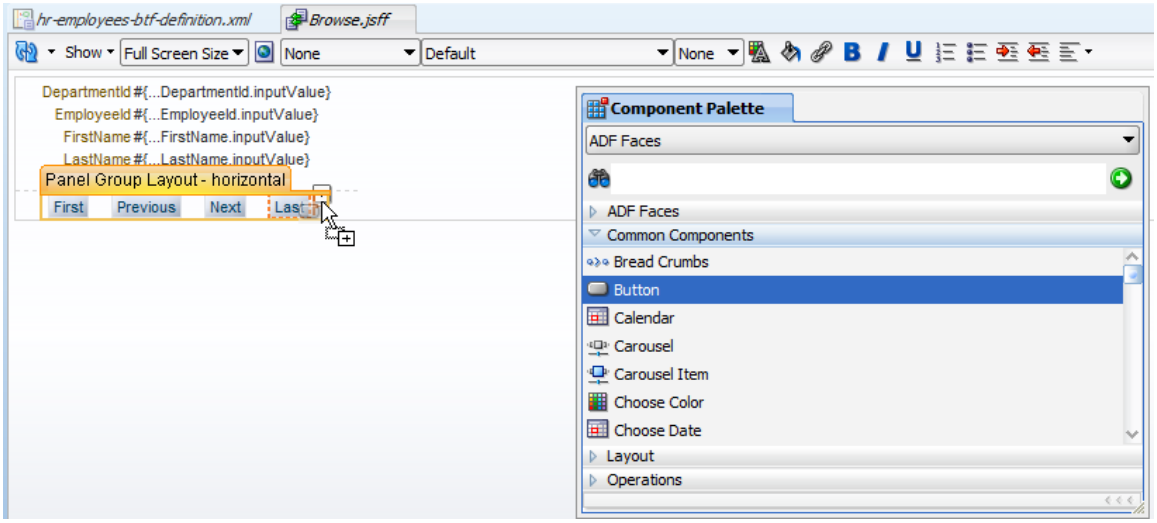
13. In the opened context menu choose Form | Read-Only form



14. Select the "Include Navigation Controls" checkbox for JDeveloper to add command buttons to navigate the employees collection

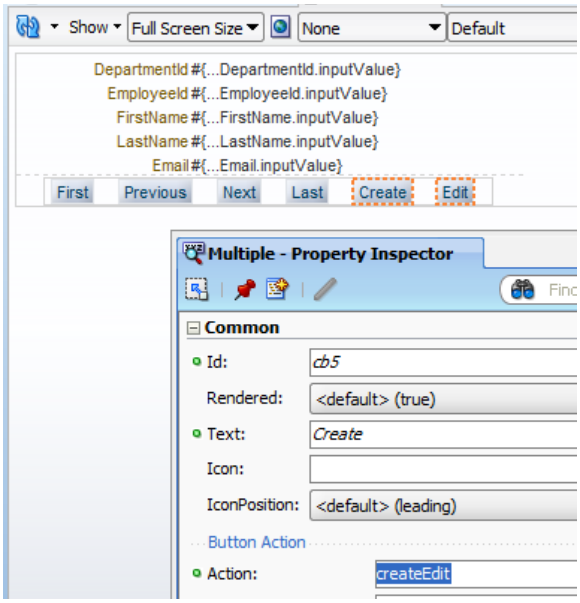
15. Ok the dialog

16. With the JSF visual editor open, open the Component Palette (ctrl+shift+P)



17. In the "Common Components" section, select the "Button" component and drag it next to the "Last" button

18. Name the button "Create" and have its "Action" property pointing to the "createEdit" control case



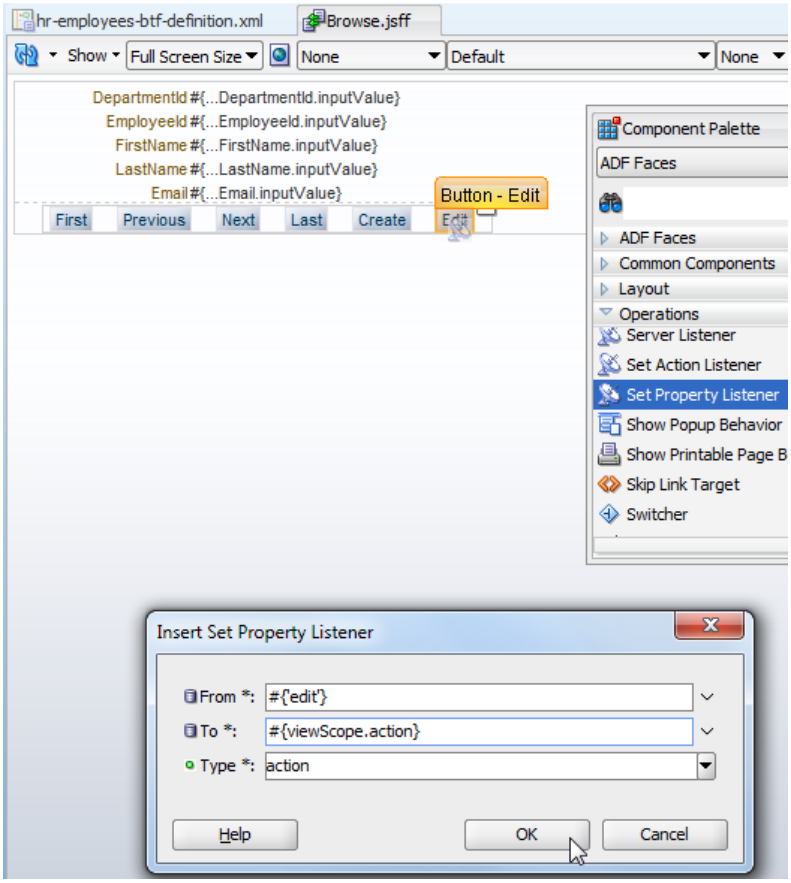
19. Drag another Button next to the "Create" button and name it "Edit"

20. In the Component Palette, expand the "Operations"section and select the "Set Property Listener" entry

21. Drag the "Set Property Listener" component onto the "Edit" button

22. In the opened dialog, provide the following information

From	#{'edit'}
To	#{viewScope.action}
Type	action



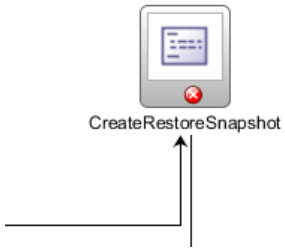
Note: Using the "Set Property Listener" component, the value of the "From" field is copied into an attribute stored in the viewScope memory. Pressing the Edit button thus sets this value to "edit". The information is later read by the router component.

23. Repeat steps 21 and 22 for the Create button. However, for the Create button. Provide the following information

From	#{'create'}
To	#{viewScope.action}
Type	action

So pressing the Create button sets the "action" attribute value to "create"

24. The next control flow stop is the "CreateRestoreSnapshot" method binding. The method binding is supposed to take a snap shot of the current transaction and view stack for a possible cancel action performed when editing an existing row or new row



25. Select the CreateRestoreSnapshot and open the PropertyInspector

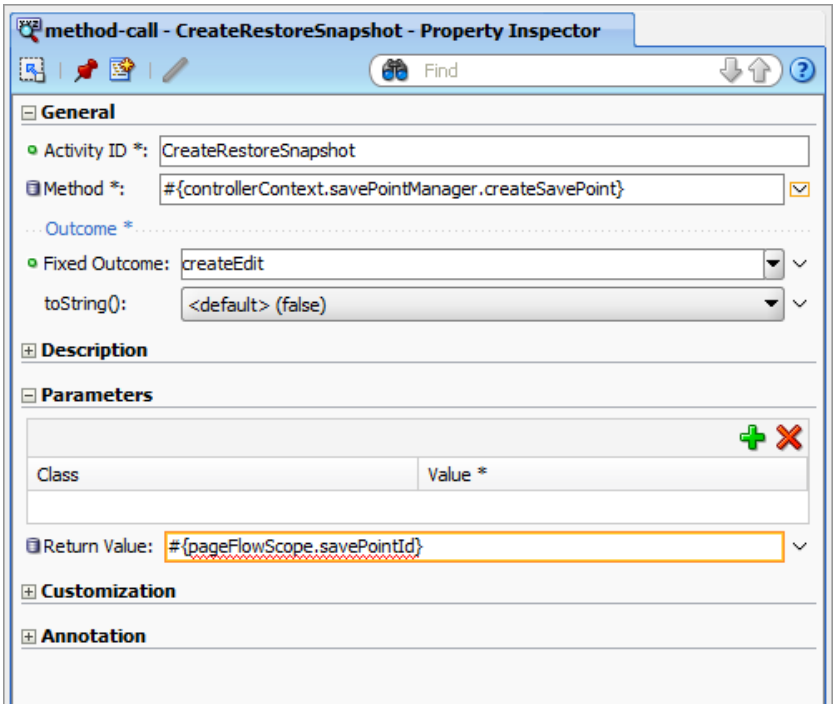
26. Press the "down arrow" icon at the right of the "Method" Property

Note: The "Method" uses EL to point to the managed bean or framework method that should be executed when the control flow arrives as the method activity

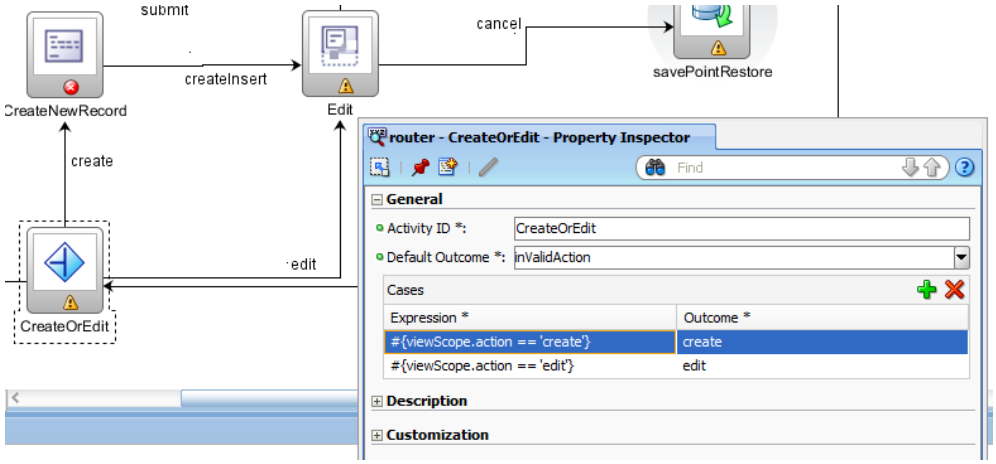
27. From the Context menu, open the Expression Builder dialog to create the following EL entry

#{controllerContext.savePointManager.createSavePoint}

28. Note that the "Fixed Outcome" is set to "createEdit", which is the control flow case to follow to continue with the process



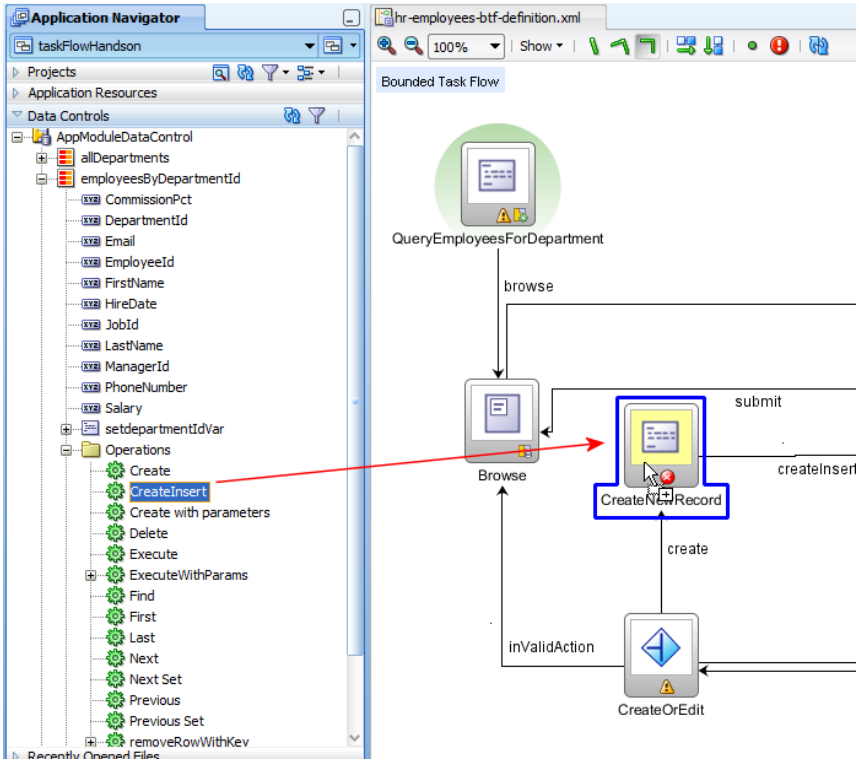
29. Note the "Return Value" property that saves a String returned from the createSavePoint method to the "savePointId" attribute in the task flow pageFlowScope. The Return Value property is generally used by developer to save the value returned by a called method.



30. Select the Router Activity icon and open the Property Inspector

31. The router has two Expressions pre-defined by the template. If the viewScope.action reference is set to "create" then the router continues following the "create" navigation case. Otherwise, if the outcome is "edit" the edit navigation case is followed.

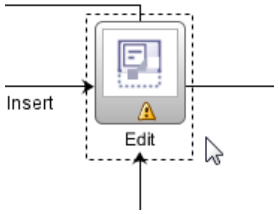
Note: Instead of using the "Set Property Listener" component to set the "create" and "edit" values to a memory attribute and the router to evaluate this setting, we could have used two navigation cases – one leading to the create form path and one to the edit form path – referenced in the "Action" property of the "Create" and "Edit" buttons. However, it was our goal in this hands-on to introduce the router activity as it is a very powerful while still declarative option for branching navigation paths.



32. Drag the "CreateInsert" operation from the DataControls panel to the "CreateNewRecord" method activity. Make sure you use the "CreateInsert" operation under the "employeesByDepartmentId" View

Object. This creates a PageDef file for the operation and references the binding from the method call activity's "Method" property

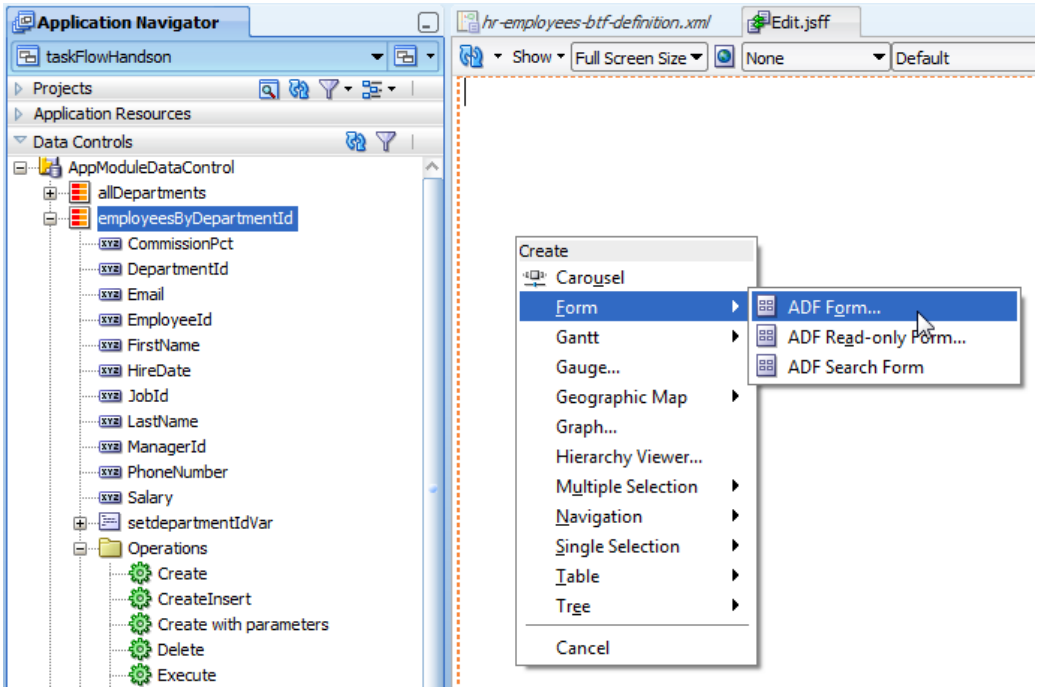
33. Double click the "Edit" view activity to create a new "Edit" form



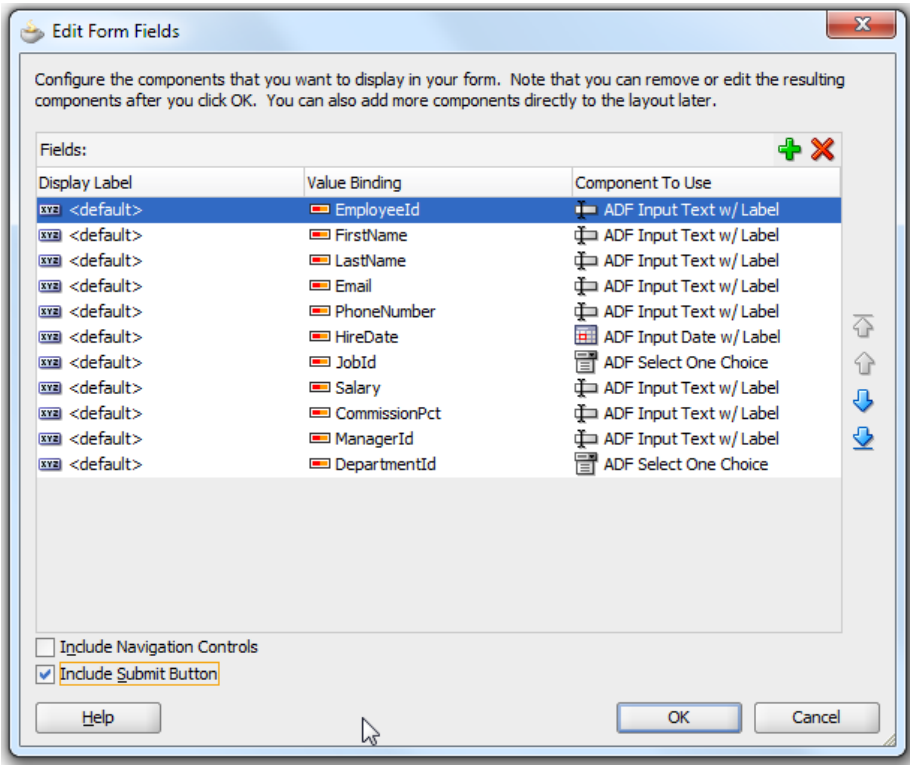
34. Accept the default settings and press "Ok" to close the dialog and create the page fragment

35. Drag the "employeesByDepartmentId" collection from the Data Controls panel to the Edit.jsff view

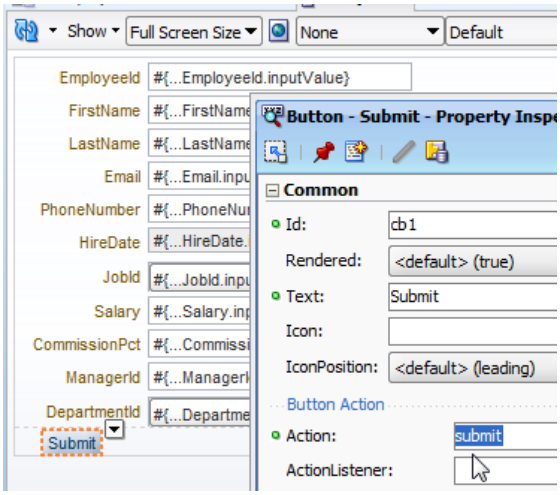
36. Choose Form | ADF Form from the context menu



37. Check the "Include Submit Button" check box and press OK to close the "Edit Form Fields" dialog



38. Selectthe Submit button and set its "Action" property to the "submit" String. This way, pressing the submit button will navigate back to the employees Browse page

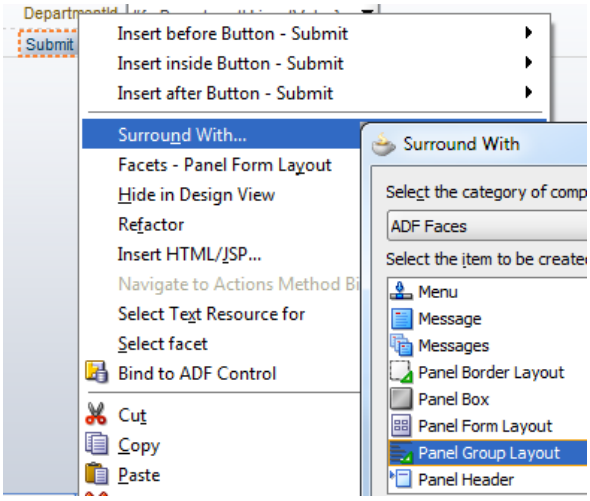


39. To create a Cancel button that aligns with the Submit button, select the "Submit" button and choose "Surround With" from the context menu.

40. In the Surround With" dialog, select the "Panel Group Layout" component

41. Press "Ok"

42. Select the "Panel Group Layout" component and sets its "Layout" property to "horizontal" in the Property Inspector



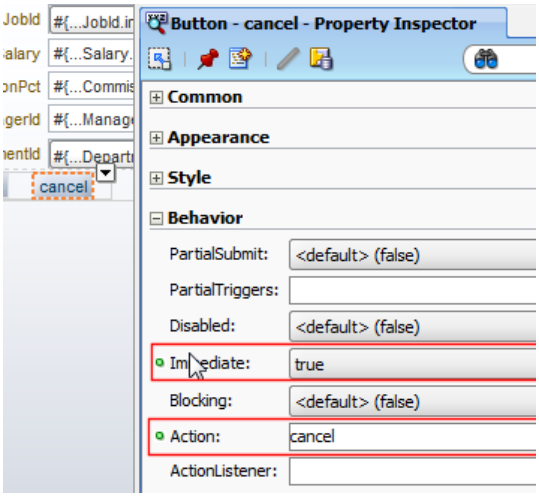
43. Drag and drop a "Button" component from the Component Palette onto the Edit view, right next to the "Submit" button

44. Select the new button and open the Property Inspector

45. Set the button Text Property to "Cancel"

46. Set the "Immediate" property to "true" to suppress field validation when cancelling the edit

47. Set the "Action" property to cancel to follow the cancel navigation path to restore the "old" controller state



48. Save and close the page edit tab

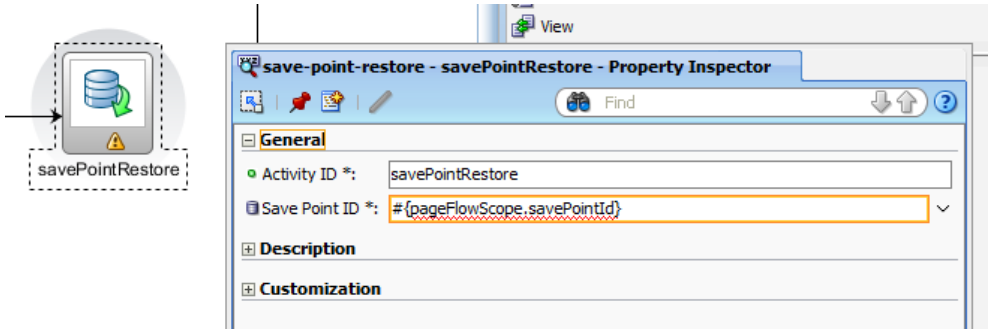
49. Select the "savePointRestore" activity in the task flow diagram and open the Property Inspector. The Save Point Restore activity references the Save Point Id created previously by the method call activity.

50. Reference the location of the stored savePointId in the "SavePoint Id" property. The reference is

#{pageFlowScope.savePointId}

and needs to be manually provided

Note: The save pointed can also be stored into and read from a managed bean property. The use of memory attributes is only an option, not mandatory



Enabling Savepoint

1. To create save point, the controller needs to be configured with a database connection to a schema it can use to create a table in. Once the table is created, save points are stored and read from it when the createSavePoint or readSavePoint methods are called on the ControllerContext | SavePointManager object.

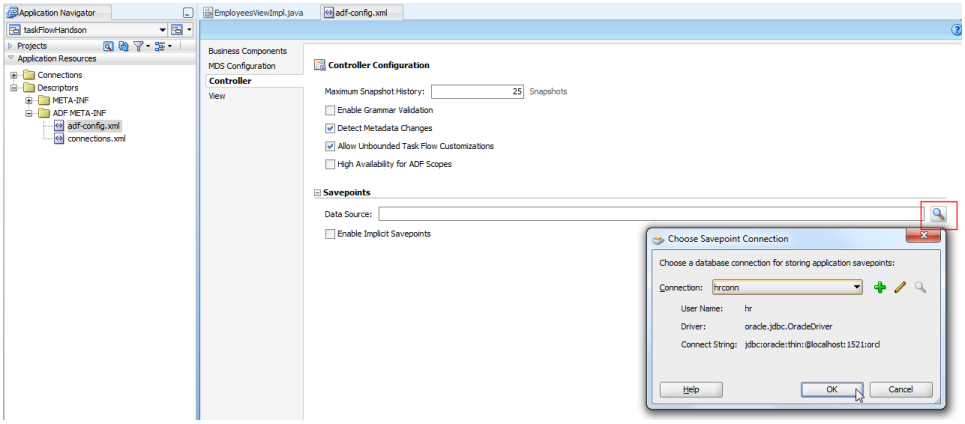
2. Expand the "Application Resources" accordion in the Application Navigator

3. Select and double click the adf-config.xml entry to open the edit dialog

4. Select the "Controller" category

5. Press the magnifier icon to select or create a database connection to use

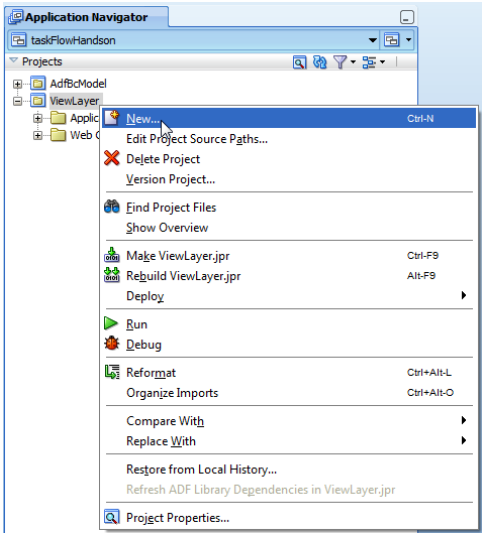
6. Ok the database connection dialog



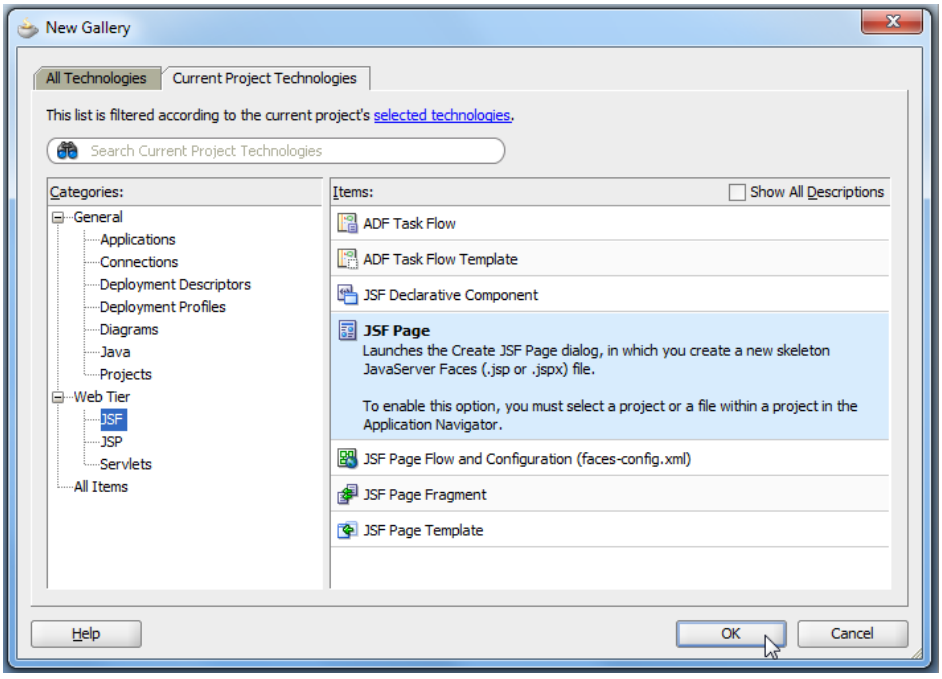
In the finished part of this hands-on you created a bounded task flows from an ADF task flow template. You used advanced features like a method call activity, router activity and save point. Next you are going to build an application to run the task flow.

Adding the Bounded Task Flow to an Application

1. Select the View Layer project and choose "New" from the context menu



2. In the Web Tier | JSF section, choose the "JSF Page" entry

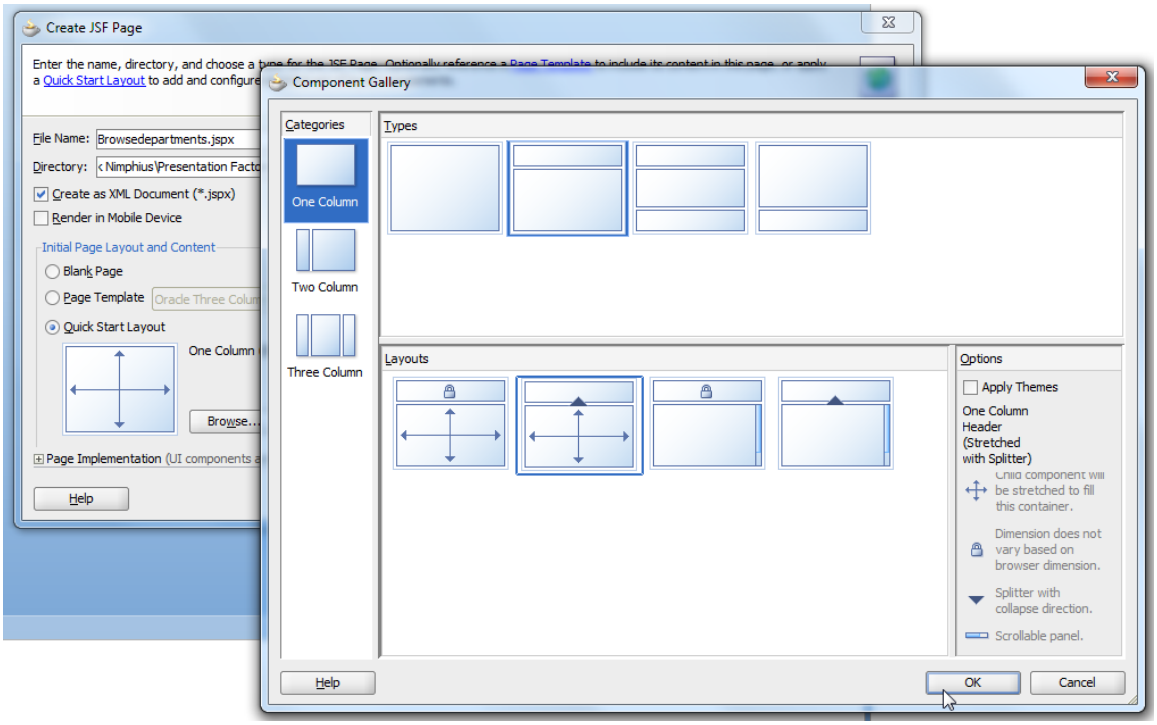


3. In the opened "Create JSF Page" dialog, define the page name as "BrowseDepartments.jspx" and make sure the "Create as XML Document (*.jspx) option is selected

4. Select the "Quick Start Layout" option and press the "Browse" button

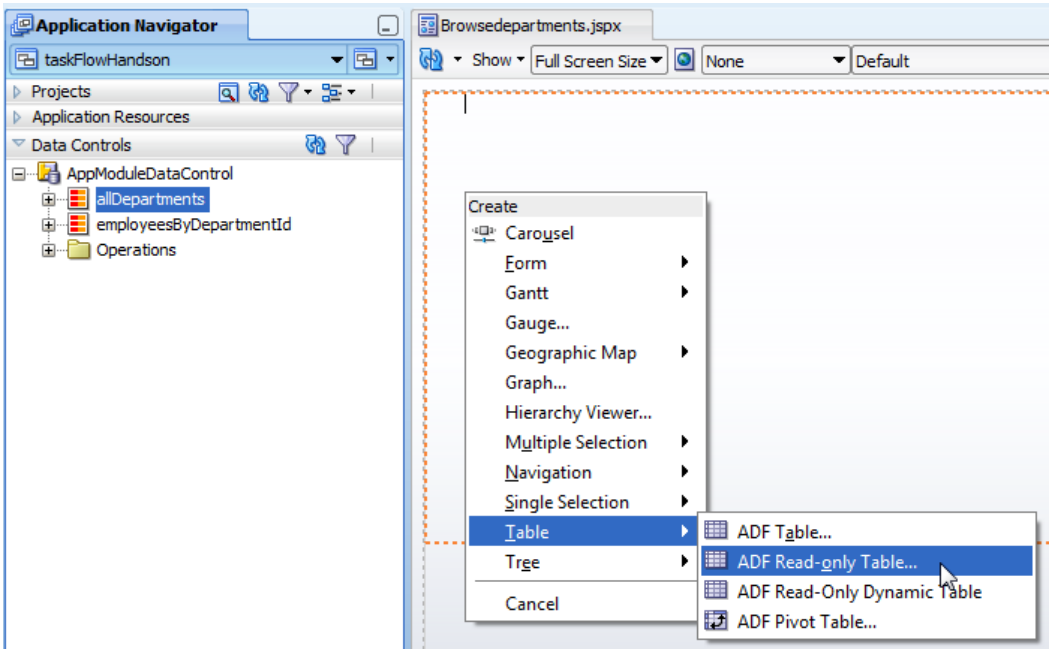
5. In the "Component Gallery", choose a "One Column" layout.

6. Select the 2nd layout type and press Ok.



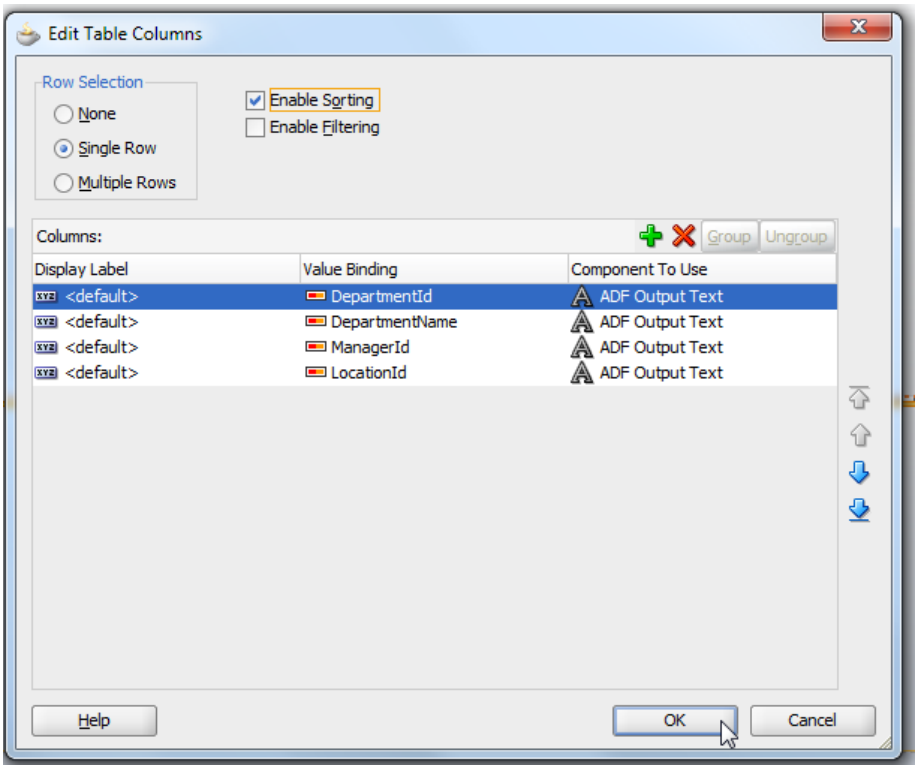
7. Drag the "allDepartments" collection from the Data Control panel to the top of the page

8. Choose Table | ADF Read-only Table from the context menu



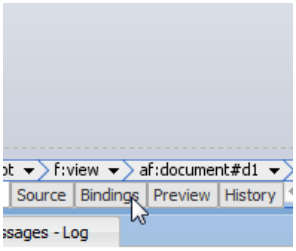
9. In the table creation dialog, ensure you select "Single Row" in the "Row Selection"

Note: The screen shot is taken from JDeveloper 11.1.1.4. The table edit dialog looks different in earlier releases of Oracle JDeveloper.



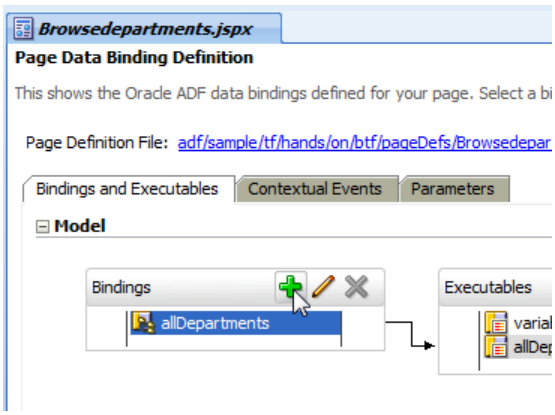
10. Press OK to close the dialog

11. Back in the visual page editor, click on the "Bindings" tab at the bottom of the visual editor

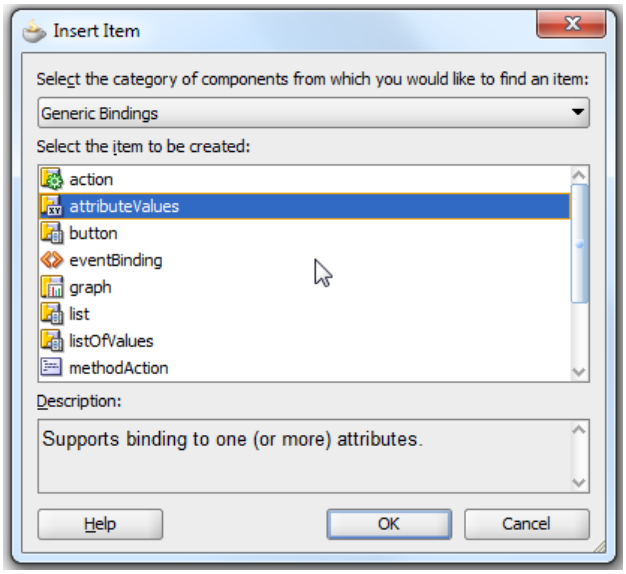


12. To simplify passing the department Id of the selected DepartmentsView table row to the ADF bounded task flow, you create an attribute binding for the DepartmentId attribute. The ADF binding layer ensures that the attribute binding always points to the current row's department Id value.

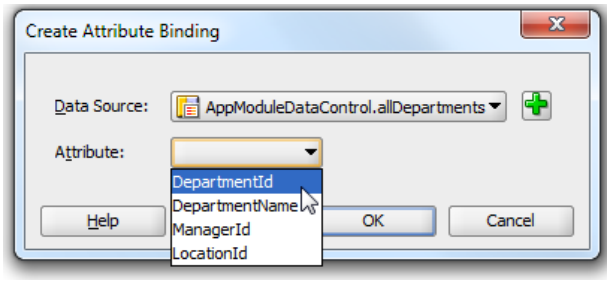
13. Press the green plus icon in the "Bindings" section



14. From the list of "Generic Bindings", select "attributeValues" and press Ok

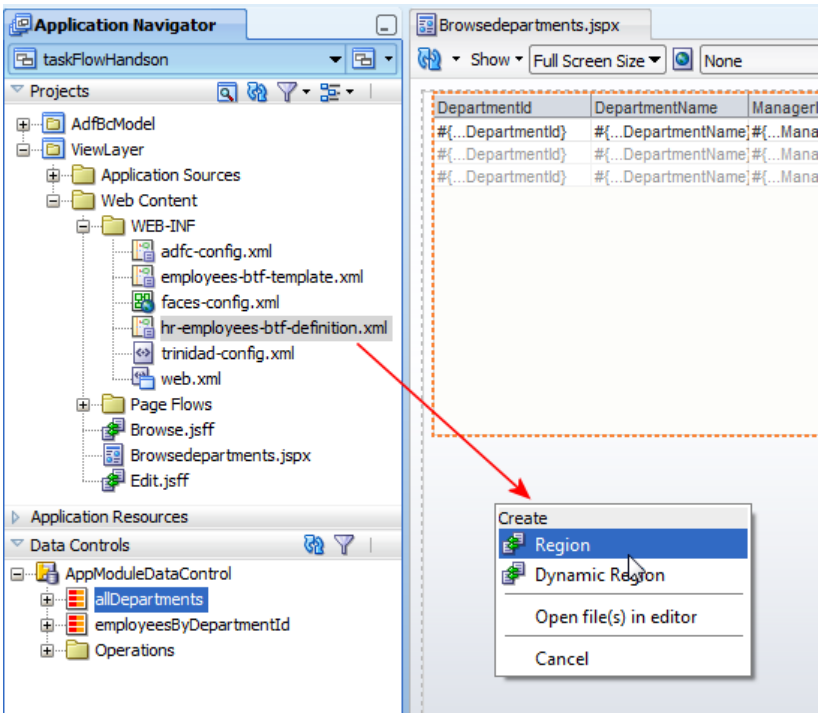


15. In the "Create Attribute Binding" dialog, choose the "allDepartments" iterator entry and "DepartmentId" Attribute.



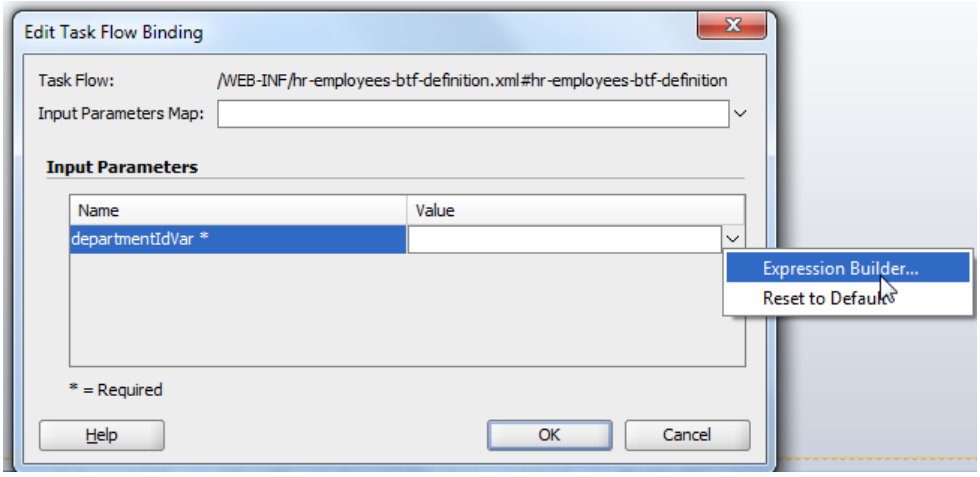
16. Ok the dialog

17. Drag the create bounded task flow "hr-employees-btf-definition.xml" entry located in the View Layer | Web Content | WEB-INF folder into the lower half of the page

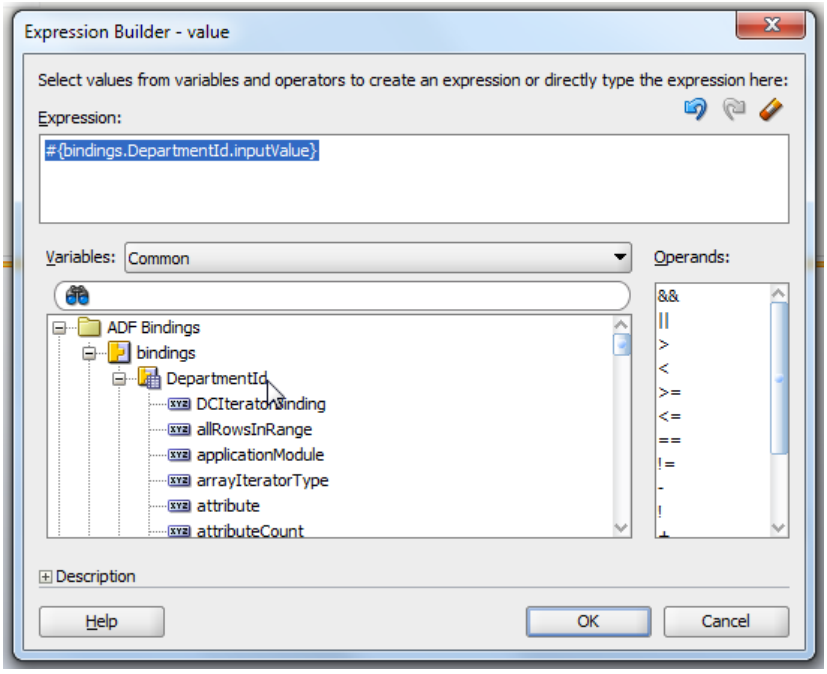


18. Choose "Region" from the context menu

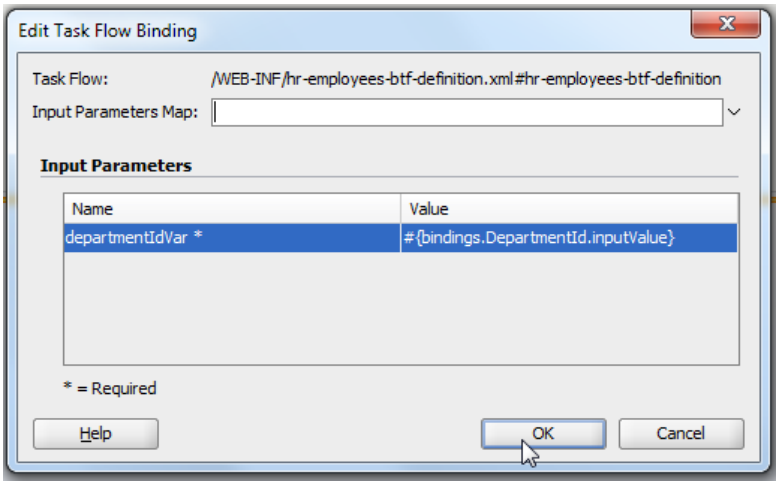
19. In the "Edit Task Flow Binding" dialog that opens for providing the mandatory task flow input parameter, click onto the Expression Builder context menu option



20. In the "Expression Builder" dialog, expand the ADF Bindings | bindings | DepartmentId entry and select the inputValue attribute. The inputValue then contains the value of the department Id for the current selected table row



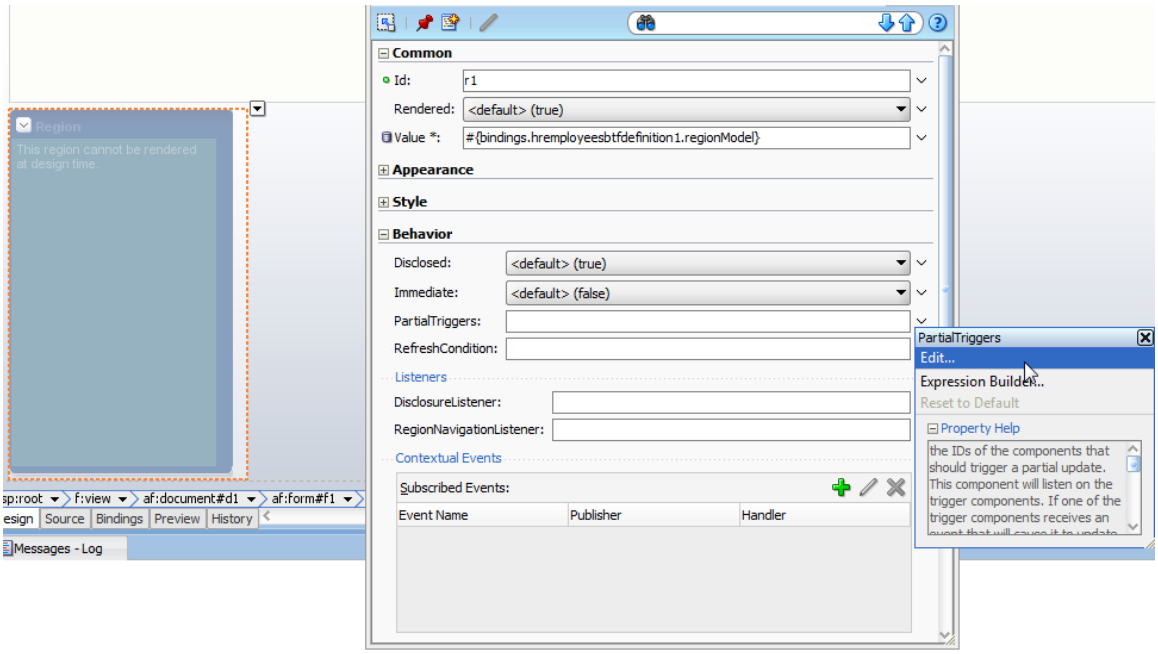
21. Ok the dialog and the next



22. The ADF Region needs to be refreshed to show its content updated for a row selection. For this. Select the region component and open the Property Inspector

23. In the Property Inspector, search for the "Partial Triggers" property

24. Use the Edit context menu option that you see when clicking the arrow icon next to the Partial Triggers property field

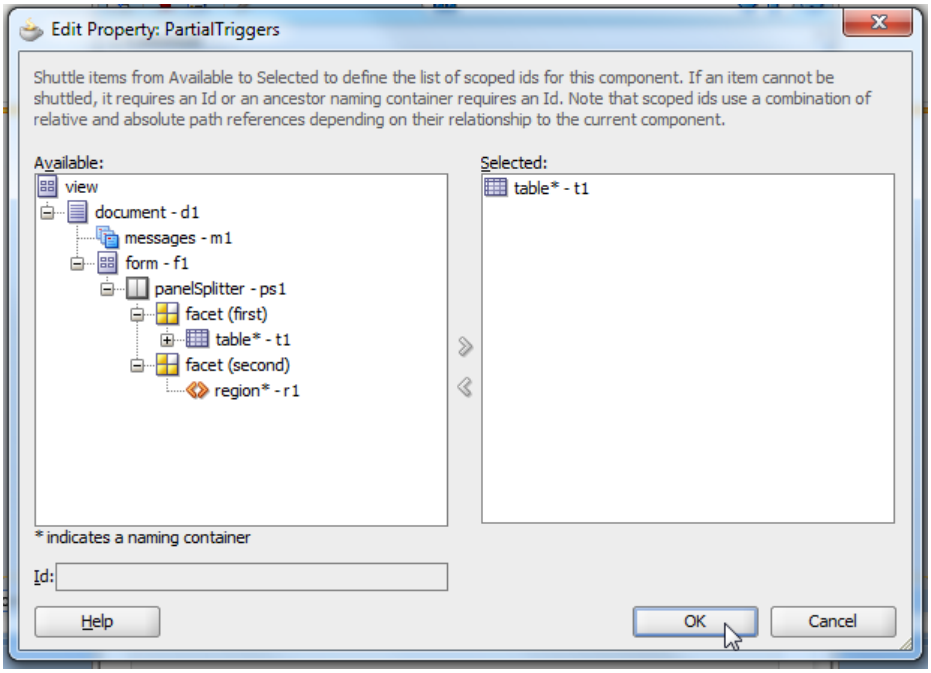


25. Navigate to the table component in the first panel facet of the Panel Splitter component

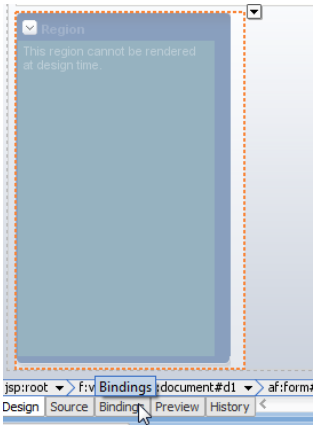
26. Select the table and move it to the "Selected" area

27. Ok the dialog

Note: Partial triggers in ADF Faces allow components to be notified when an event – like selection – is performed on another component. In response to this notification, the listening component is refreshed

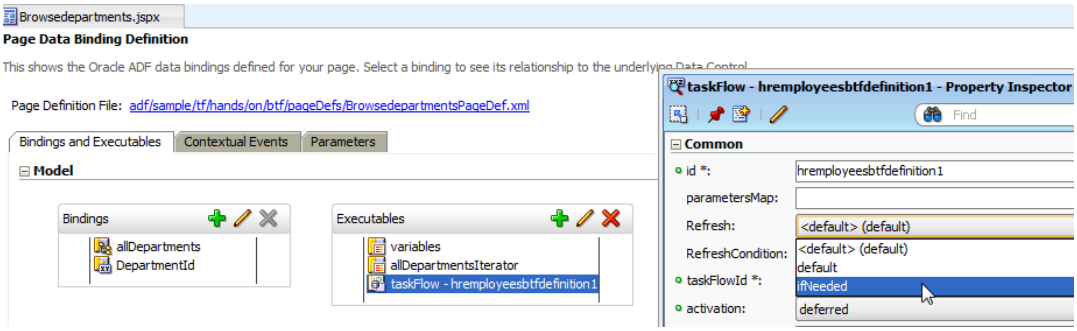


28. Click onto the "Bindings" tab at the bottom of the visual page editor



29. In the Binding editor, select the taskFlow entry in the "Executables" section and open the Property Inspector

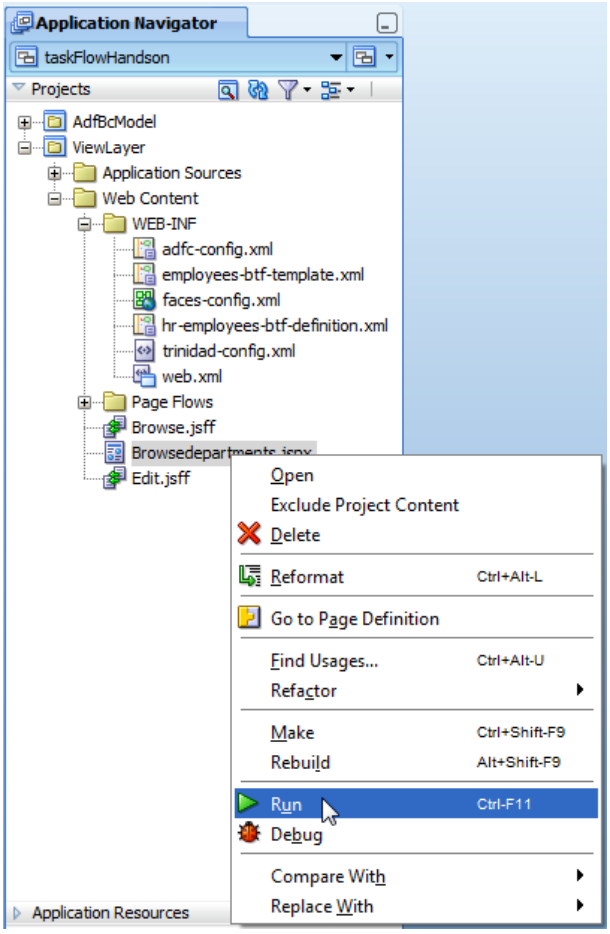
30. In the PropertyInspector set the "Refresh" property to "ifNeeded". This ensures the task flow is re-initialized when the input parameter value changes, which is the case when the user selects a different row in the table.



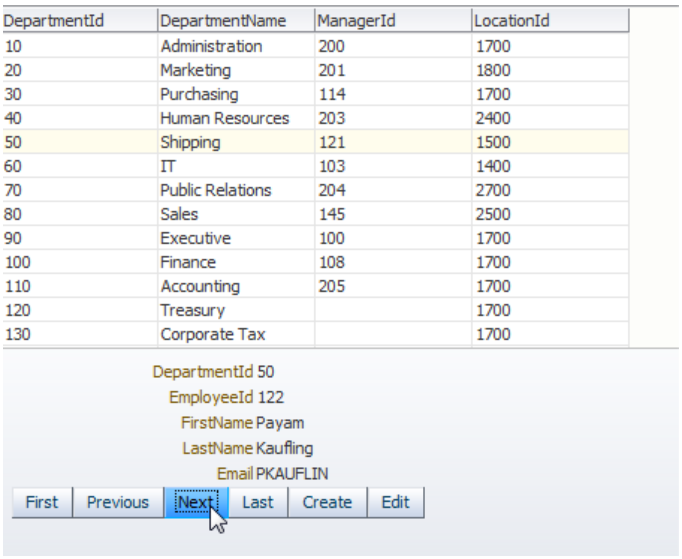
Running the Application

Run the JSPX page from the Application Navigator

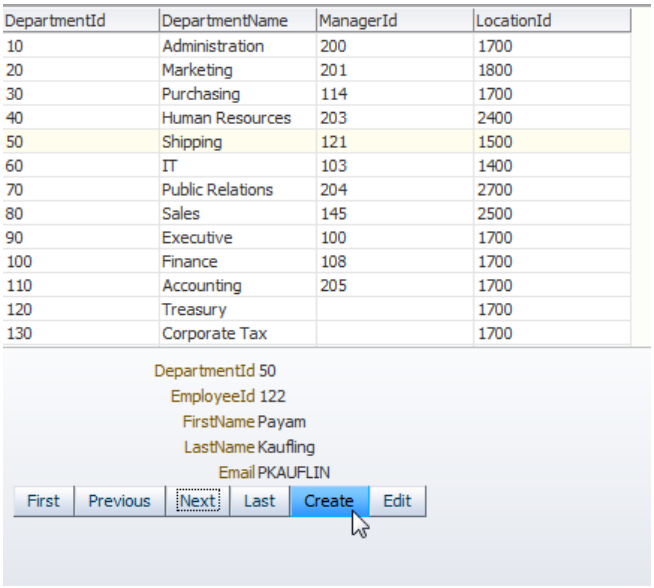
1. Use the "Run" option in the context menu to run the JSF application using the integrated WLS server



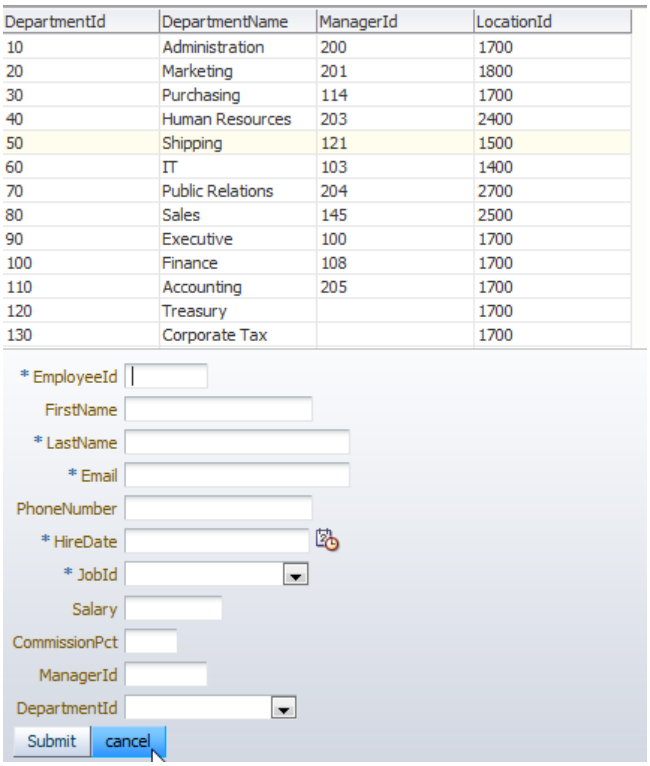
2. Select a Department in the departments table and scroll the list of employees



3. Press "Create" to create a new employee and navigate to the employee edit form



4. Note that the page does not need to do a full refresh. Instead, navigation is performed within the context of the region area only



5. Press "cancel"

Note: Pressing cancel suppresses field validation and navigates to the save-point restore activity. The restore activity sets the transaction state and the view state back to the time you pressed the "Create" button.

RELATED DOCOMENTATION

☒	
☒	
☒	