ADF Code Corner

72. Hands-on: Creating an ADF application based on an EJB WS using a JAX-WS proxy client and the POJO Data Control

twitter.com/adfcodecorner

Abstract:

In this hands-on, you build an ADF Faces rich client application querying and updating a JAX-WS Web Service created from an Enterprise Java Beans model using JSR 181 annotations and the Oracle ADF JavaBean binding. The application shows a master-detail behavior between departments and employees and allows you to update a selected department. If time is short, a completed solution is provided in the downloads

Duration 60 minutes.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
17-FEB-2011

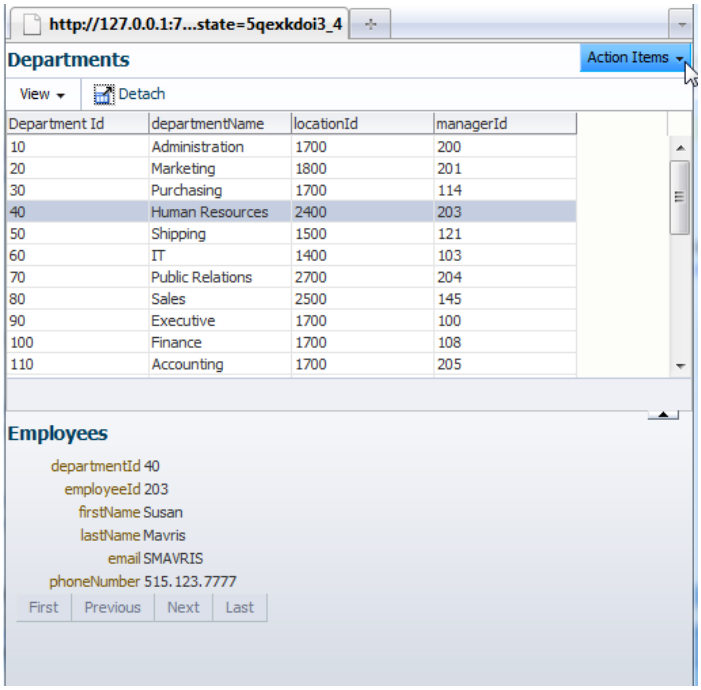
Introduction

There are three options for developers to integrate services in ADF applications

- Web Service Data Control
- Web Services Proxy Client
- URL Data Control

In this hands-on, a Web Services Proxy Client is created for a remote deployed Web Service. The proxy client is decorated by a JavaBean for adding pre- and post processing instructions and to ensure custom code is not impacted when the Web Service proxy client is re-generated. The ADF integration is realized by a JavaBean Data Control configured from the decorator JavaBean class.

Running the application, the following page is displayed



Prerequisite and Setup

This hands-on requires a database to be available that has the HR schema installed and enabled. The Oracle XE and Oracle enterprise database both have this schema available and no scripts need to be run.

The hands-on requires you to open the following two Oracle JDeveloper applications:

1. HumanResourceService
2. PrxyClientApp

The **HumanResourcesService** application contains an Enterprise JavaBean / JPA project that connects to the HR database schema to query and update the database using Eclipse Link. The EJB session bean is annotated to become a Web Service upon deployment. You use this workspace as the Web Service to access when building the ADF application on top of it.

Open Oracle JDeveloper and choose Application | Open from the file menu.

Navigate to the HumanResourcesService folder and select HumanResourcesService.jws

To change the RDBMS connection, select **View | Database | Database Navigator** from the Oracle JDeveloper menu. Select the database connection node in the " HumanResourcesService" node and use the right mouse context menu to edit its properties and change the connect information.

The **PrxyClientApp** workspace is the starter workspace for you to build the hands-on application. Open the workspace by choosing Application | Open in the Oracle JDeveloper file menu.

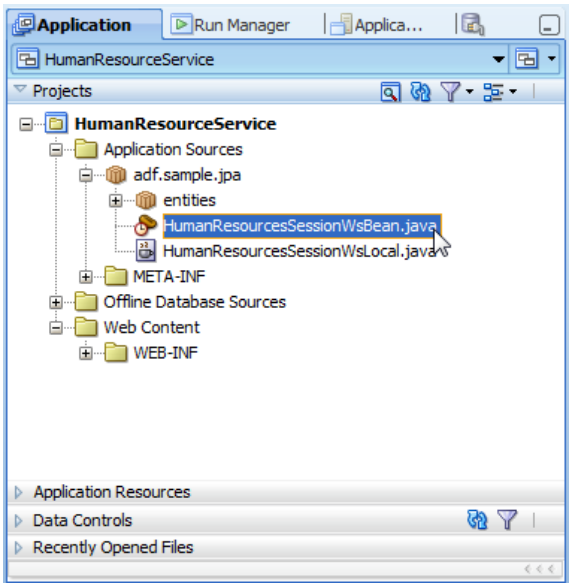
Navigate to the PrxyCleintApp folder and select HumanResourcesService.jws

Deploying and Testing the Web Service

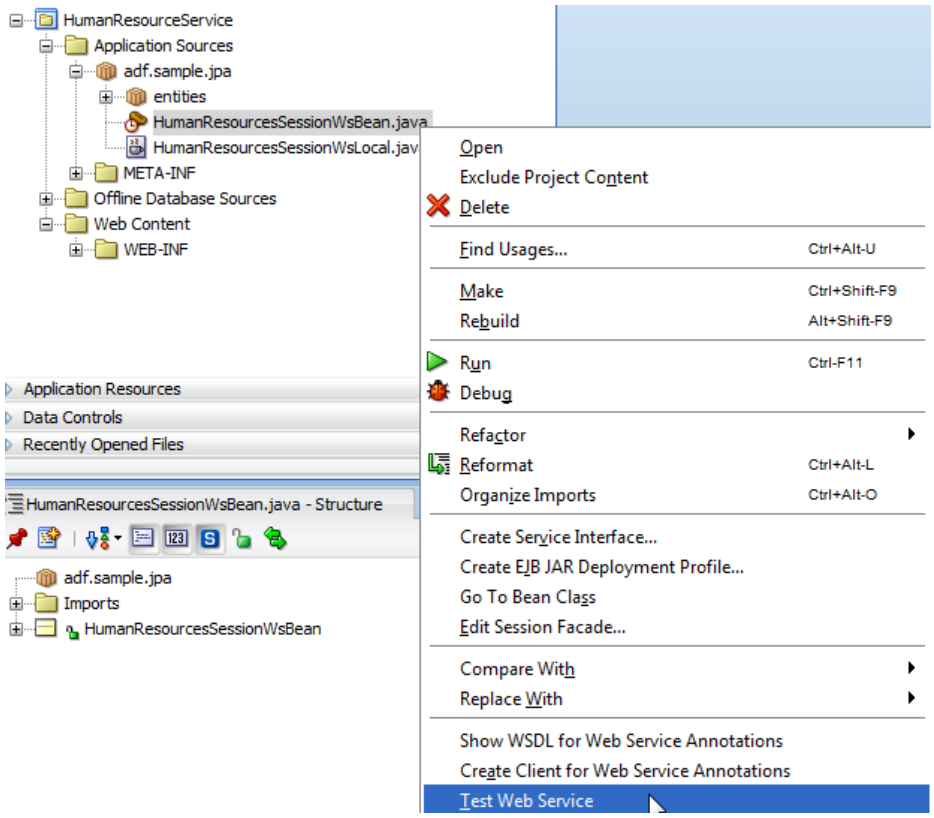
Before you get started building the application, verify the Web Service is working by running it in the Oracle JDeveloper Web Service tester. In addition, this step provides you with the WSDL reference you need to build the hands-on applications.

1. Select the "HumanResourceService" application in the Oracle JDeveloper selection dialog.

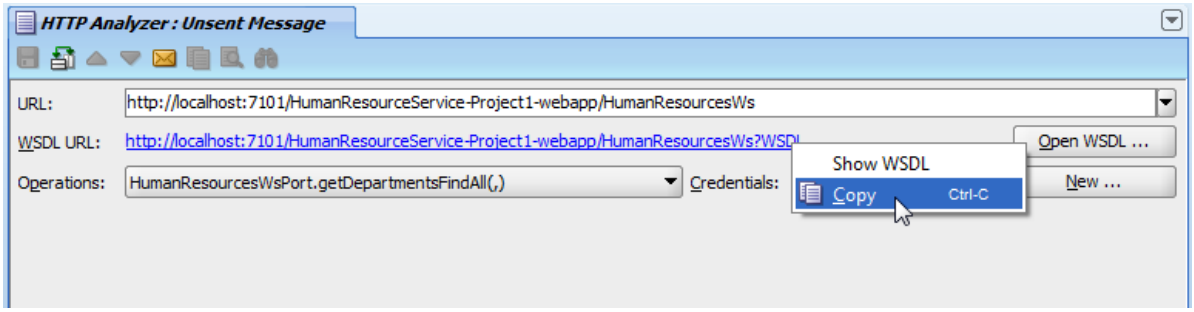
2. Expand the Application Sources | adf.sample.jpa folder and select the "HumanResourcesWsBean" Java file with the right mouse button



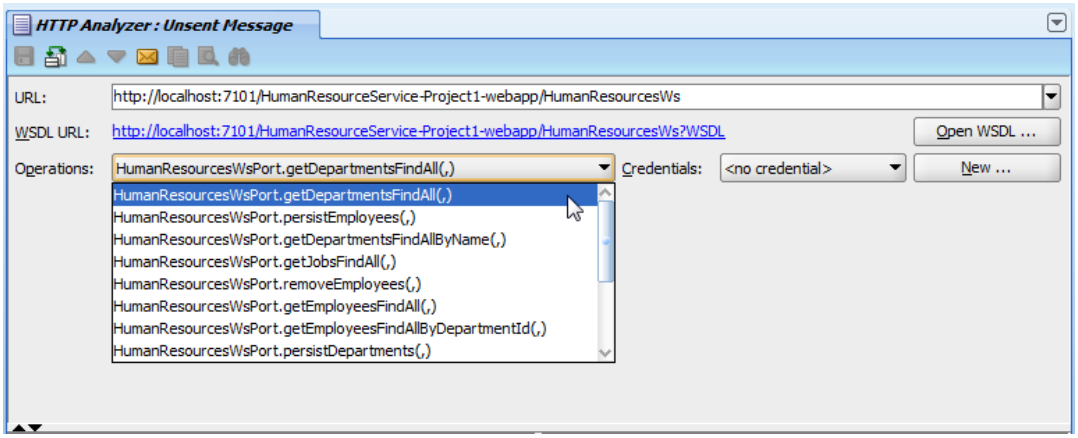
3. Choose the "Test Web Service" option from the opened context menu

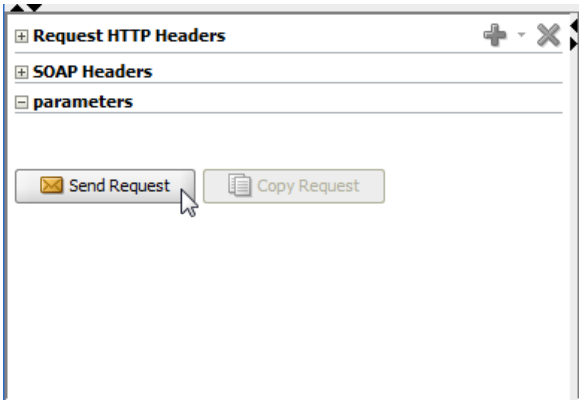


4. In the Web Service tester, click onto the WSDL URL with the right mouse button and choose "Copy" from the context menu

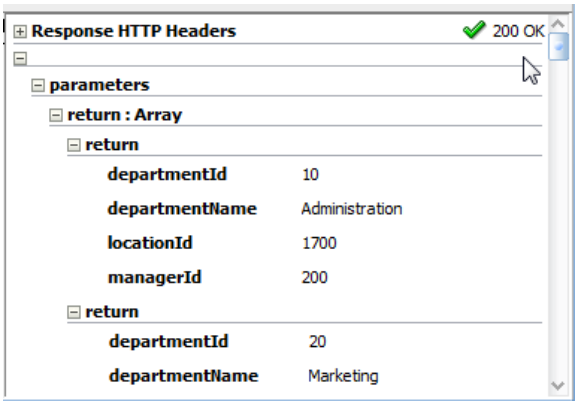


5. From the "Operations" list, select "getDepartmentsFindAll" and press the "Send Request" button





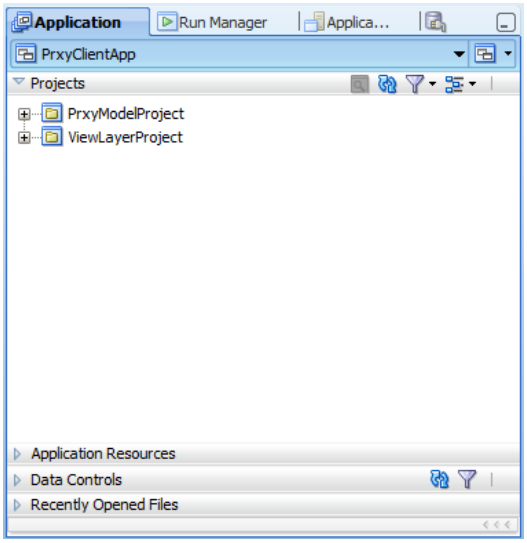
5. If all is well a list of departments is printed as shown below



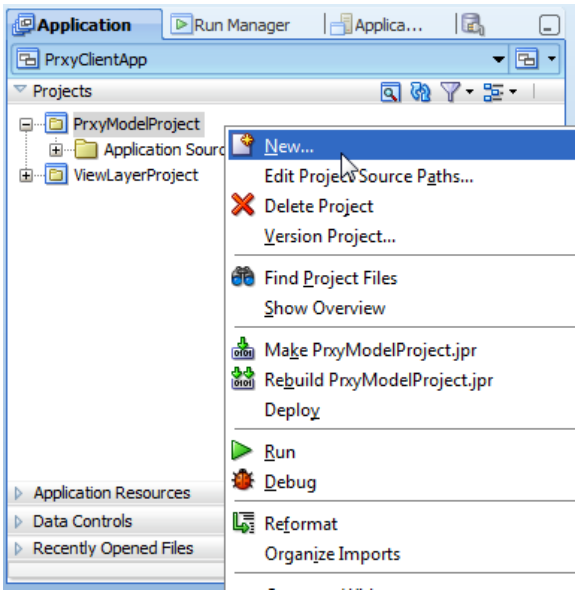
6. Close the Web Service tester window.

Creating the WS Proxy Client model

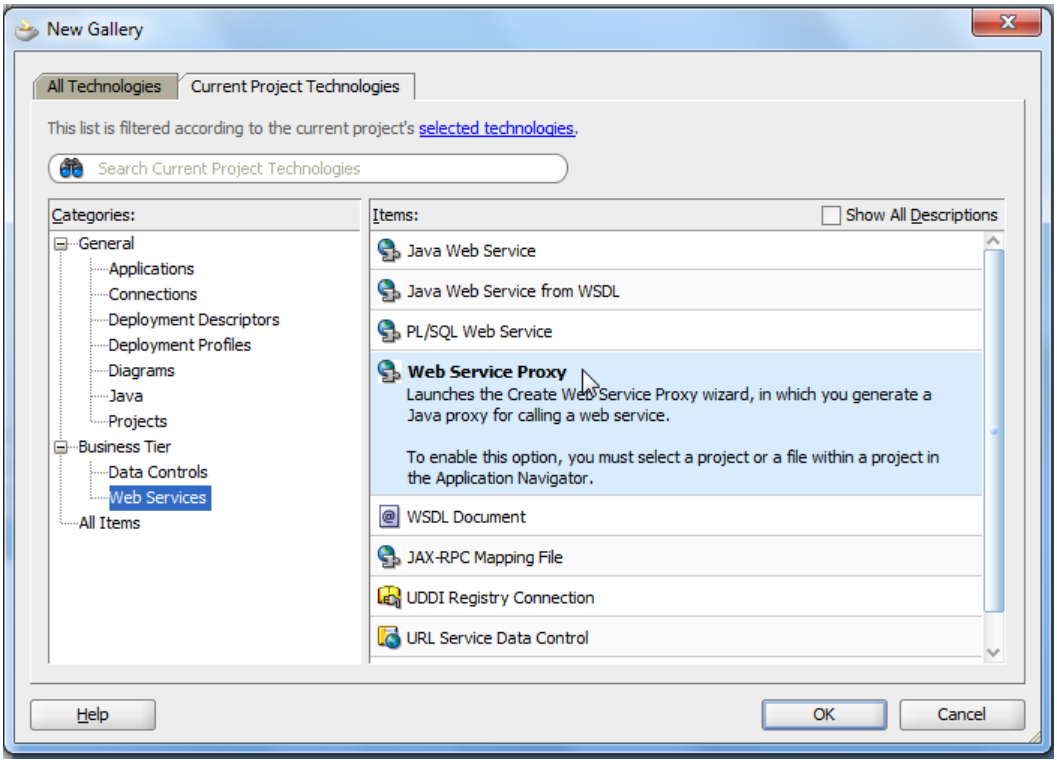
1. Select the "PrxyClientApp" in the Oracle JDeveloper Application Navigator selection list.



2. Select the "PrxyModelProject" and choose "New" from the right mouse context menu

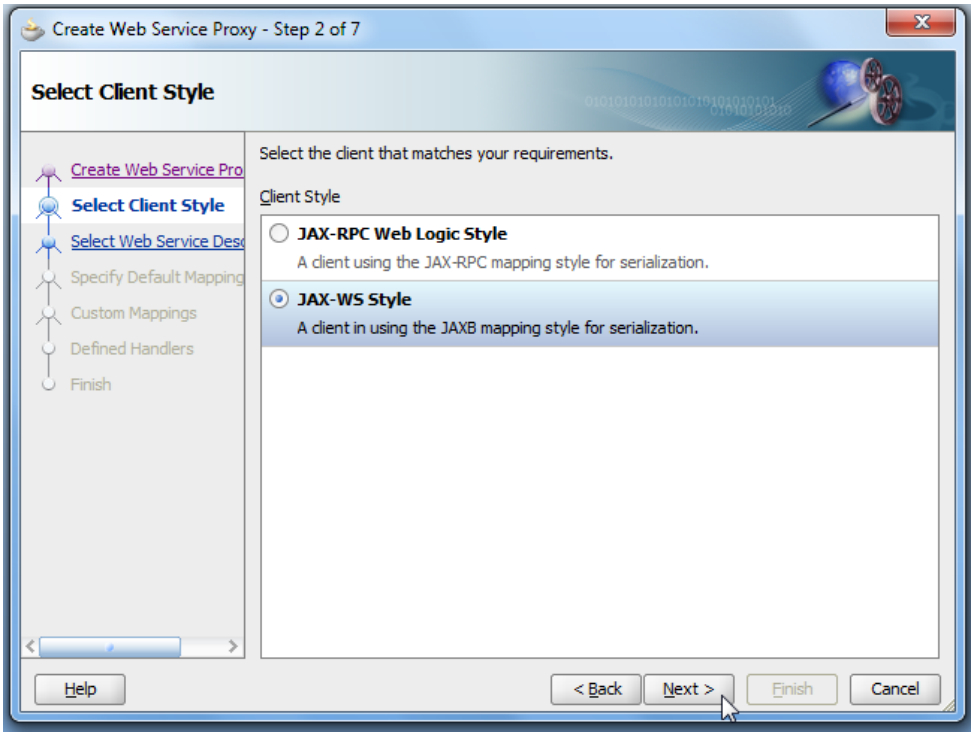


3. In the opened New Gallery window, choose "Web Service Proxy" in the select item list of the Business Tier | Web Services node.

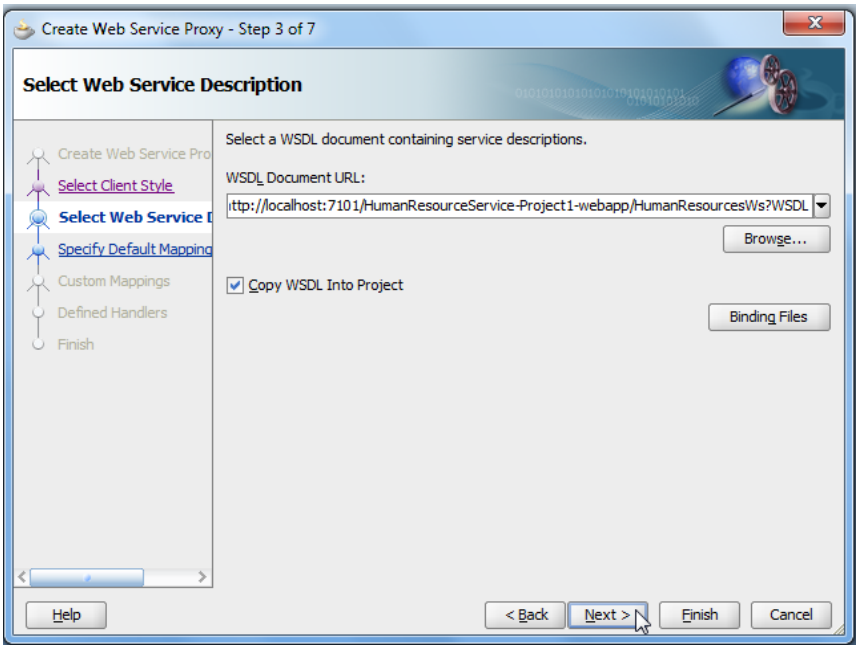


4. Press OK

5. In the "Create Web Service Proxy" dialog, select the JAX-WS Style option and press "Next"



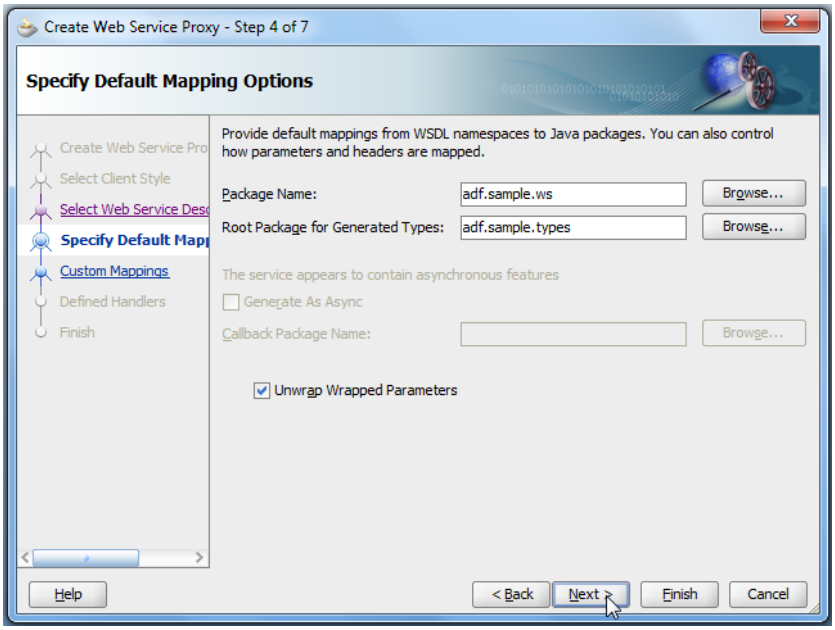
6. Paste the WSDL URL into the WSDL Document URL. The WSDL URL is what you copied to the clipboard in the first part of this hands.-on



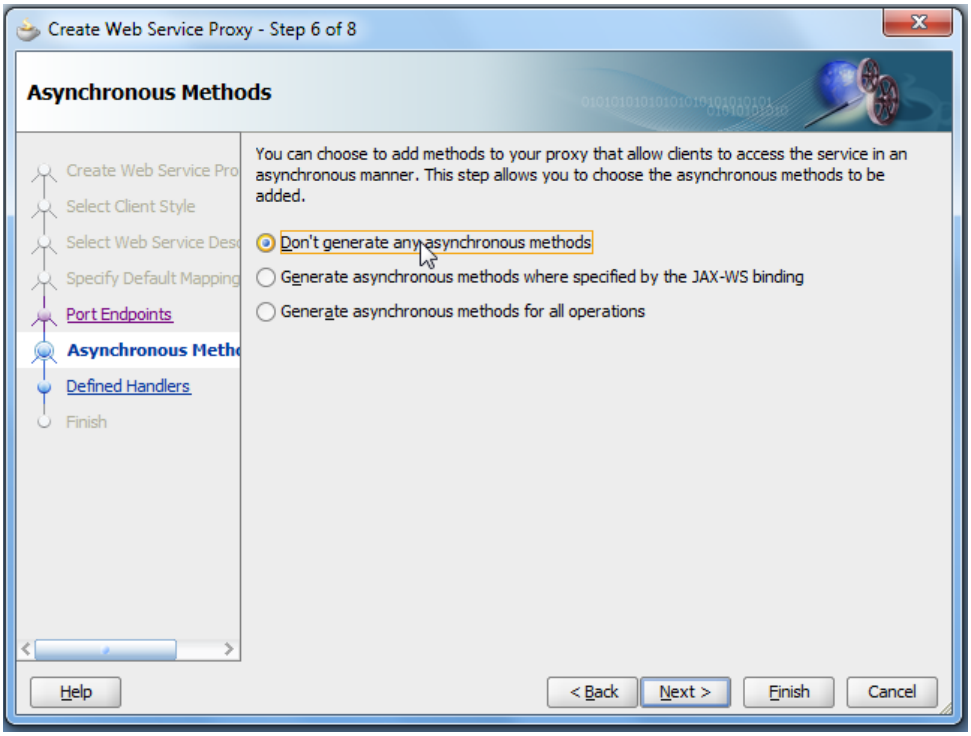
7. Press Next

8. Define the following values in the dialog fields

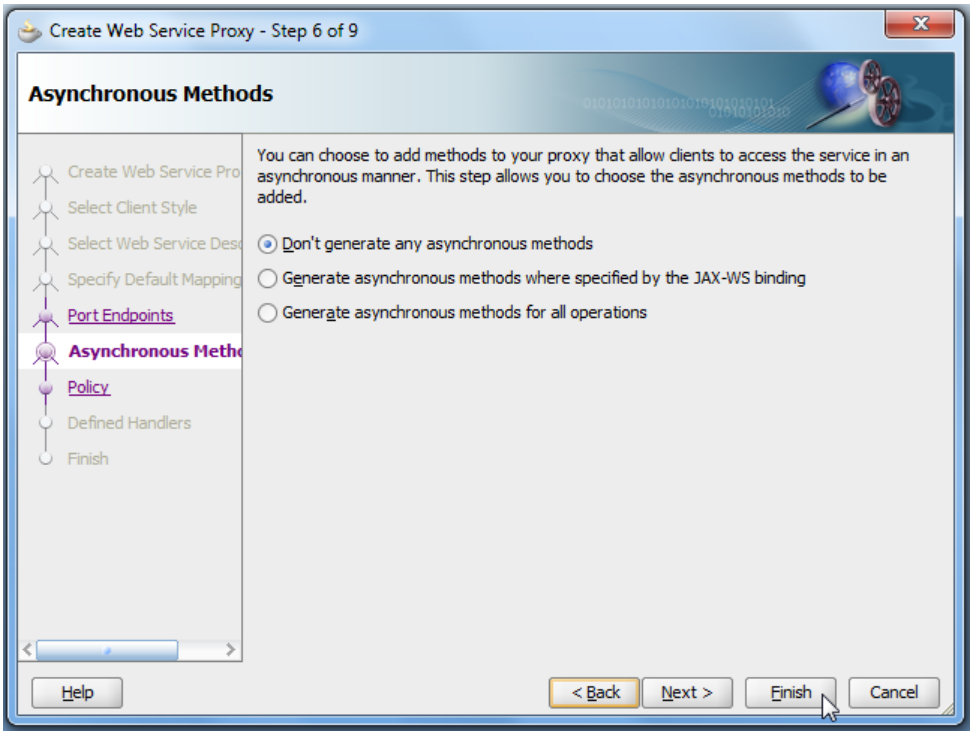
Package Name	adf.sample.ws
Root Package for Generated Types	adf.sample.types



9. Press "Next"

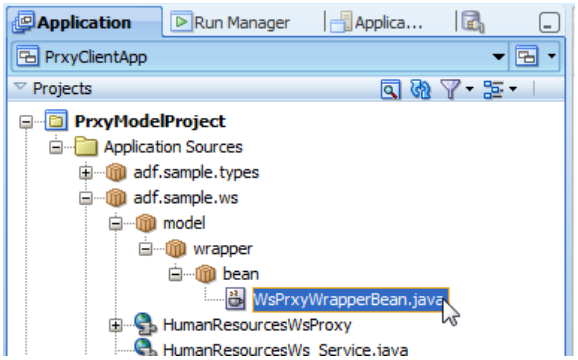


10. Select the "Don't generate any asynchronous methods" option



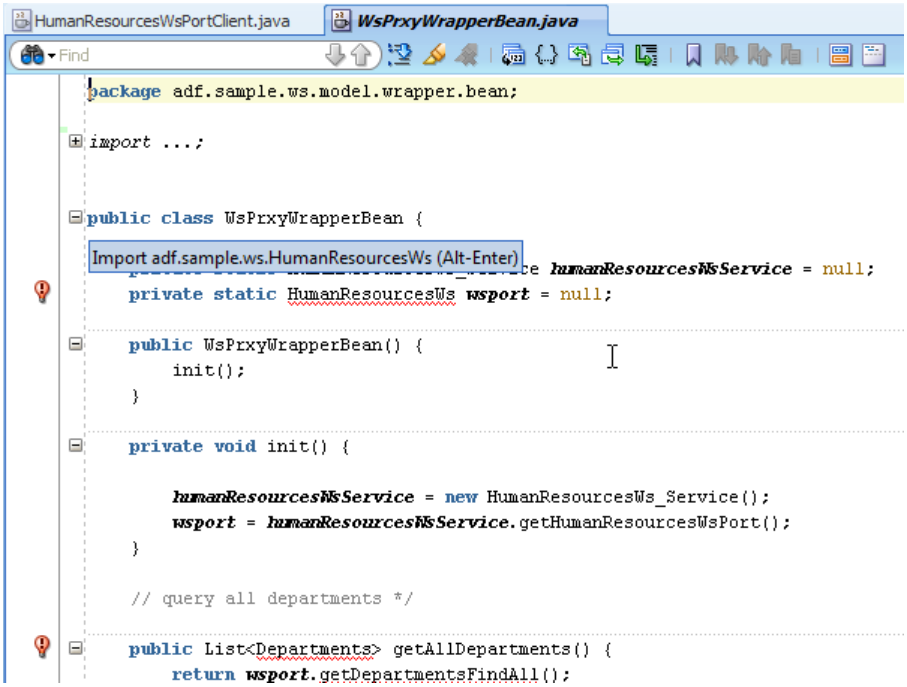
11. Press "Finish"

12. In the Oracle JDeveloper Application Navigator, expand the adf.sample.ws | model | wrapper | bean package and double click on the "WsPrxyWrapperBean.java" file entry.



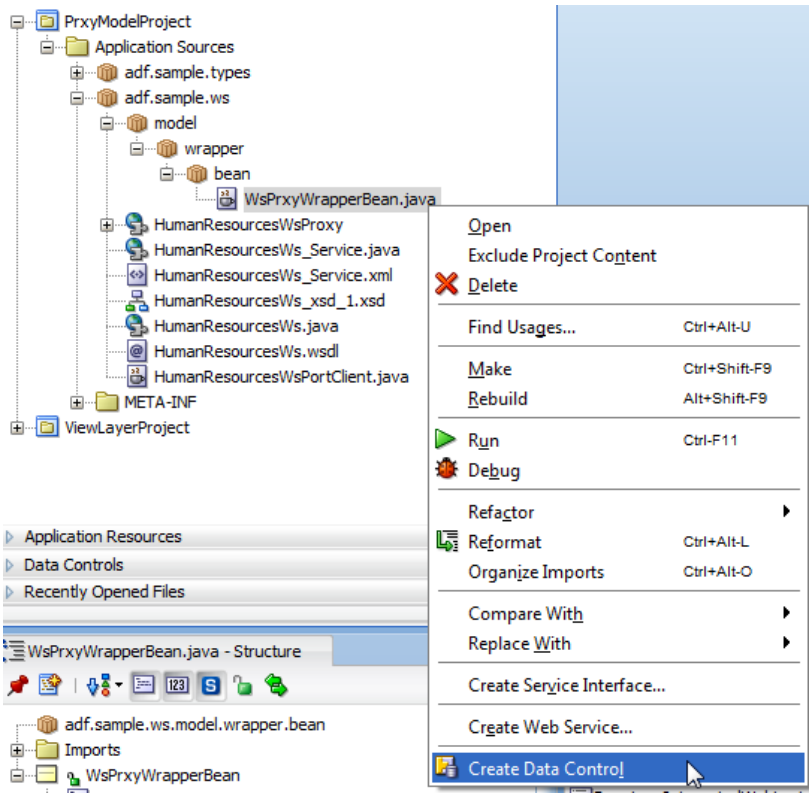
13. The "WsPrxyWrapperBean.java" is a bean that is pre-created to wrap the generated Web Service proxy client. In a next step, you create the JavaBean Data Control from this class. Before however, the Java file needs to be completed with the import statements for the Web Service proxy client class.

14. Place the cursor on the code lines that have a red underline. When the blue import tooltip pops up, press **alt+Enter** to create the import statement

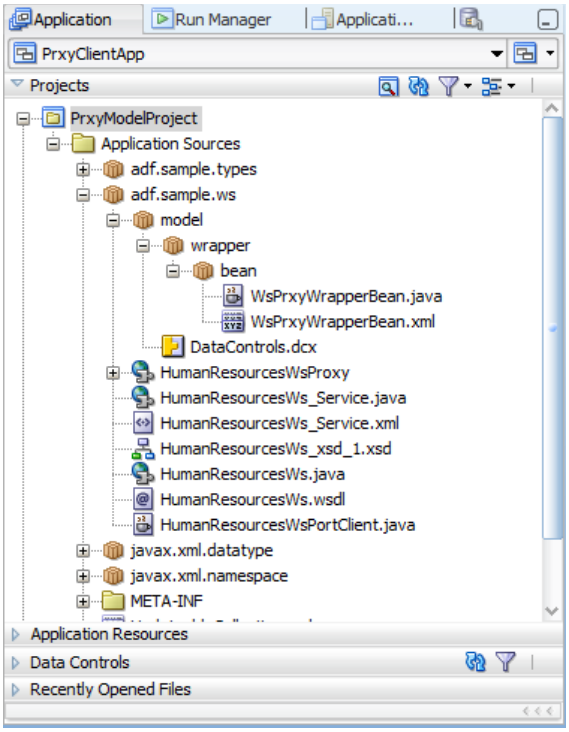


15. The WsPrxyWrapperBean.java class contains all the methods that are needed to build the ADF application. Save the Java file and close the Code Editor.

16. Select the " WsPrxyWrapperBean" entry in the Application Navigator and choose "Create Data Control" from the right mouse context menu



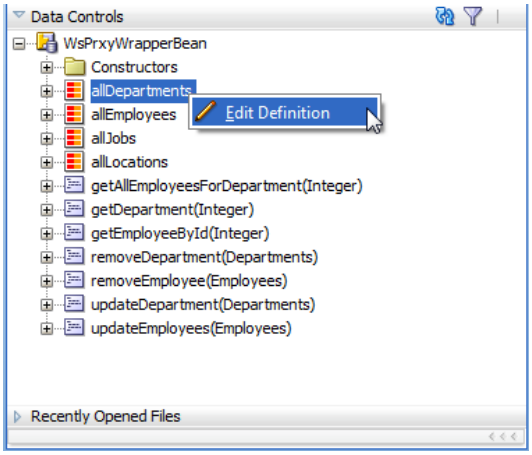
17. After Data Control creation, the "PrxyModelProject" looks at shown below



18. The Data Control definition can be further edited to create consistent – translatable – labels and tooltips, define Primary Key attributes or define validation rules.

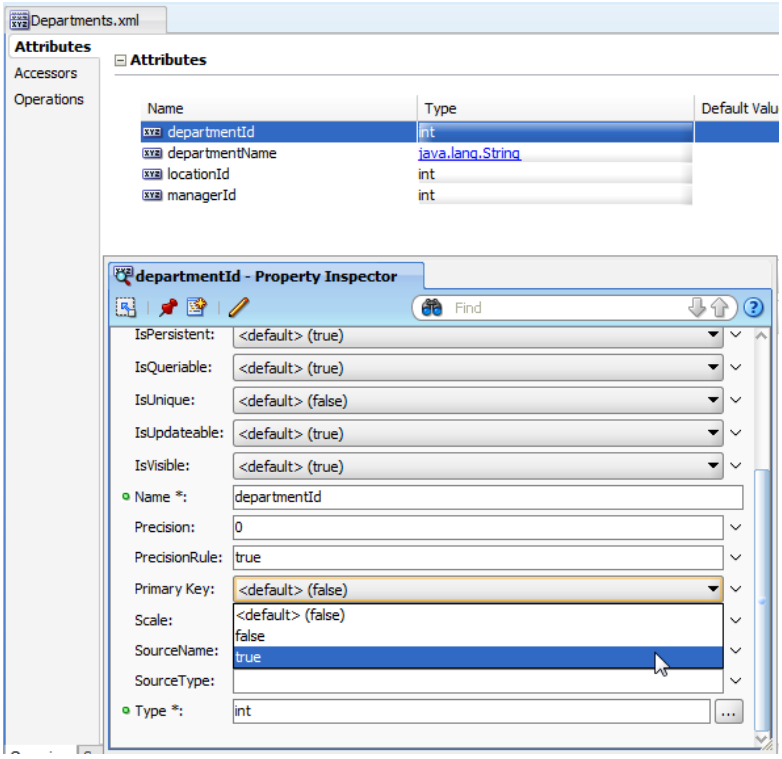
19. Expand the DataControls accordion and click the blue refresh icon if the Data Control definition is not shown

20. Select the "allDepartments" entry and choose "Edit Defintion" from the right mouse context menu.

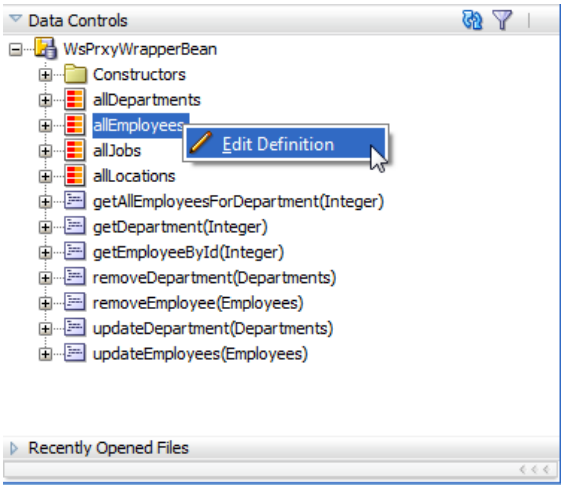


21. In the opened dialog, select attributes, like "departmentId" and open the Property Inspector (ctrl+shift+I)

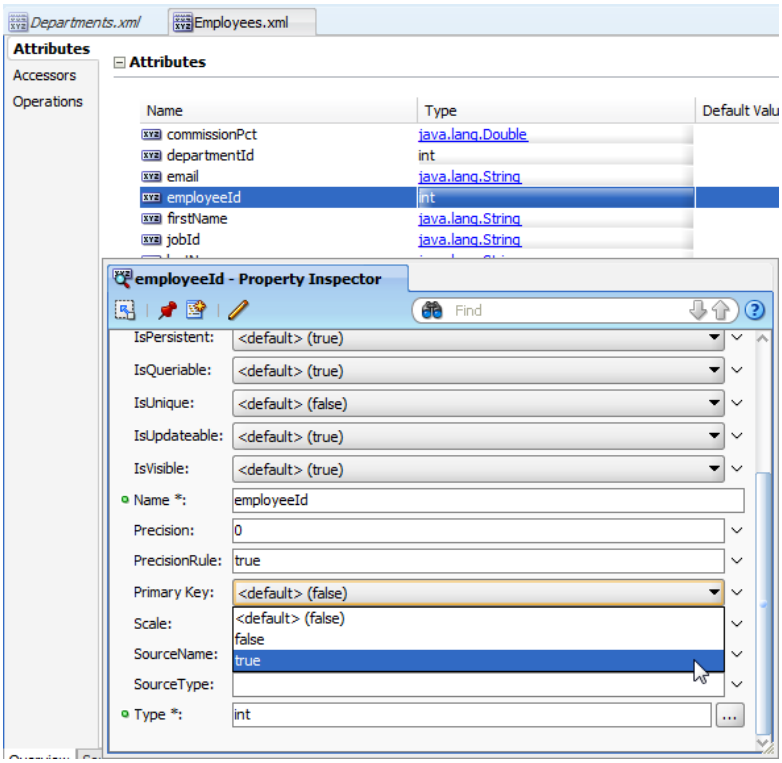
22. Select the "Primary Key" property and set its value to true. This defines the departmentId attribute as a key attribute, which is required for building the row key in the ADF iterator



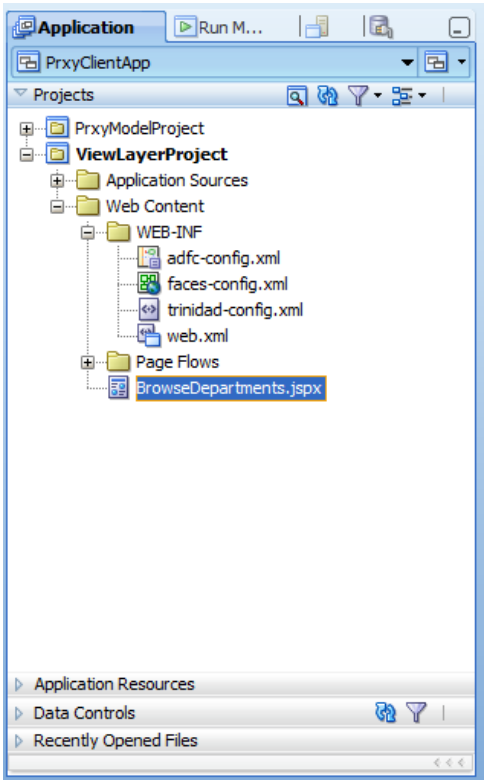
23. Select "allEmployees" in the Data Controls panel and choose "Edit Definition"



24. Select the "employeeId" and open the Property Inspector. Set the Primary property to "true"



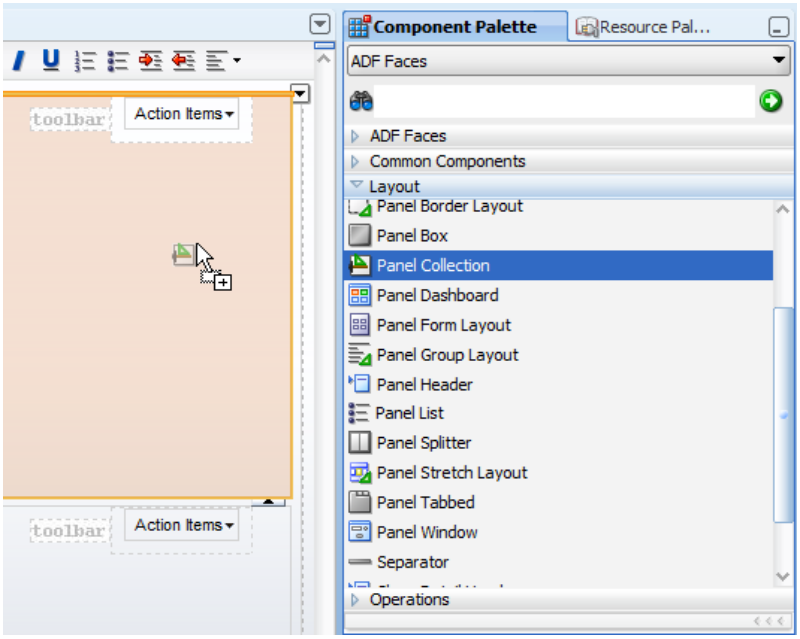
25. Repeat the above steps for "allJobs" (set jobId as the Primary Key) and "allocations" (set locationId as the Primary Key)



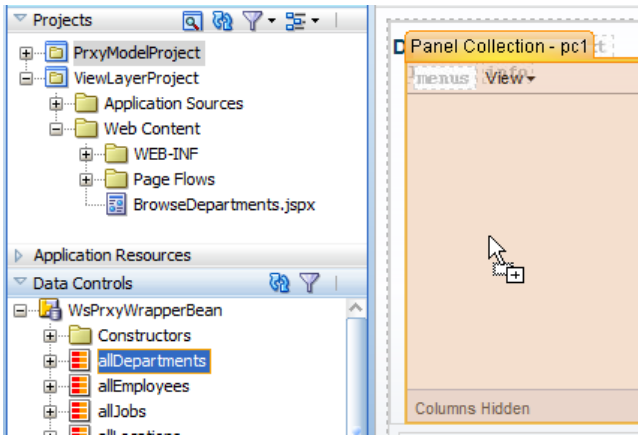
26. Select the BrowseDepartments.jspx page and double click it to open the visual page editor

27. Open the Component Palette (ctrl+shift+P)

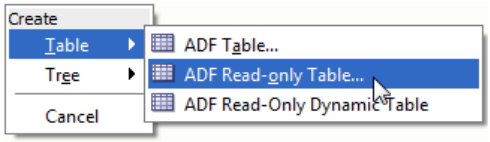
28. Drag the "Panel Collection" from the Layout section into the top facet of the Panel Splitter on the page



29. Drag and drop the "allDepartments" collection entry from the Data Controls panel to the Panel Collection you just added

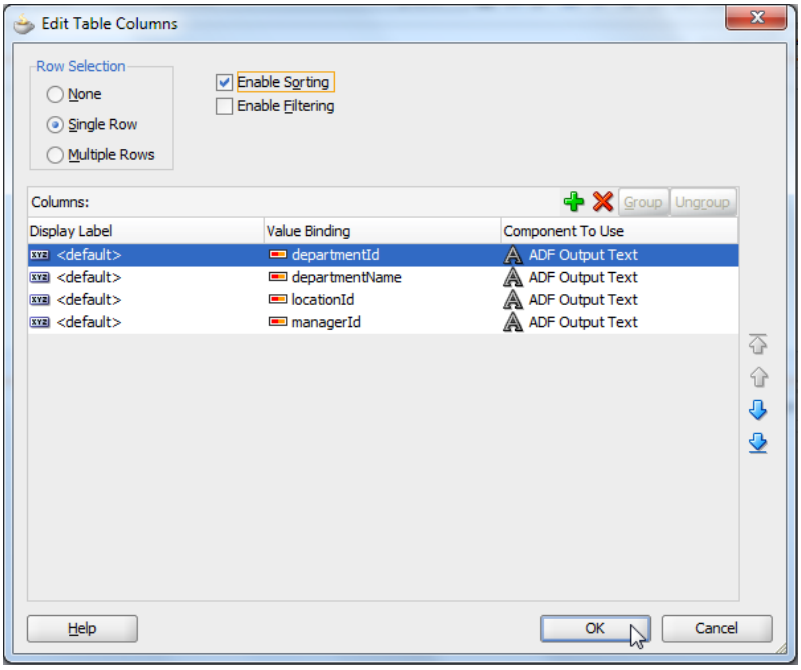


30. Choose Table | ADF Read-only Table from the context menu

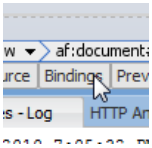


31. In the table configuration dialog, enable sorting and select the "Single Row" select option

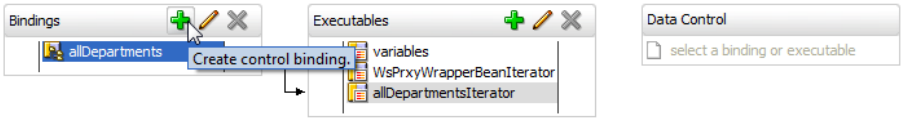
32. Press OK to close the dialog



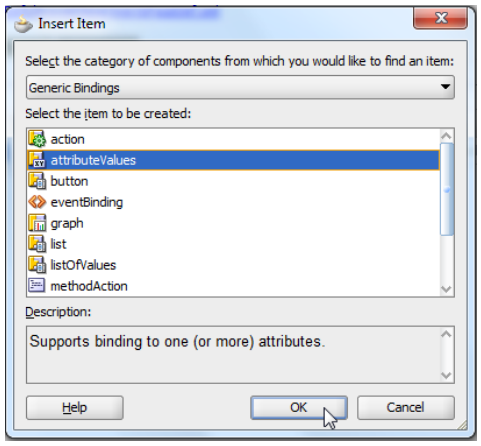
33. To reference the selected department row from the WS method that queries the employees child data, you need to create an attribute binding for the "allDepartments" departmentId Primary Key attribute. For this, click the "Bindings" tab at the bottom of the visual page editor.



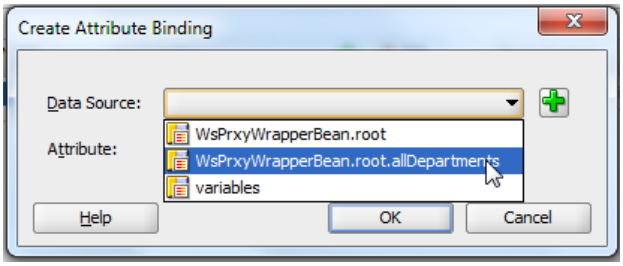
34. In the binding editor, press the green plus icon to create a new attribute binding



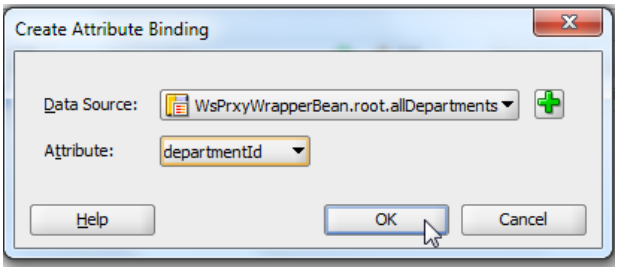
35. Select the "attributeValue" biding entry and press OK



36. In the "Data Source" select box, choose the "WsPrxyWrapperBean.root.allDepartments" entry. It's the entry that got created when dragging the "allDepartments" collection as a table to the page.

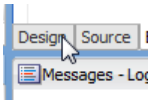


37. Select the "departmentId" attribute as the attribute to be exposed by the binding



38. Press OK

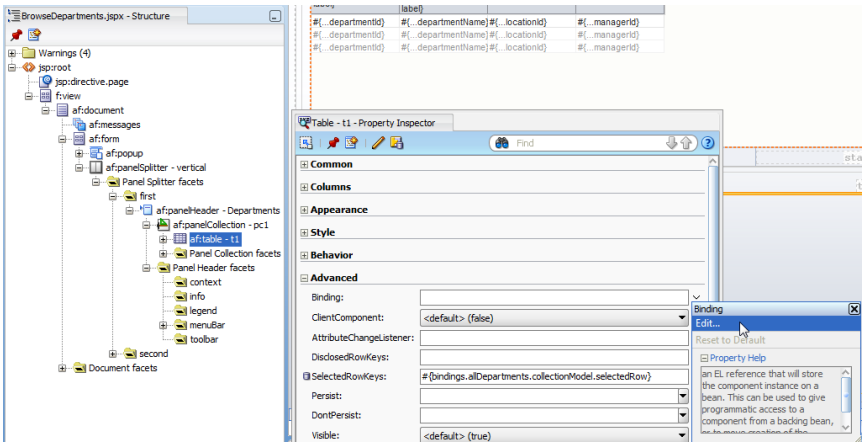
39. Click the "Design" tab at the bottom of the binding dialog to return to the visual page editor



40. Select the table component in the Structure Window (or the visual editor) and open the Property Inspector (ctrl+shift+I).

41. In the Property Inspector, select the "Binding" property.

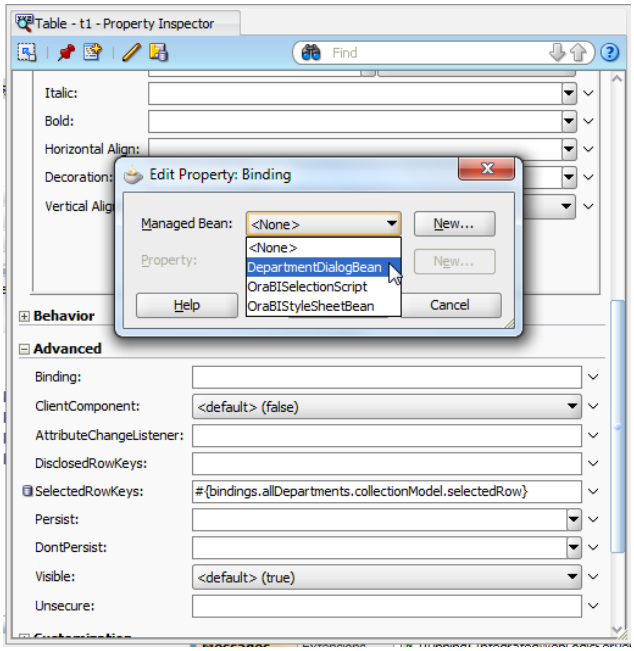
42. Click the arrow icon at the end of the "Binding" property field



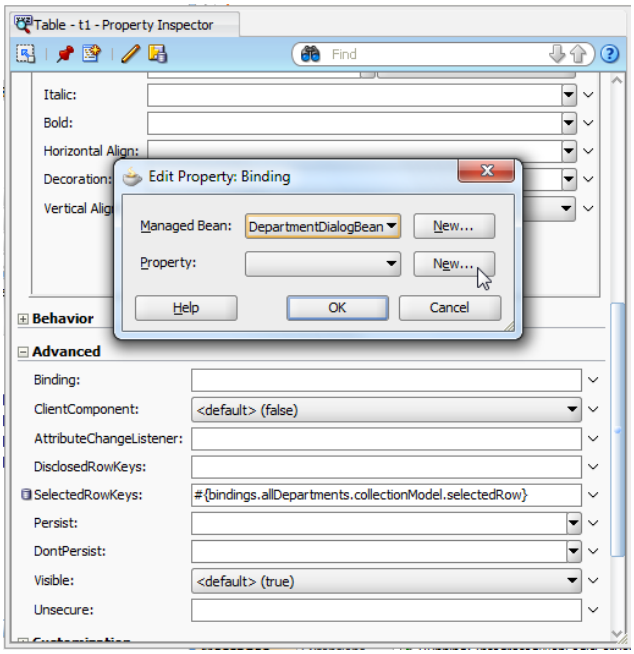
43. Select the "Edit" option from the context menu

44. Select the pre-created "DepartmentDialogBean" in the "Managed Bean" field.

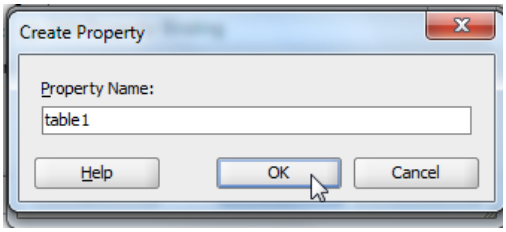
Note: The DepartmentDialogBean contains code to update or cancel the update of a selected Department row.



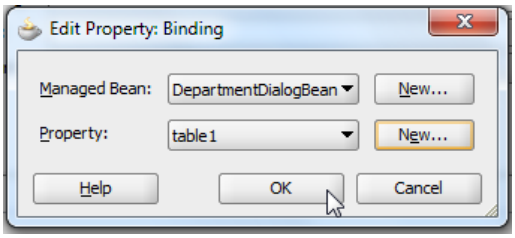
45. Press the "New" button to create a new setter/getter pair for the table component binding



46. Type "table1" as Property Name

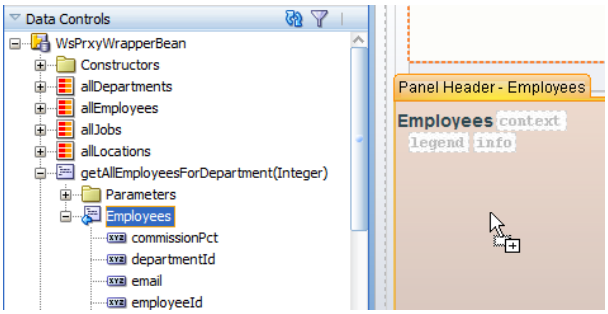


47. Pres OK

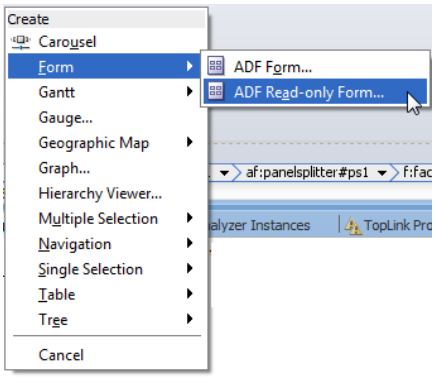


48. Press OK to close the Edit Property Binding dialog.

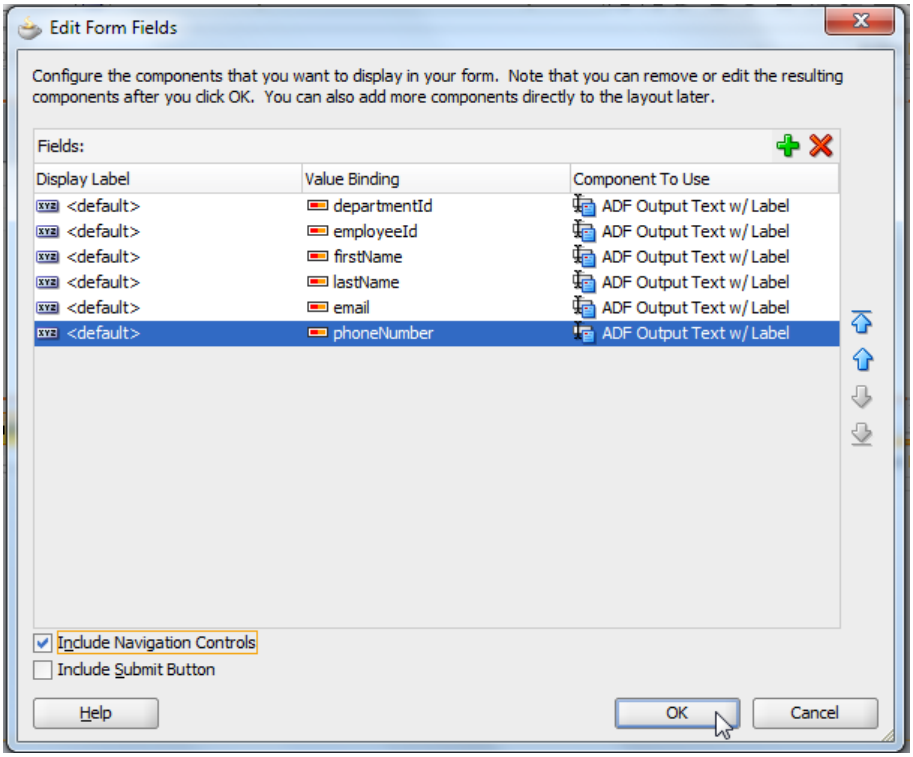
49. To create the Employee detail relationship for the Departments collection, drag and drop the "Employees" result of the "getAllEmployeesForDepartment(Integer)" to the lower facet of the Panel Splitter contained in the page



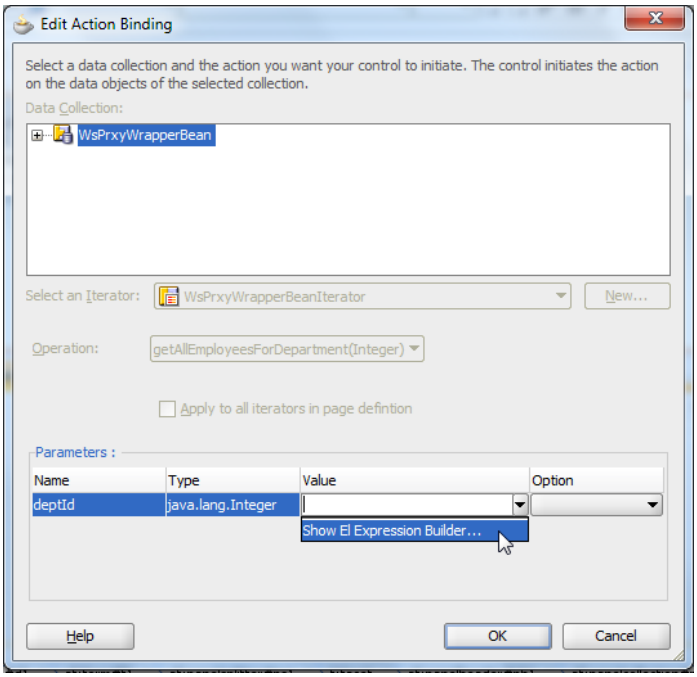
50. Choose ADF Read-only Form from the opened context menu



51. In the "Edit Form Fields" editor, remove all attributes, **except** of "departmentId", employeeId, firstName, lastName, email and phoneName.



52. Select the "Include Navigation Control" checkbox. This creates a navigation button bar for you to navigate the employee detail data

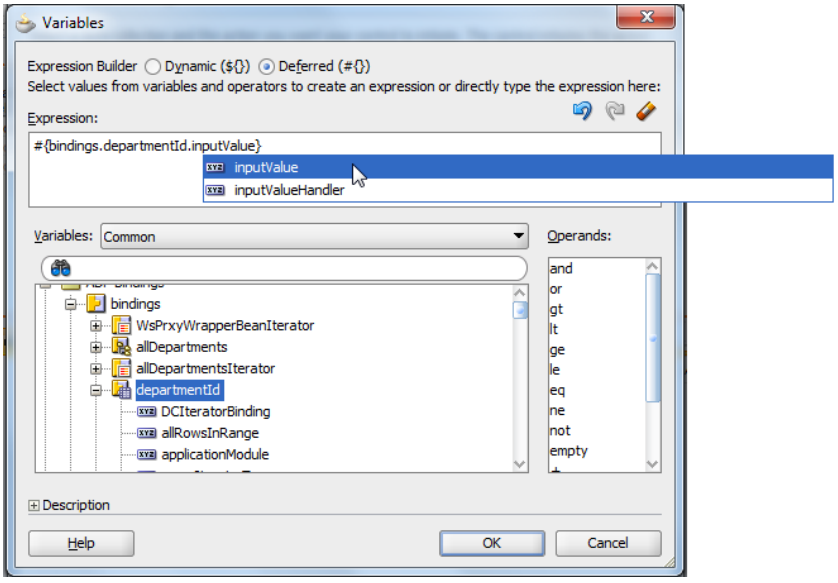


53. To create the Departments / Employee dependency, the department Id of the selected department is passed to the " getAllEmployeesForDepartment" method

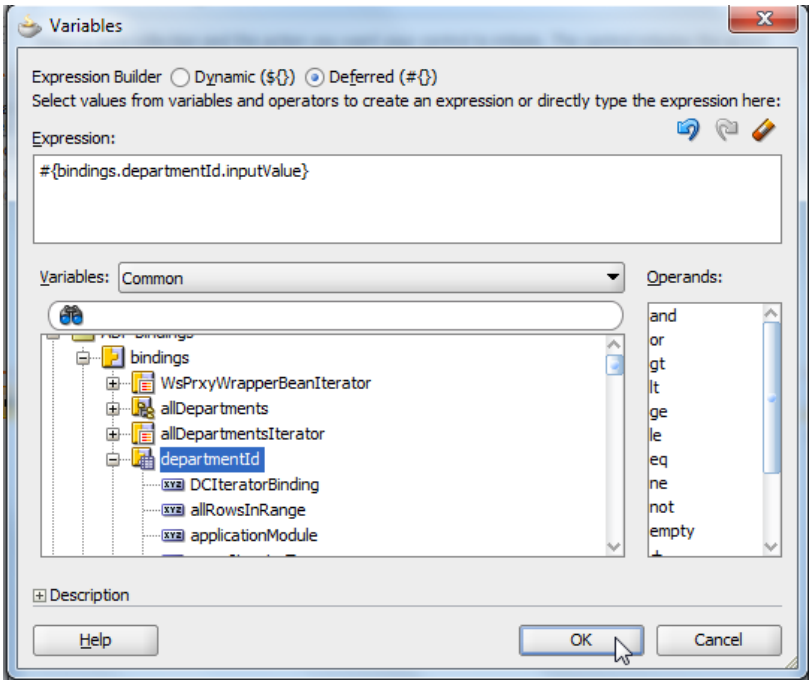
54. In the Value property field, select "Show EL Expression Builder" to browse the PageDef file for the "departmentId" attribute binding defined earlier

55. In the Expression Builder, select the "departmentId" attribute under the "bindings" node

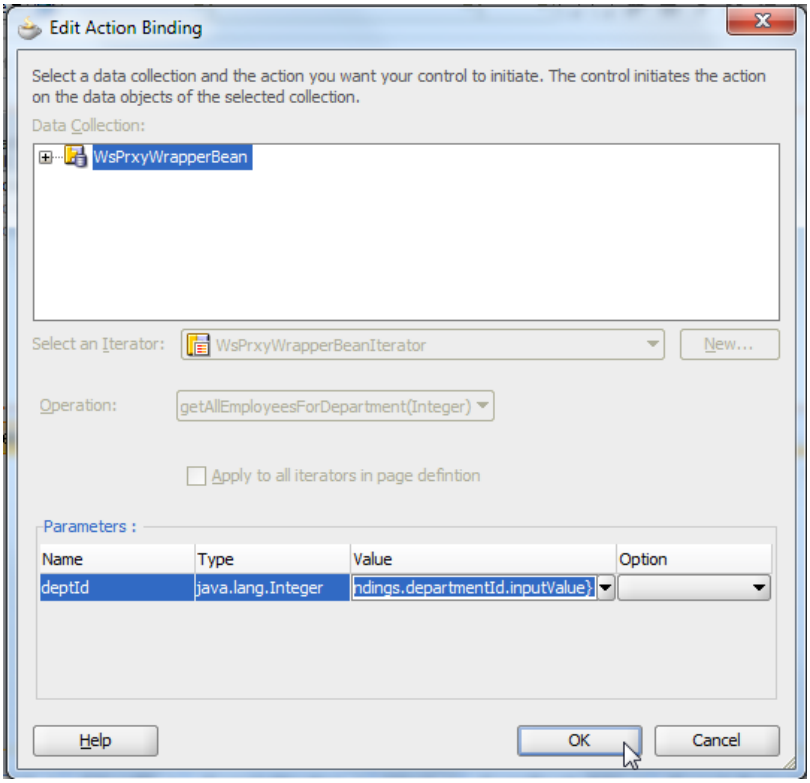
56. Complete the expression by appending "inputValue" to the departmentId attribute



57. The EL string should look as shown below

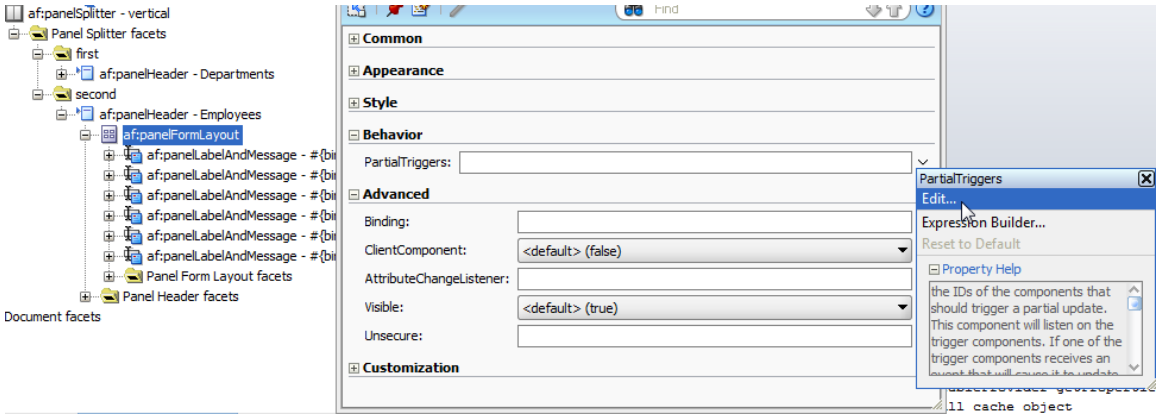


58. Press OK



59. Press OK to close the dialog

60. When the row currency in the department table changes, the employees form needs to be refreshed. For this you set up PPR between the table and the form



61. In the Structure Window, select the Panel Form Layout component that contains the read only form fields

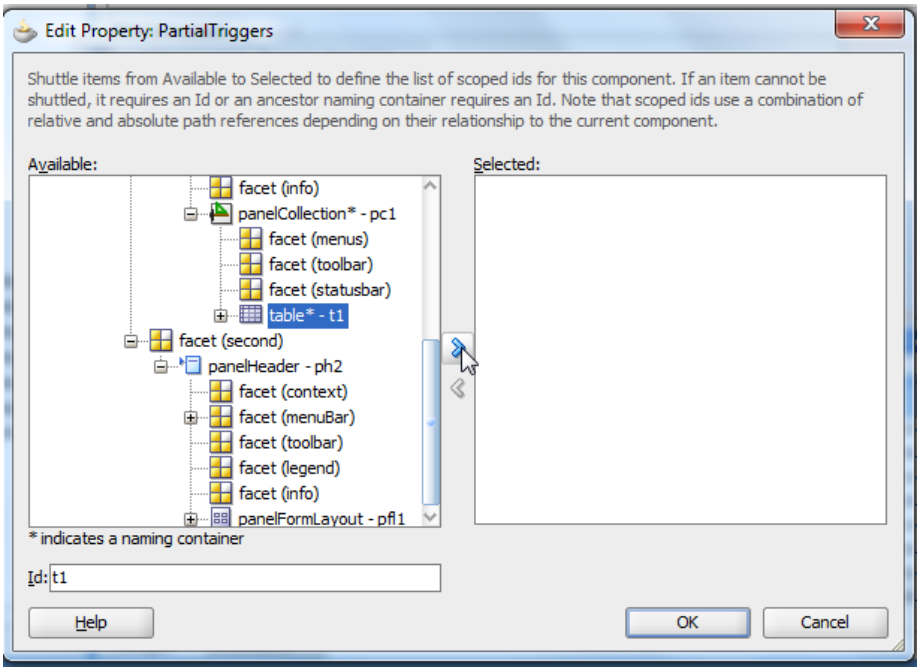
62. Open the Property Inspector

63. Select the Partial Triggers property and click the arrow icon at the end of the field

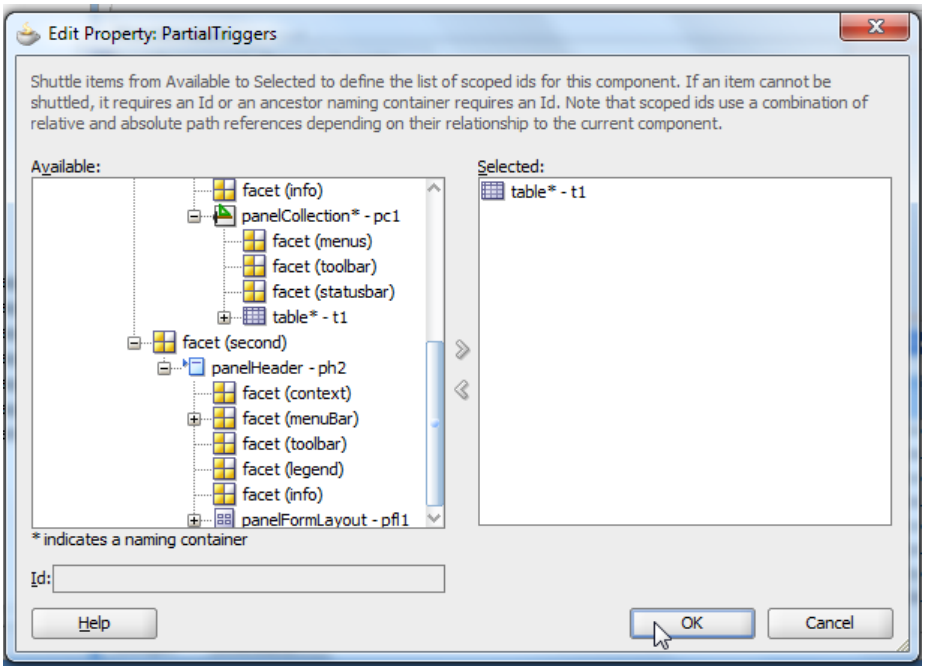
64. Choose "Edit" in the opened context menu

65. Browse the page structure in the "Available" list and select the "table" component contained in the panelCollection.

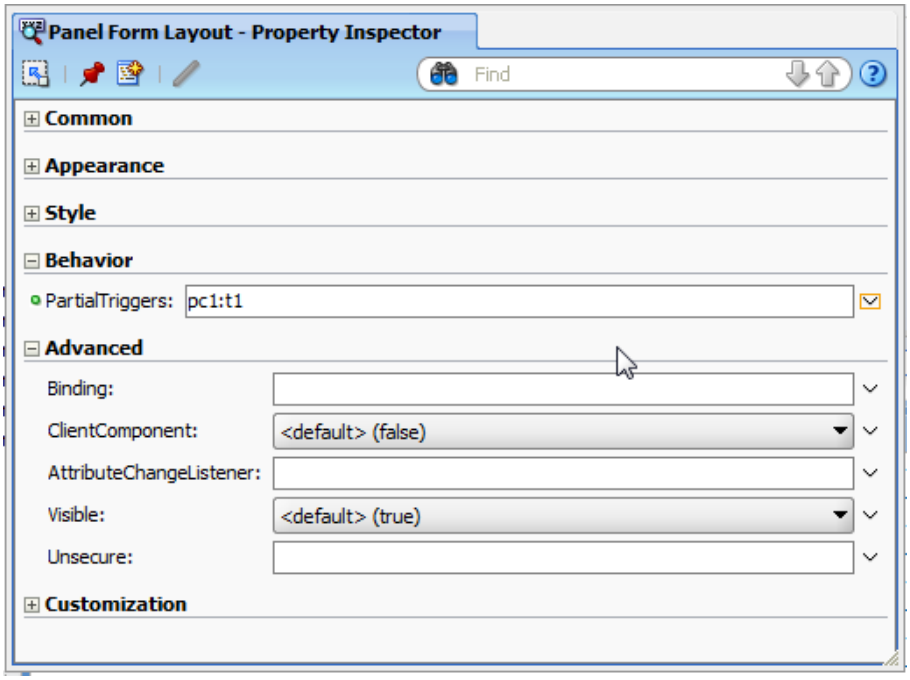
66. Shuttle the table to the "Selected" list



67. Press OK



68. The "Partial Triggers" property value is shown below

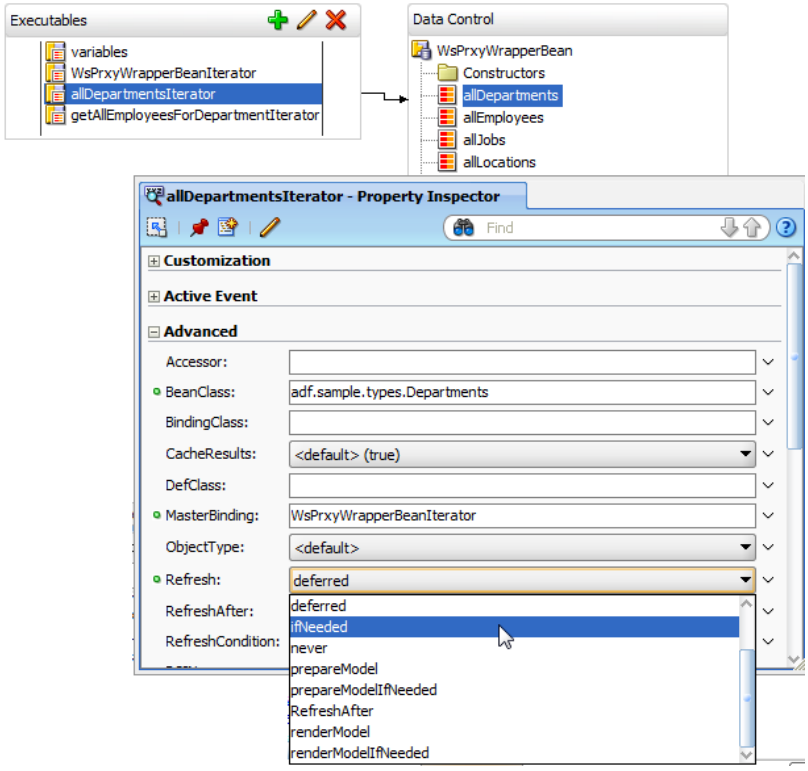


69. Open the visual page editor (if not already opened) and select the "Bindings" tab at the bottom of the page.

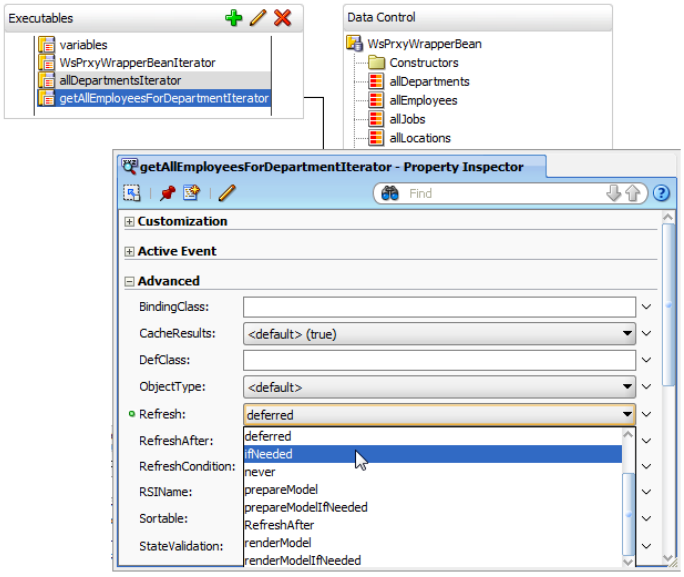
70. To ensure master-detail synchronization to work, the ADF iterator refresh option needs to be set to "ifNeeded" for both iterators. The default "deferred" option does not allow the synchronization to happen in time.

71. Select the "allDepartmentsIterator" and open the Property Inspector

72. Select "ifNeeded" in the "Refresh" option select list



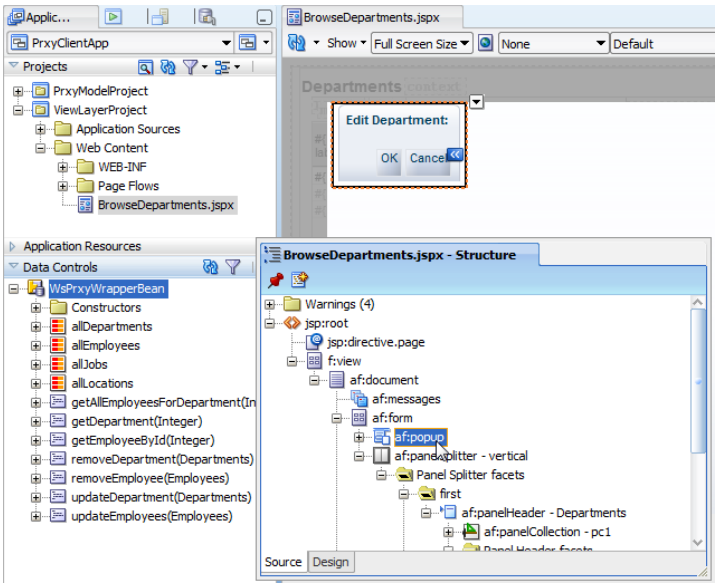
73. Select the "getAllEmployeesForDepartmentIterator" and set "ifNeeded" as the "Refresh" property value



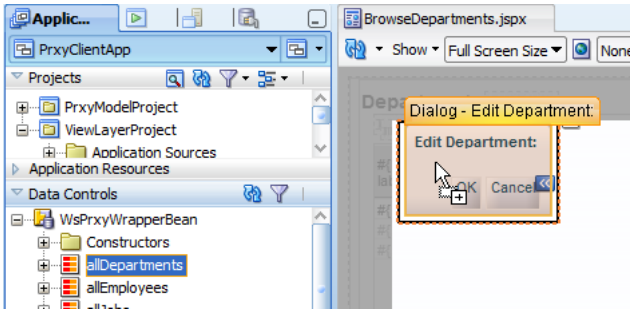
What you've done so far: So far, you created a Web Service Client proxy class from a Web Service WSDL reference. You create a master/detail UI relationship between a table displaying departments and a read only form showing employee data. Next you are going to create an update form for departments.

74. The update form will be launched in a popup dialog, which at runtime you start from a menu item. Select the BrowseDepartments.jspx page in the Oracle JDeveloper Application Navigator and open the Structure Window (ctrl+shift+S)

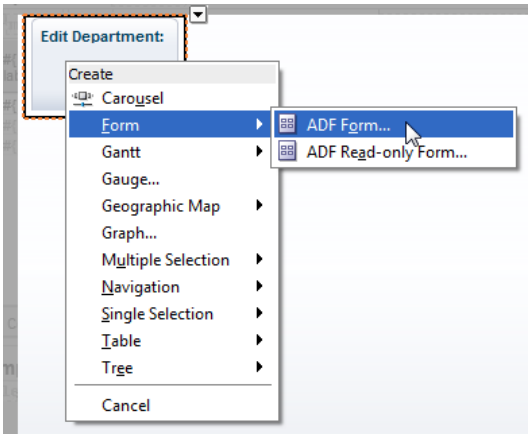
75. Expand the f:view | af:document | af:form hierarchy and select the af:popup entry. Selecting the af:popup component displays the popup dialog in the visual editor



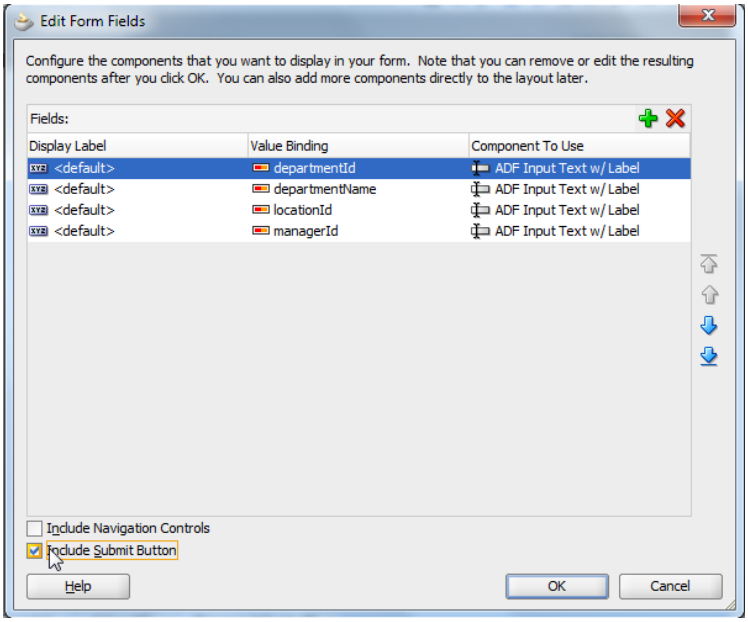
76. In the Data Controls panel, select the "allDepartments" collection entry and drag it into the dialog. Note that the "allDepartments" collection is the same collection that is used to populate the departments table.



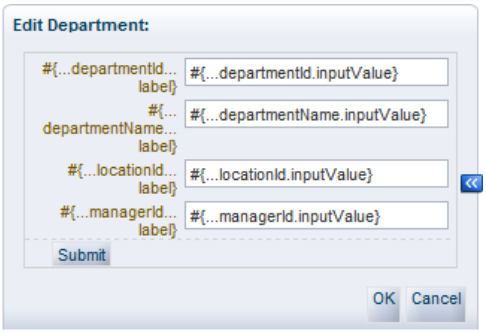
77. In the opened context menu, choose Forms ADF Form



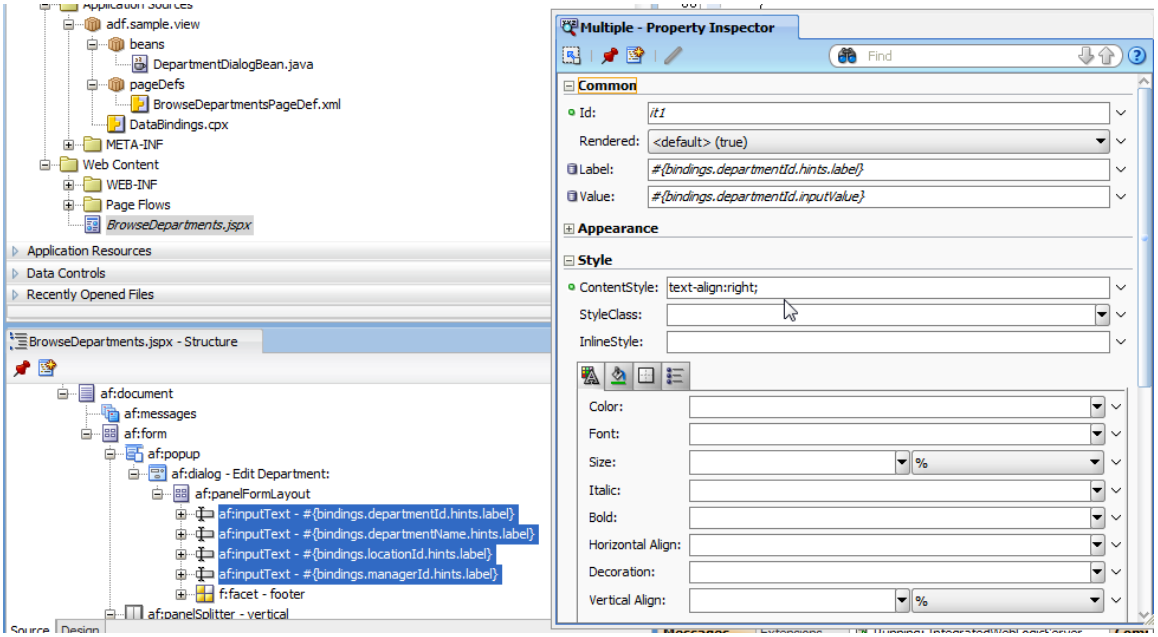
78. Select the "Include Submit Button" checkbox for JDeveloper to add a command button to the form



79. Press OK to close the form creation dialog



80. To right align the form fields, navigate to the Structure Window and select each of the form fields with the ctrl-key pressed



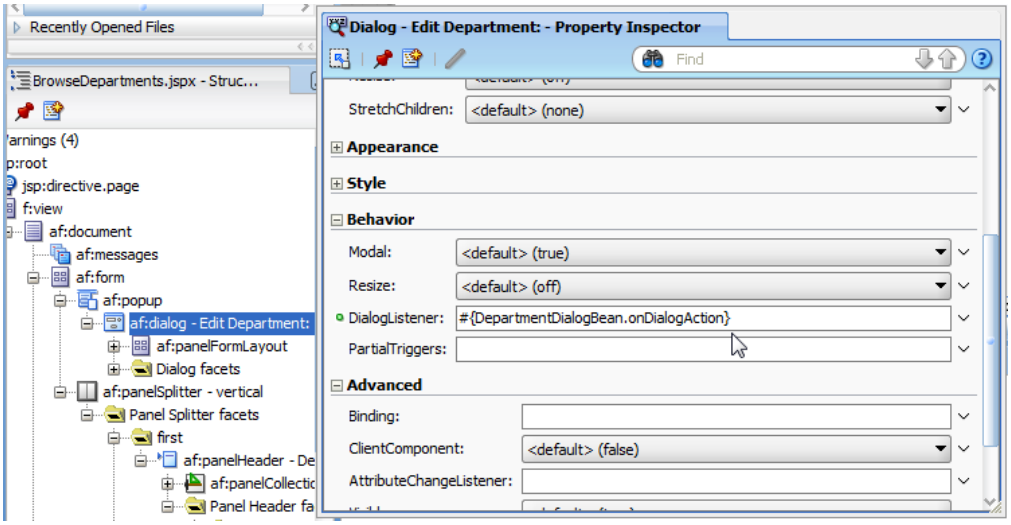
81. Open the Property Inspector (ctrl+shift+I) and type the following CSS string into the "ContentStyle " property

text-align:right;

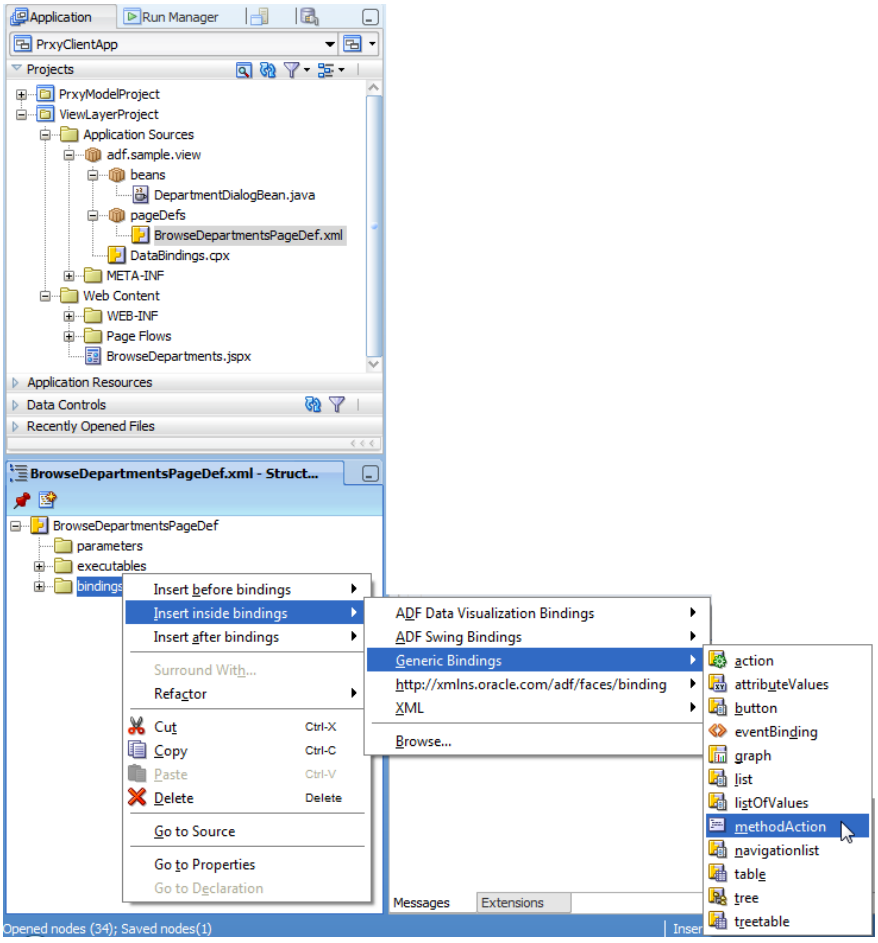
The ContentStyle property ensures that the style sheet is applied to the text field content area so that all data input appears right aligned.

82. The ViewLayerProject has a managed bean configured that has a method defined to handle the dialog close event. The dialog close event provides the information about the button pressed by the user. If "Ok" is pressed, the input form is updated and the change is persisted in the database. If the user pressed "Cancel", the form is closed and the form is not updated at all.

The managed bean is referenced from the "DialogListener" property defined on the af:dialog component, as shown in the image below.



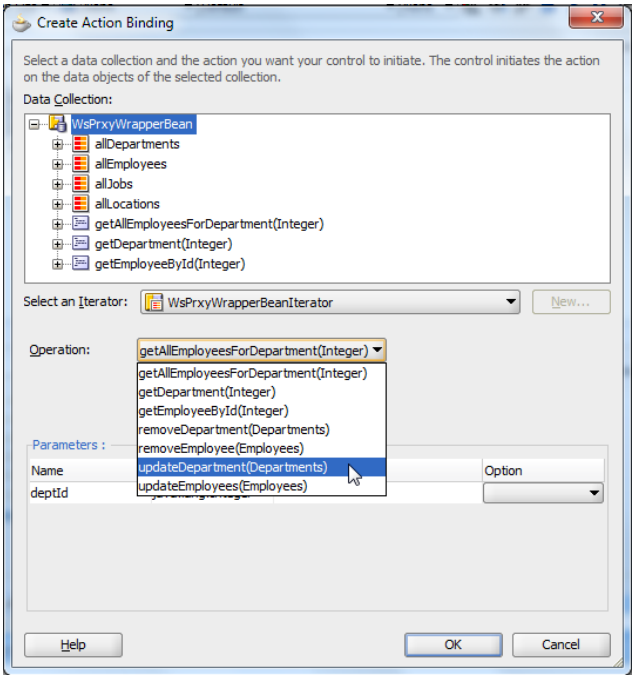
83. Next, you create a method binding that is called by the managed bean method to update the form



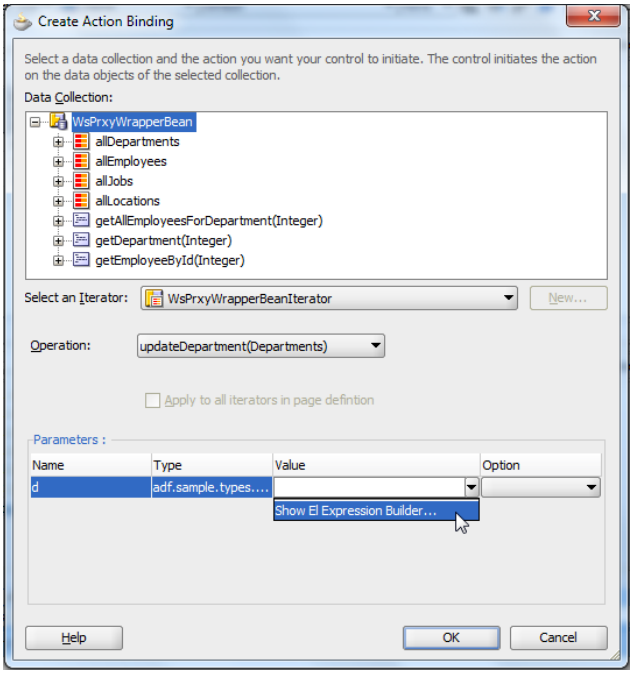
84. Select the BrowseDepartmentsPageDef.xml file entry in the adf.sample.view.pageDefs package and open the Structure window

85. Right mouse click onto the "bindings" node and choose Insert inside bindings | Generic Bindings | methodAction from the context menu

86. In the opened dialog, select the "WsPrxyWrapperBean" node and choose the "updateDepartment(Department)" method from the list of operations.

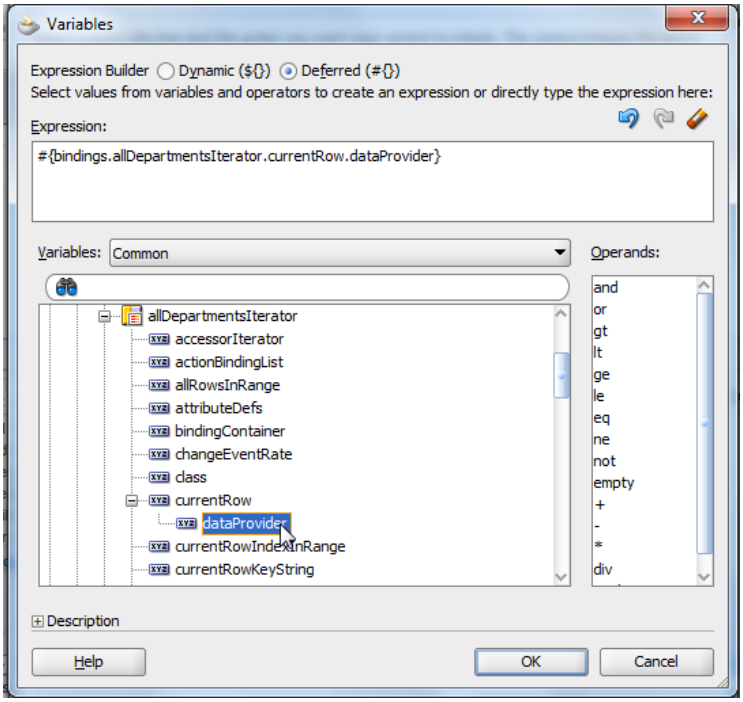


87. The method requires a Department object to be added as an argument. Click into the Parameter value field of the deptId parameter.



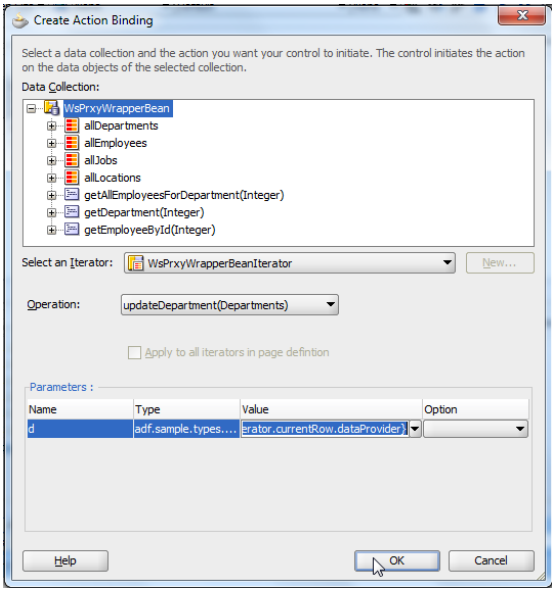
88. Open the select menu by clicking the list icon to the right

89. Click onto the "Show EL Expression Builder" option



90. Select the allDepartmentsIterator | currentRow | dataProvider entry in the binding dialog. The dataProvider of the current selected table row is an object of type Department. Because the input form shares the same collection with the table, it also shares the ADF iterator binding; making sure the selected row is displayed in the form for update.

91. Press OK to close the EL builder dialog



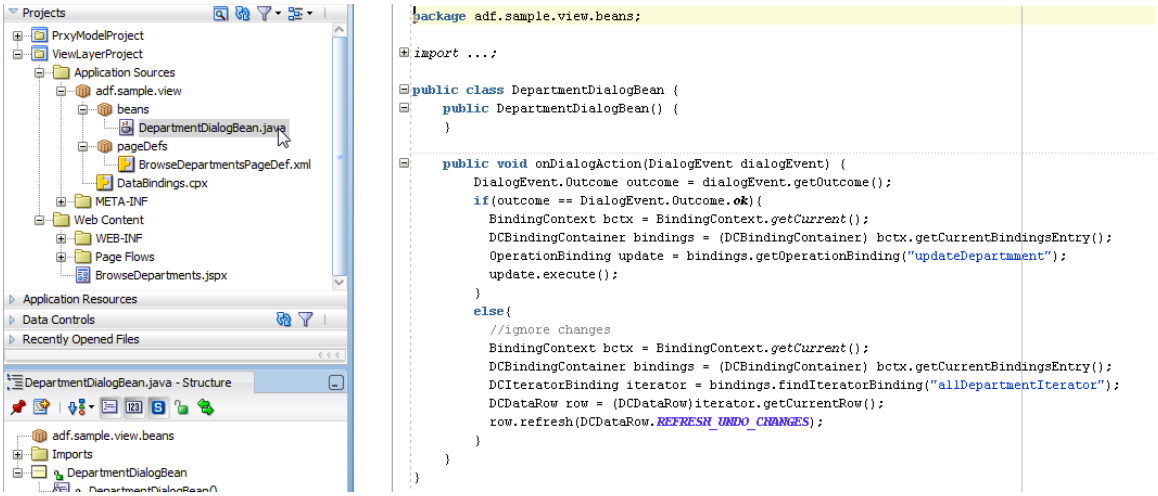
92. Press Ok to close the Action Binding dialog. The Parameters Value property should have the following value set

#{bindings.allDepartmentsIterator.currentRow.dataProvider}

93. Earlier in this hands-on, you created a table component binding to the DepartmentDialogBean managed bean. Select the DepartmentDialogBean.java class and double click it to open the code editor

94. Add the following code line after the **update.execute()** method

```
AdfFacesContext.getCurrentInstance().addPartialTarget(table1);
```

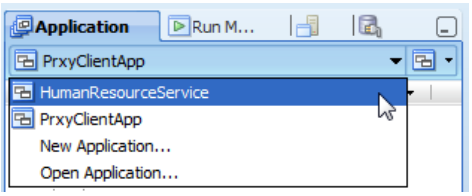


The updated managed bean method should look as shown below

```
19    public void onDialogAction(DialogEvent dialogEvent) {
20        DialogEvent.Outcome outcome = dialogEvent.getOutcome();
21        if(outcome == DialogEvent.Outcome.ok){
22          BindingContext bctx = BindingContext.getCurrent();
23          DCBindingContainer bindings = (DCBindingContainer) bctx.getCurrentBindingsEntry();
24          OperationBinding update = bindings.getOperationBinding("updateDepartment");
25          update.execute();
26          AdfFacesContext.getCurrentInstance().addPartialTarget(table1);
27        }
28        else{
29          //ignore changes
30          BindingContext bctx = BindingContext.getCurrent();
31          DCBindingContainer bindings = (DCBindingContainer) bctx.getCurrentBindingsEntry();
32          DCIteratorBinding iterator = bindings.findIteratorBinding("allDepartmentIterator");
33          DCDataRow row = (DCDataRow)iterator.getCurrentRow();
34          row.refresh(DCDataRow.REFRESH_UNDO_CHANGES);
35        }
36    }
```

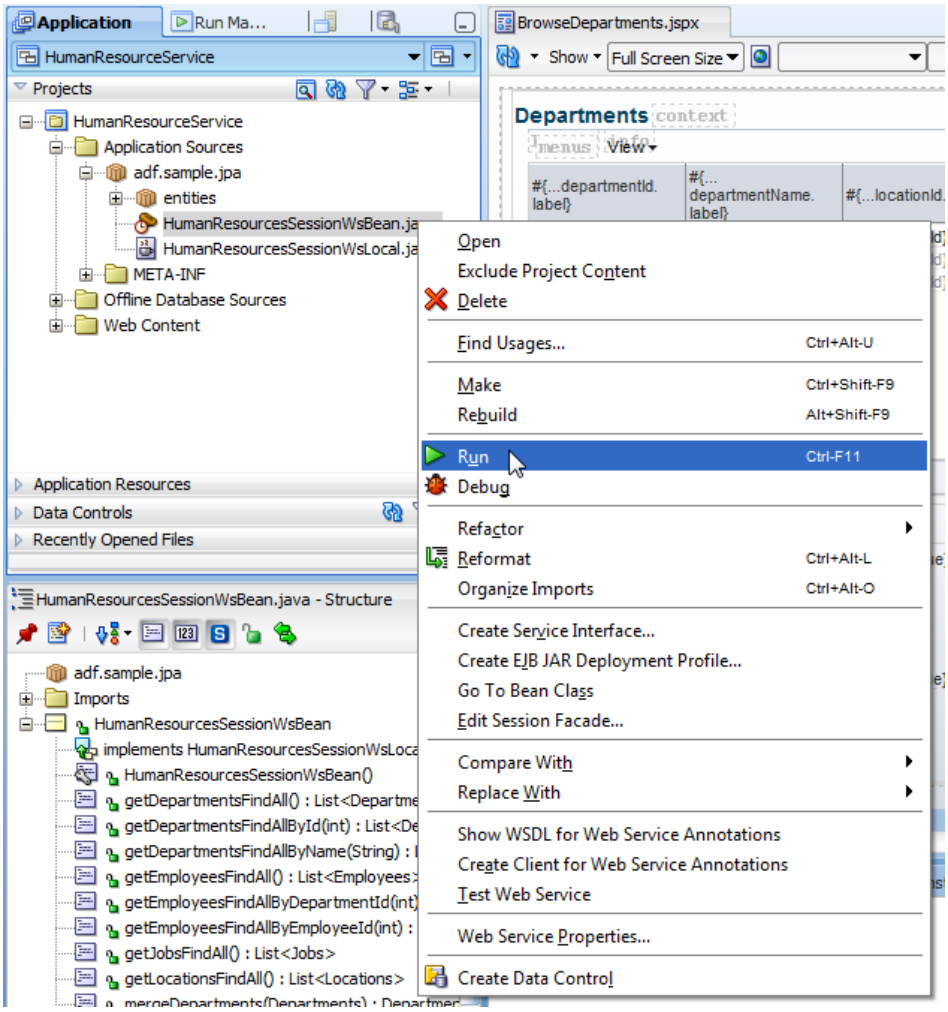
The code in the managed bean calls the "updateDepartment" method binding you just created. When calling the method, the current row data provider object is passed to the Web Service for update. The Web Service passes it to the EJB session façade method to persist the change in the database. The last code line you added ensures the table is refreshed to show the change.

95. In the Application Navigator select list, choose the "HumanResourceService" entry.

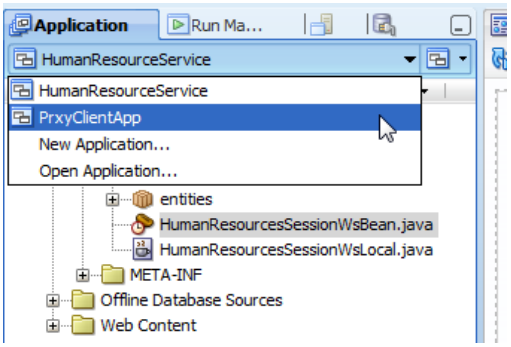


96. Expand the Application Source | adf.sample.jpa.entities package and select the "HumanResourcesSessionWsBean.java" entry

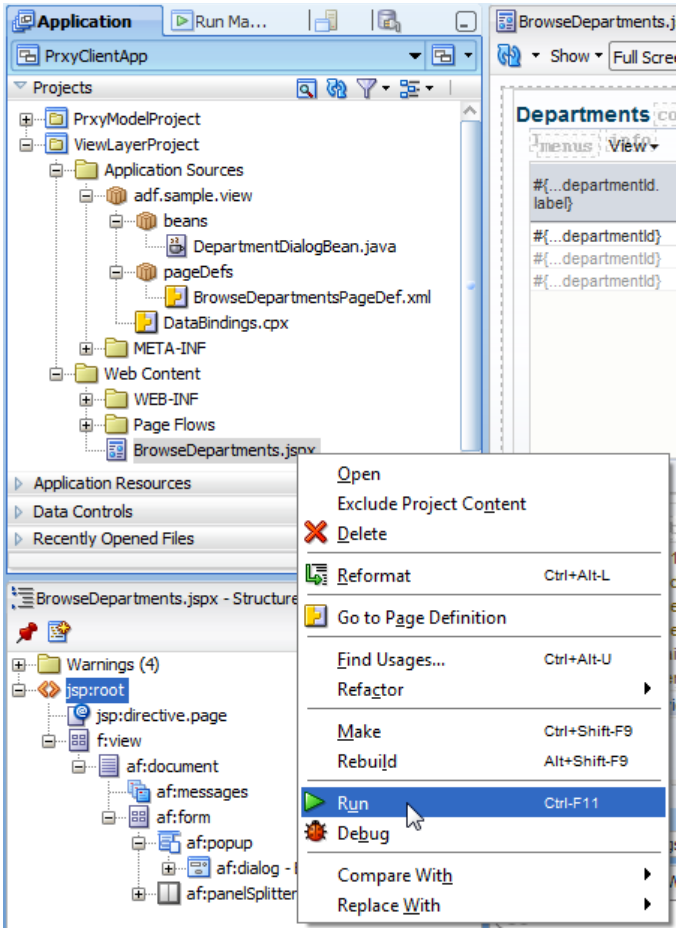
97. Choose "Run" from the right mouse context menu to ensure the Web Service is deployed and running on the integrated WLS server



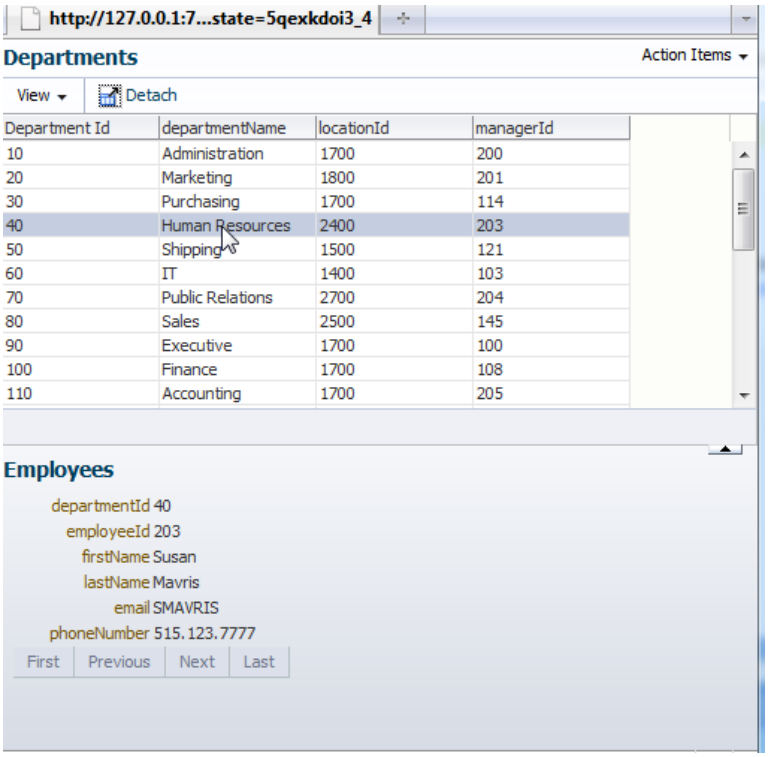
98. Switch the application selector back to the "PrxyClientApp" application



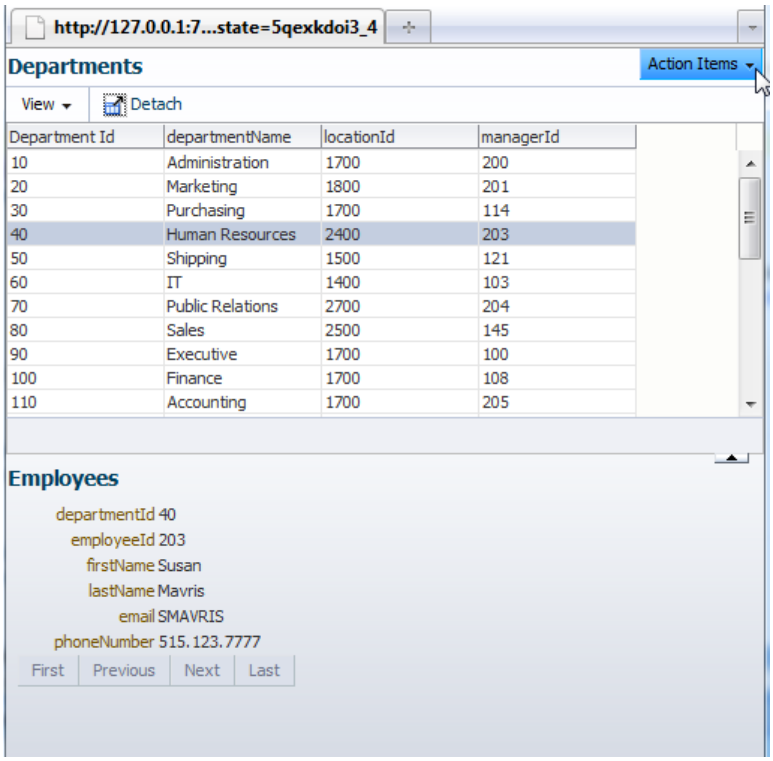
99. Select the BrowseDepartments.jspx entry and choose "Run" from the right mouse context menu



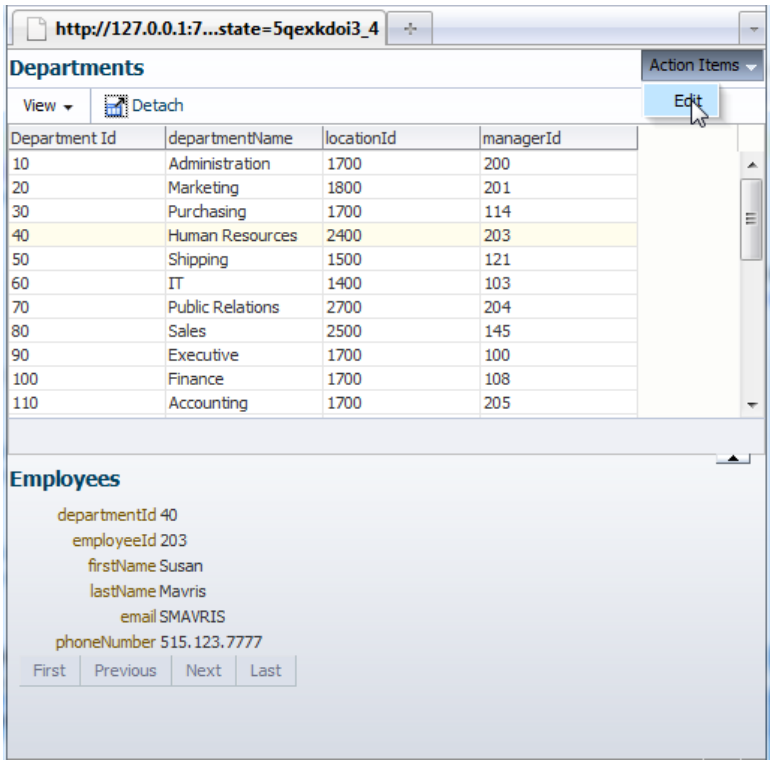
100. In the running application, select a department row and see the employee detail getting updated.



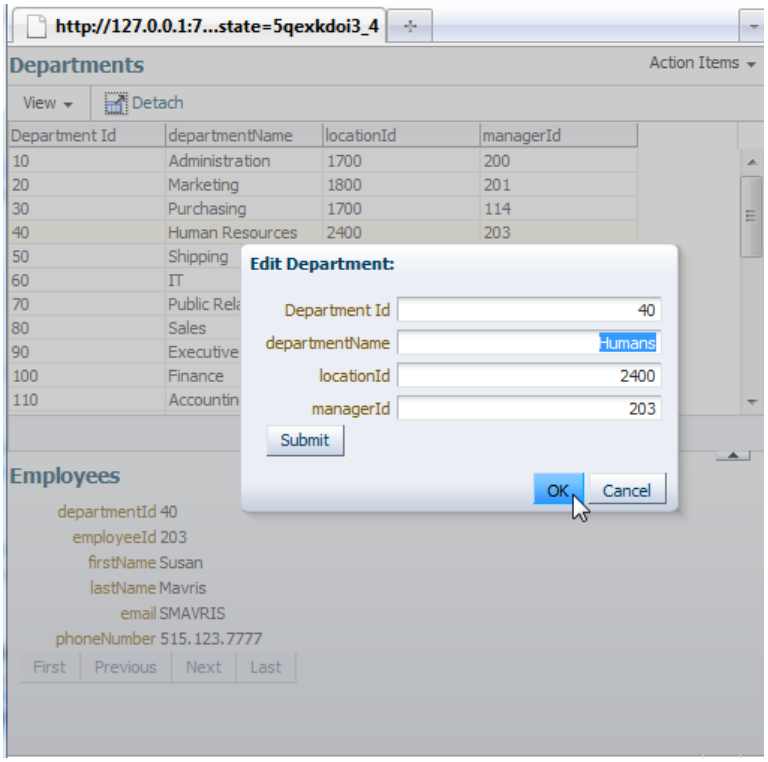
101. With a department selected, press the "Action Item" menu to expand it



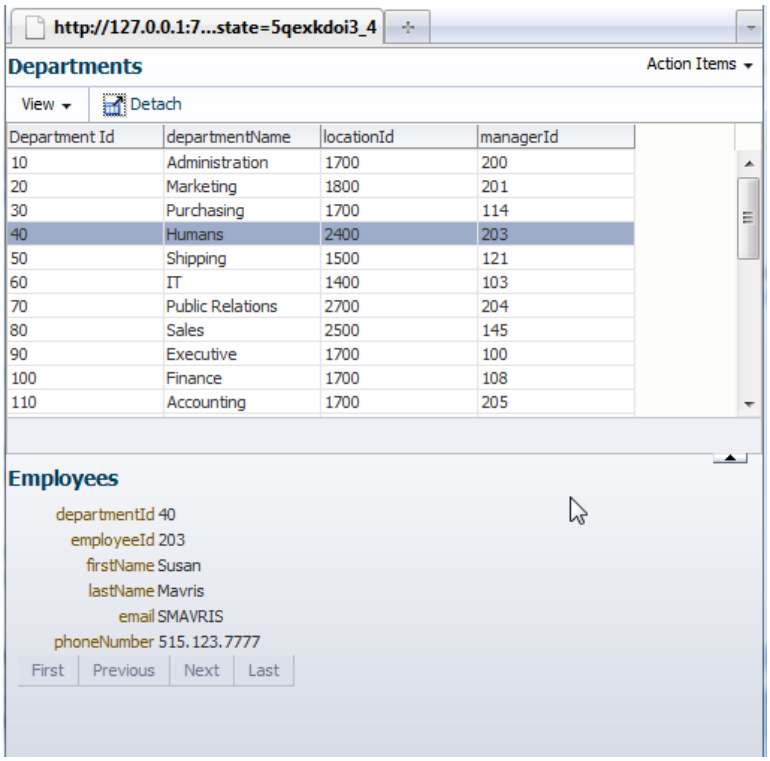
102. Click "Edit" to bring up the edit form



103. Change a value in the edit form and press Ok. This closes the dialog and invokes the managed bean method handling the dialog close event.



104. The table is refreshed through PPR to show the change



Download

The Oracle JDeveloper workspaces with the EJB Web Service, the starter application and the completed application can be downloaded as sample #72 from the ADF Code Corner website:

http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html

RELATED DOCOMENTATION

☒	
☒	
☒	