ADF Code Corner

73. Hands on – Creating a search form using a POJO WS and the Web Service Data Control

twitter.com/adfcodecorner

Abstract:

This hands-on guides you through creating a Web Service based search form that uses a complex input argument for the search criteria. The Web Service is POJO based and accessed from the Web Service Data Control

Duration: 60 Minutes

The starter work space can be downloaded from ADF Code Corner as well

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

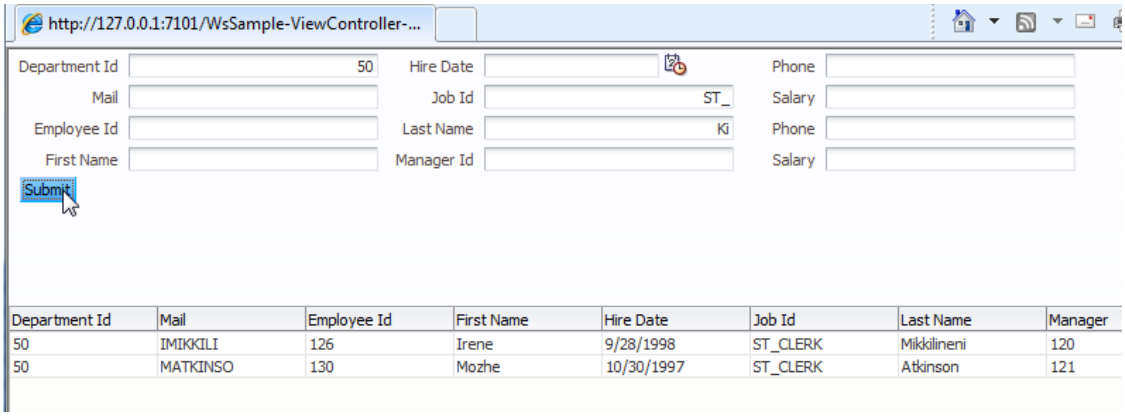
Introduction

The Web Service Data Control is one of two options ADF developers have to access JAX-WS Web Services. It exposes service collections, methods and attributes in the Oracle JDeveloper Data Controls panel for declarative use. In this end-to-end hands-on exercise you will

- create a Web Service from a JavaBean class
- configure the ADF WS Data Control
- define UI hints and validation.

A screenshot of the application runtime is shown below.

In the sample application, you use an ADF form to query data from a Web Service, which exposes a search function accepting a complex argument of type Employee. The result set is displayed in the table without the need to refresh the whole page.

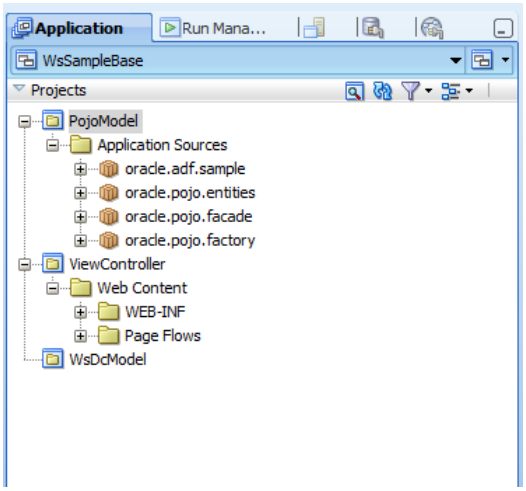


Prerequisite and Setup

The POJO model uses hard coded data values that match the data of the Oracle HR demo schema. There is no database required to run this hands-on example. You need Oracle JDeveloper 11g 11.1.1.3 or later installed to follow this hands-on.

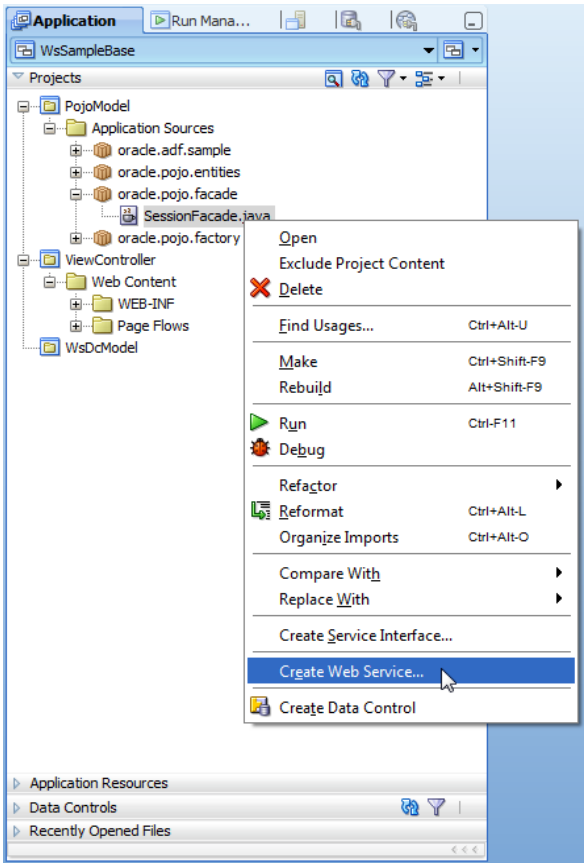
Before you start, download and extract the "WsSampleBase.zip" starter workspace. Open Oracle JDeveloper and navigate to the directory where you extracted the ZIP file and open the WsSampleBase.jws workspace file.

The initial Oracle JDeveloper IDE view is shown below



Note: In this hands-on, the Web Service project and the ADF model and view controller project are within a single JDeveloper workspace. In reality, the Web Service would be deployed on a remote server. We simplified the setup for this hands-on so you can use the integrated WebLogic Server for the Web Service runtime too.

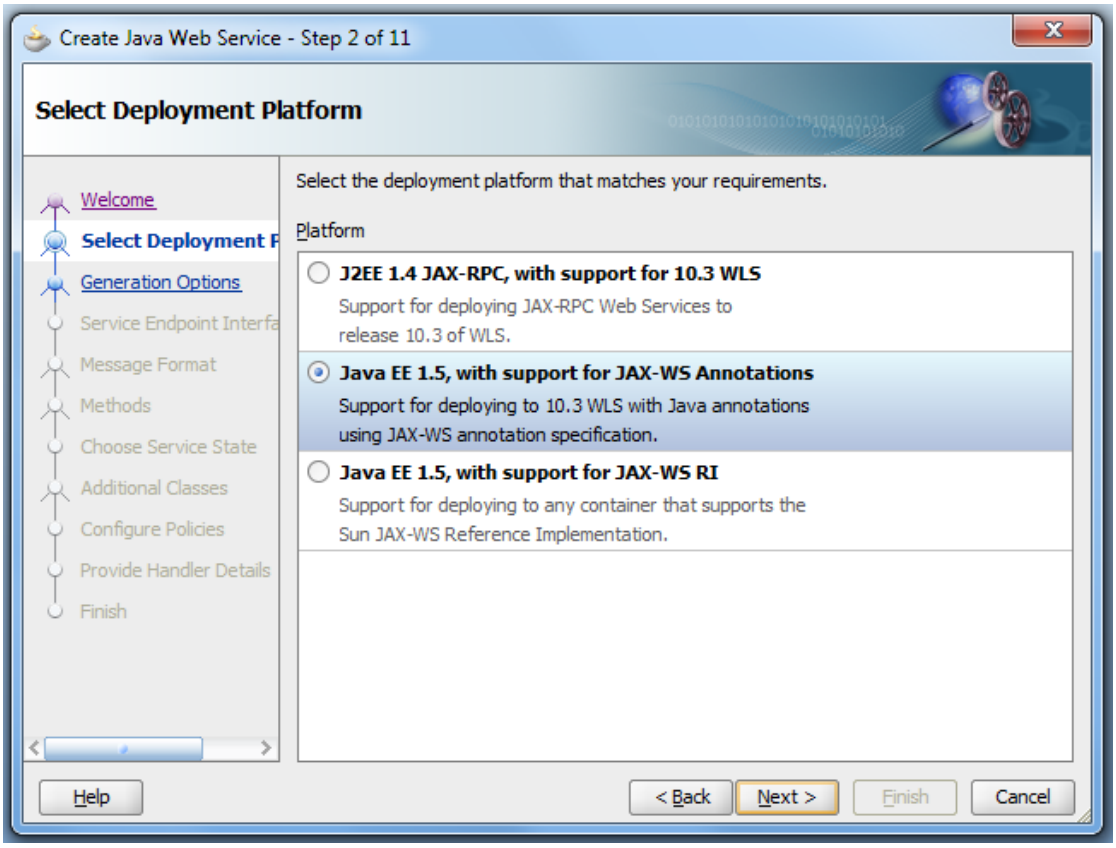
Creating the POJO Web Service



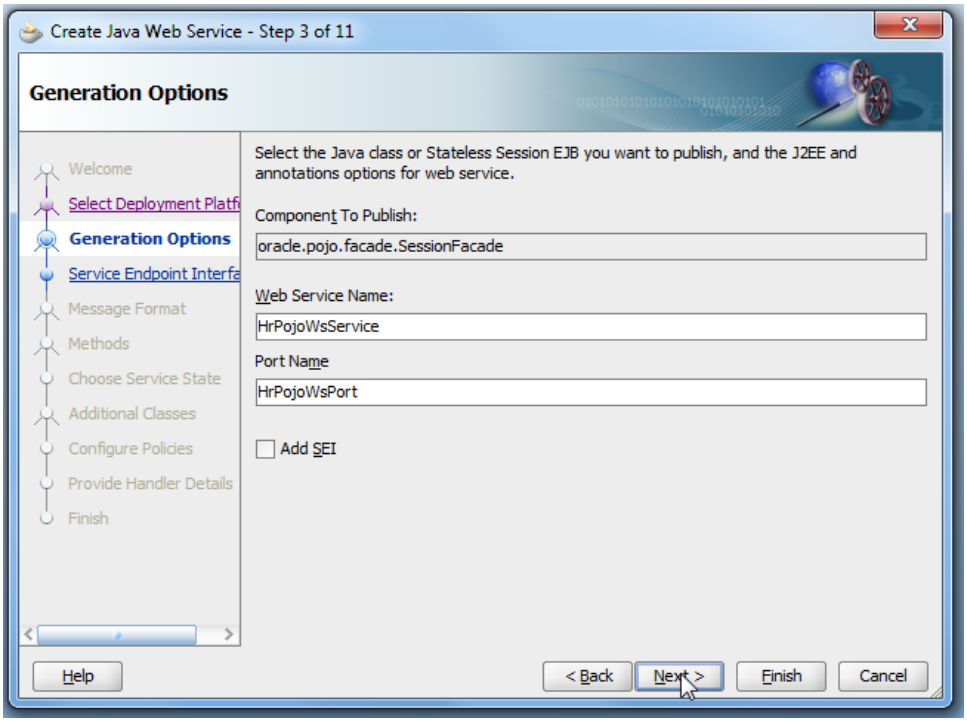
1. To create the Web Service, expand the POJO Model project's `oracle.pojo.facade` package and select the SessionFacade.java class.

2. From the right mouse context menu, choose Create Web Service

3. In the Web Service creation dialog, choose the "Java EE 1.5, with support for JAX-WS Annotations" option to build an annotation based Web Service from the POJO model.

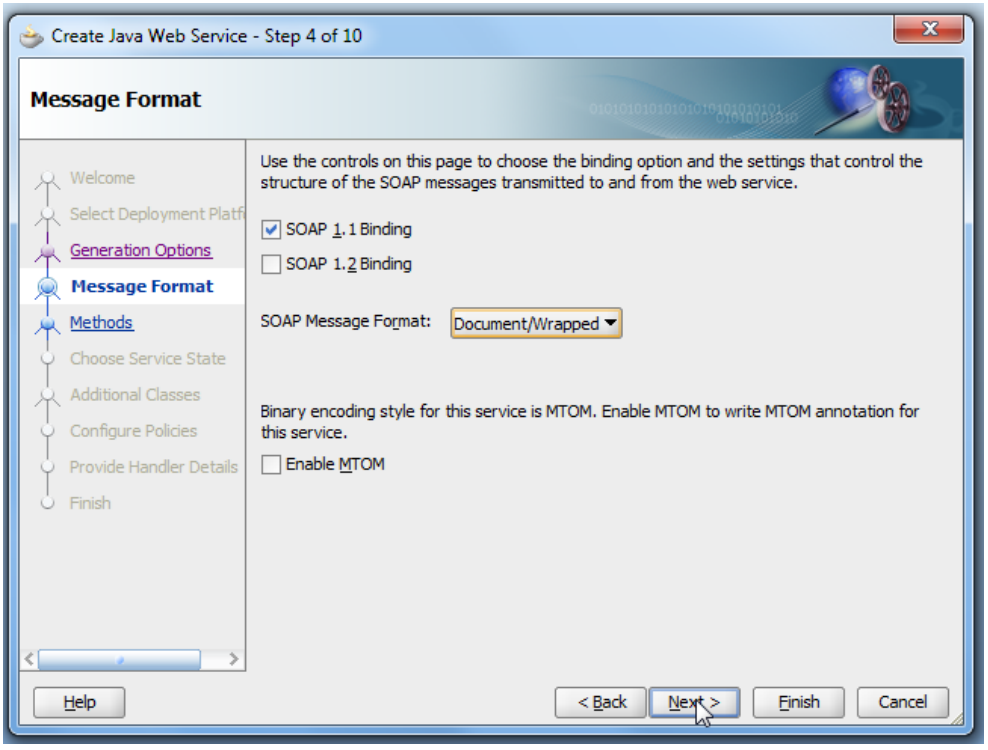


4. In the "Web Service Name" field, type "HrPojoWsService". The WS name automatically becomes the name for the WS port too. The WS port is later used when accessing the Web Services from ADF.

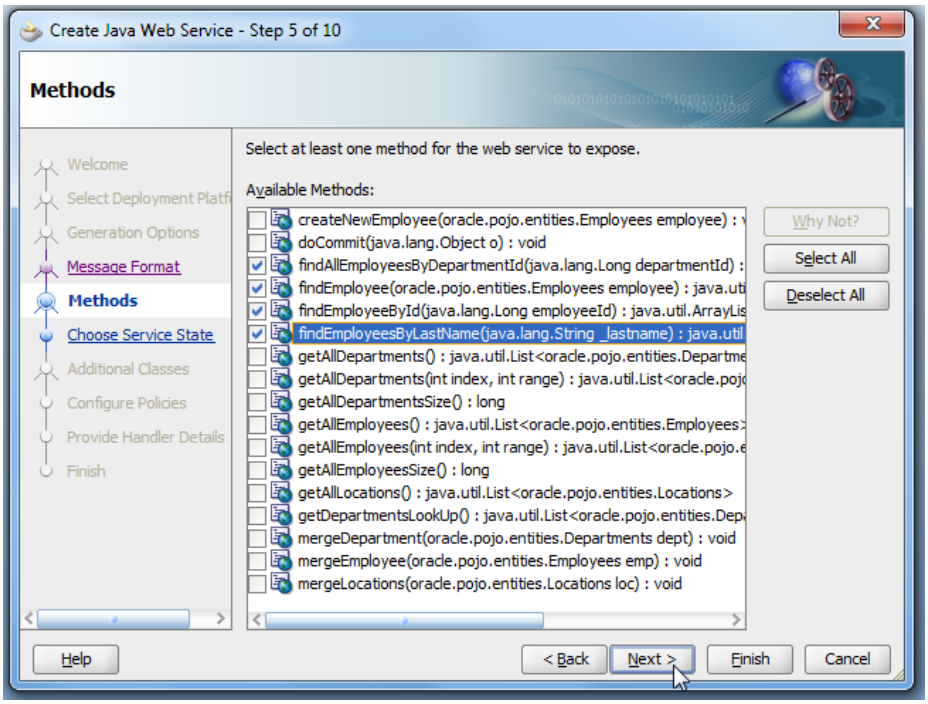


5. Click Next

6. Again, click "Next" and accept all the default settings in Step 4 of the WS Creation wizard

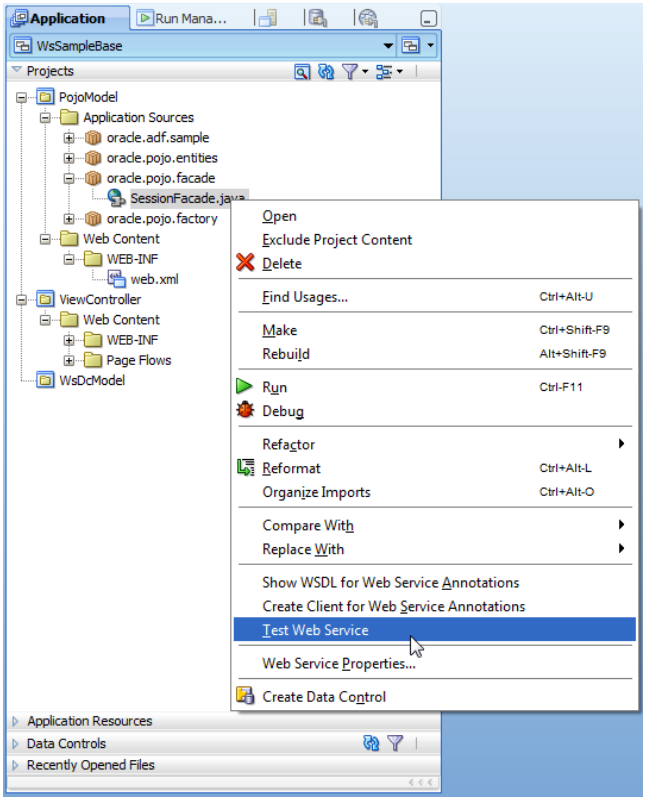


7. The POJO class exposes more methods than needed for this hands-on. Press "Deselect All" and then select all methods that have a "findEmployee" in the name.



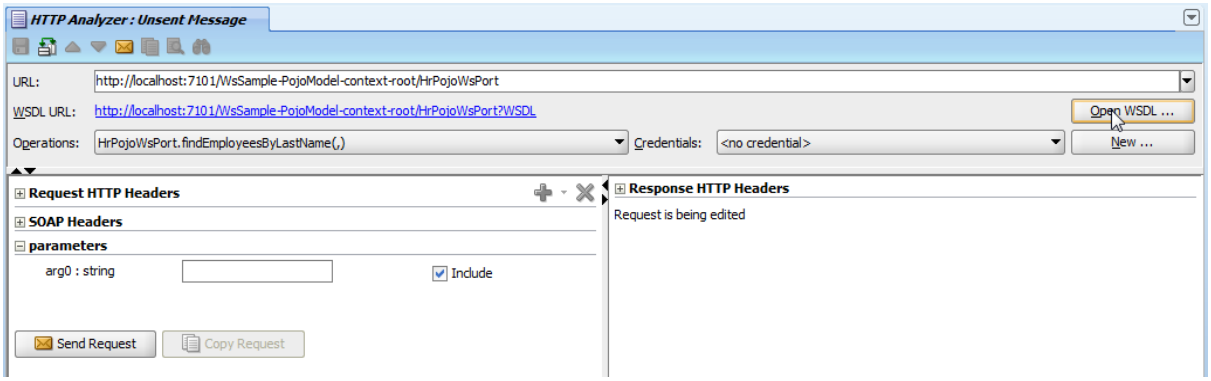
8. Click "Finish" to generate the Web Service definition

9. In the Oracle JDeveloper Application Navigator, select the "SessionFacade.java" entry, which has its icon changed to indicate the Web Service functionality, and choose "Test Web Service" from the right mouse context menu.



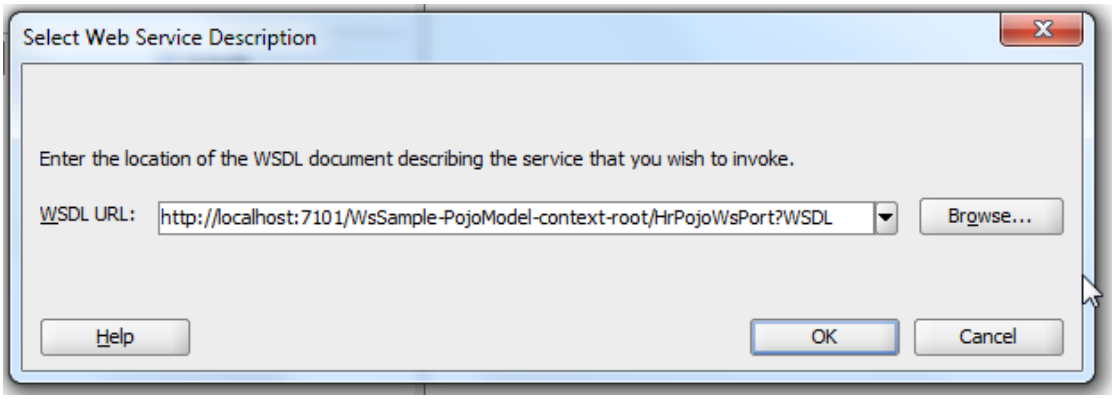
Note: If this is your first time experience with building JAX-WS Web Services, you may double click onto the Java class entry to see the added annotations.

10. The WS tester shows the Web Service WSDL file reference, a list box exposing Web Service functions, a form to input request parameters and an area to display the Web Service response



11. Click the "Open WSDL" button to open the "Select Web Service Decsription" dialog.

12. Mark the WSDL string reference and copy it to the clipboard using the ctrl+c keyboard key. The WSDL reference is needed when creating the WS Data Control. For this hands-on, the Web Service is deployed to the integrated WebLogic Server in Oracle JDeveloper.

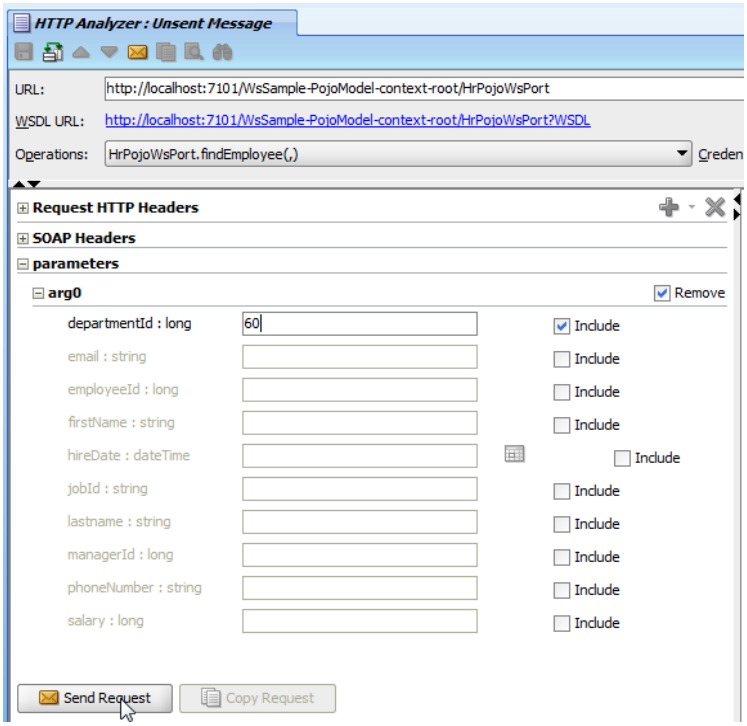


13. Ok the dialog

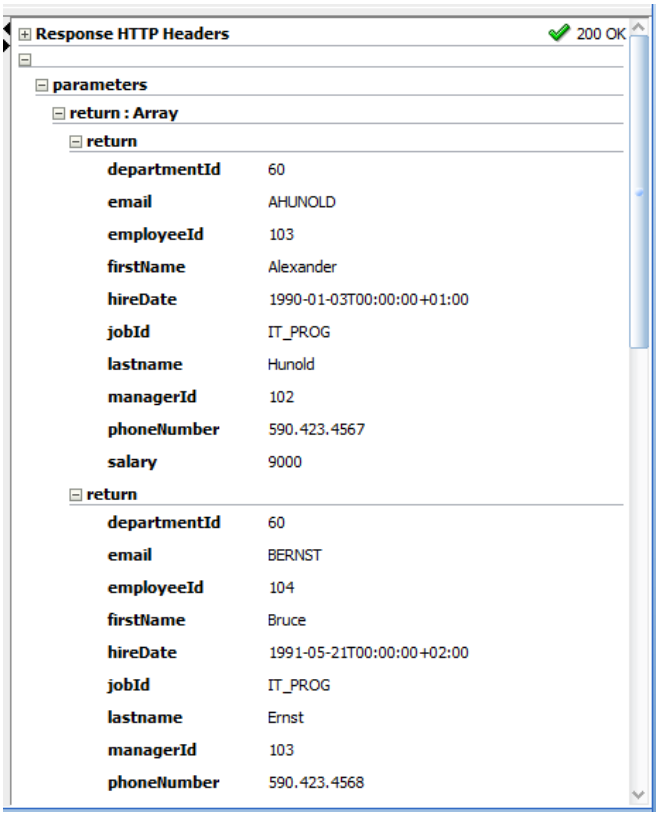
14. From the "Operations" list, select "HrPojoWsPort.findEmploye(,)", which is a function that expects an Employee object type as an input argument

15. In the parameters form, select the "departmentId" attribute to be included in the request and unselect all the other options. Define a value of "60" for the departmentId attribute (no quotes)

16. Press the "Send Request" button



17. The WS response is displayed in the response area

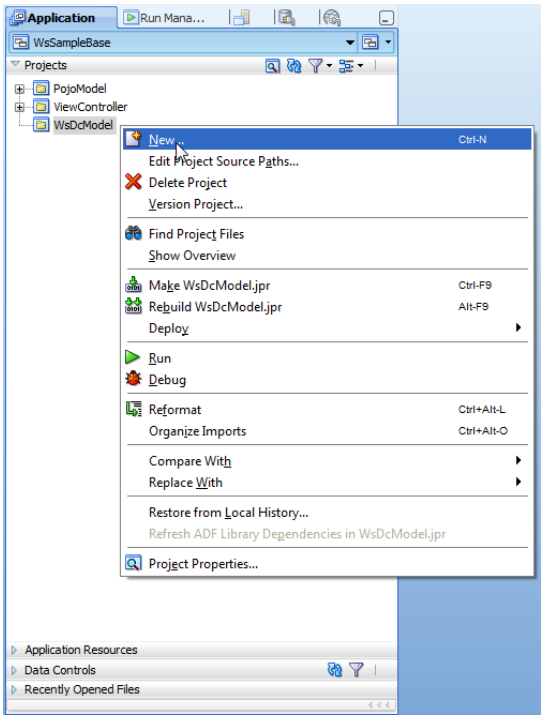


18. Close the WS tester.

Creating the WebService Data Control

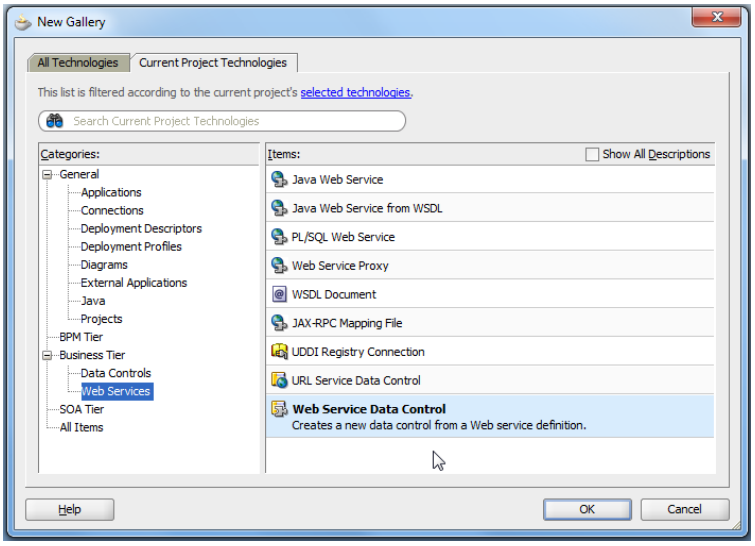
To create the Web Service Data Control definition, a WSDL reference is needed for JDeveloper to write the service structure into XML metadata.

1. Select the "WsDcModel" project and choose "New" from the right mouse context menu to create a new WS Data control configuration



2. In the New Gallery, select the Business Tier | Web Service entry

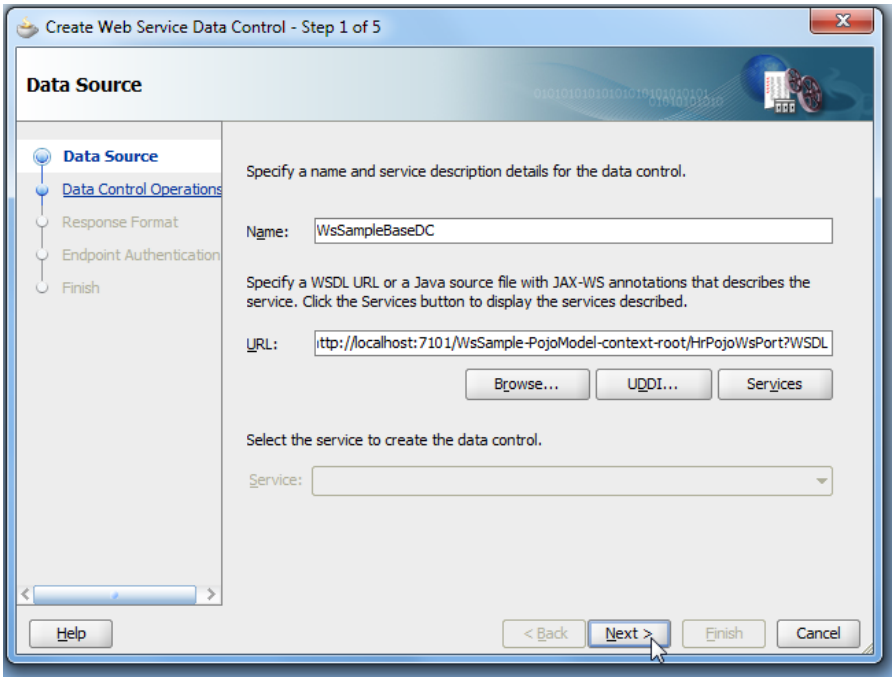
3. Select the "Web Service Data Control" entry in the "Items" area and "Ok" the dialog



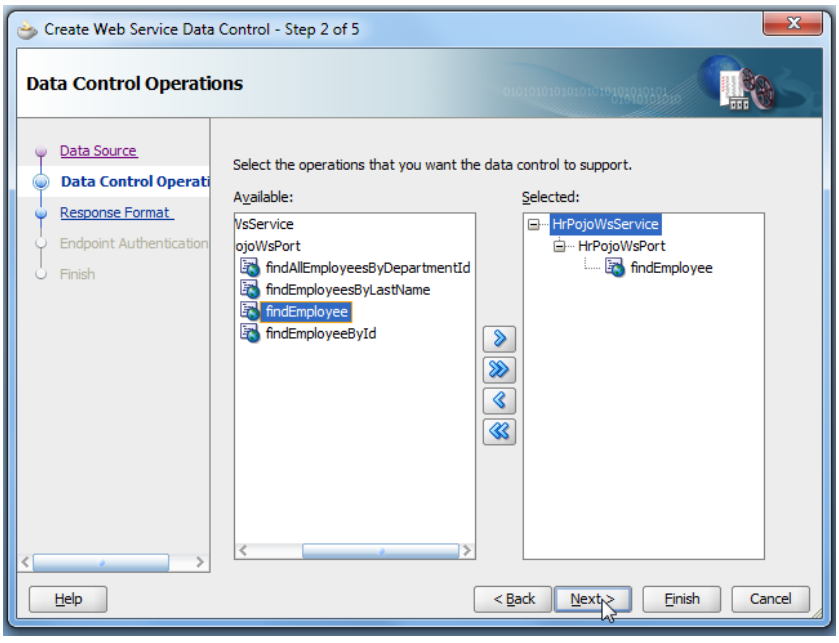
4. Type "WsSampleBaseDC" in the "Name" field to define the display name of the Data Control

5. In the URL field, paste the previously WSDL reference you copied when testing the Web Service.

6. Press "Next" to continue the configuration and to validate the WSDL access

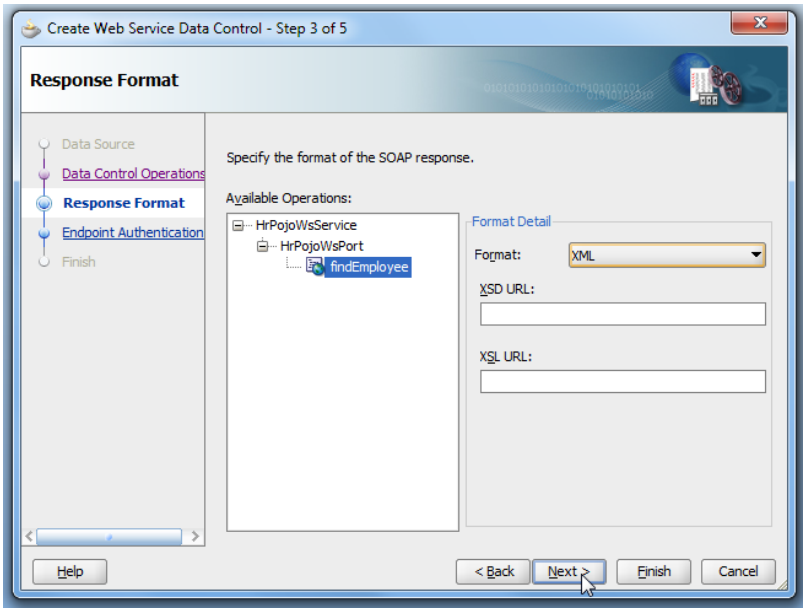


7. From the 4 employees search functions exposed by the WS, only the "findEmployee" should be exposed by the Data Control.

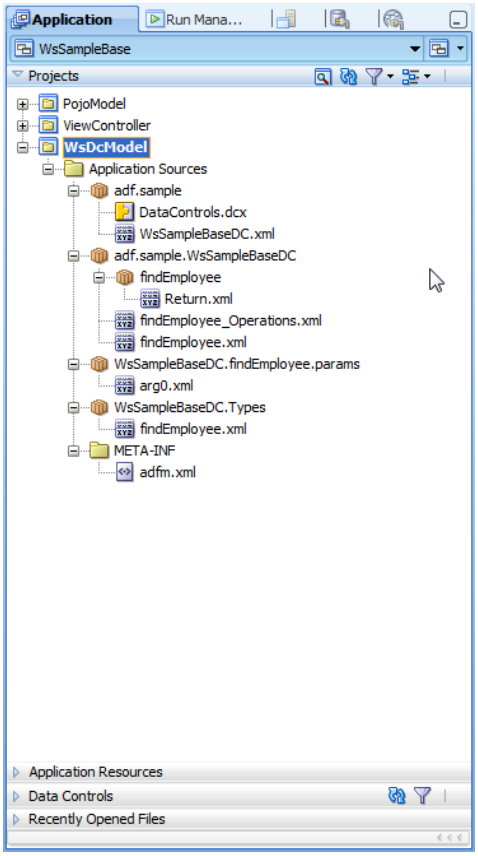


8. Press Next

9. In the Response Format dialog, accept the default format, which is XML. Press "Next"



10. The last step in the configuration (not shown as a screen shot) allows you to define authentication for the Web Service access. Ignore the dialog and press "Finish" to generate the WS DC configuration metadata



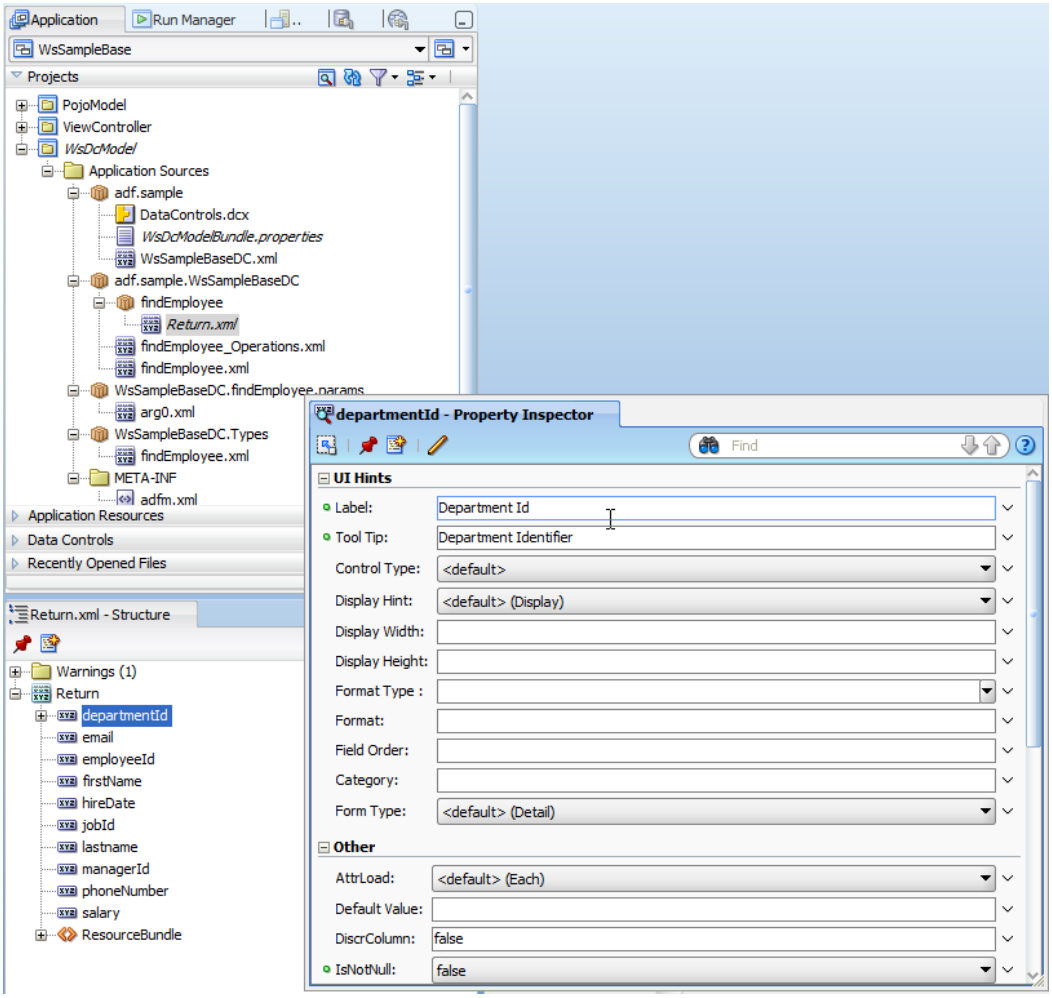
11. Expand the WSDcModel project structure and select the "Return.xml" configuration under the "findEmployee" function. The "Return.xml" document describes the result set of the WS response.

12. Open the Structure Window (ctrl+shift+S) and the Property Inspector (ctrl+shift+I)

In the Structure Window, select one of the exposed attributes to edit its "Label" and "Tool Tip" attribute. The WS Data Control allows developers to define UI Hints that are looked up by the view layer at runtime. This not only allows you to define consistent label and control hints, it also allows you to define then so they can be translated.

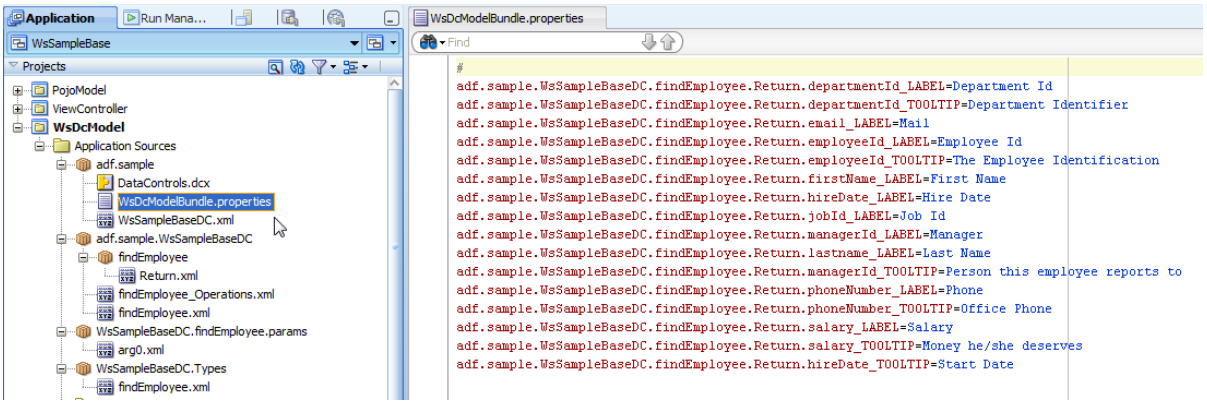
13. Provide "Label" and "Tool Tip" entries for the attributes as shown in the image below. You can detach the Property Inspector window and move it closer to the Structure Window using the mouse and folding the ctrl-key pressed.

Note: The property settings are saved when you step out of the property field. So make sure you briefly select another property before moving on to edit another attribute.



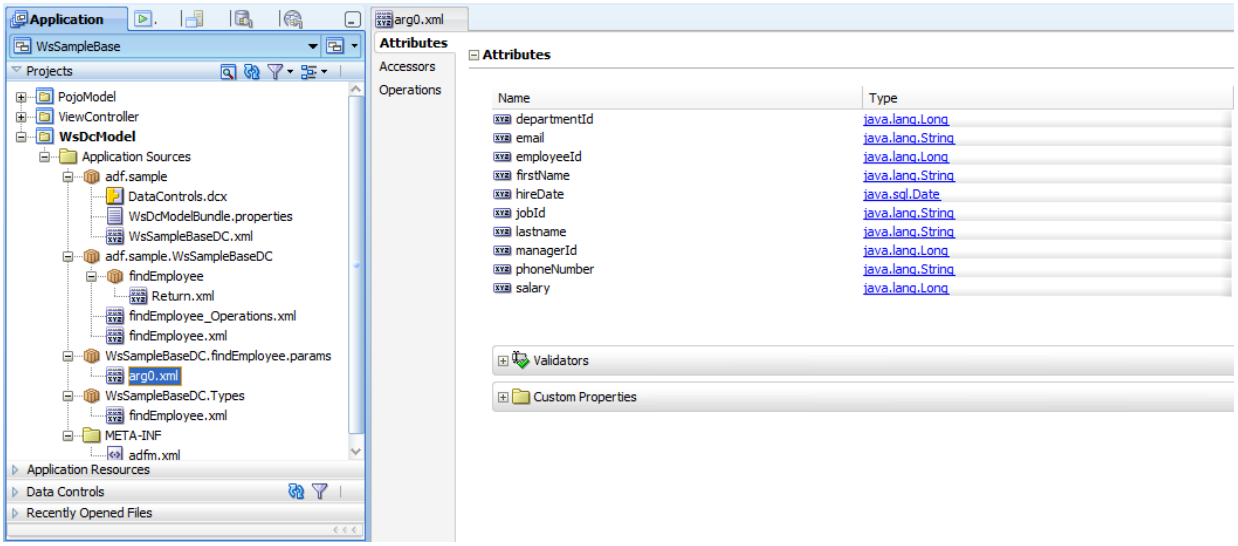
14. Select the WSDcModelBundle.properties file and double click it to open. The file defines keys and strings you provided for the attributes.

Though not needed for this hands-on, to create a translated version, you duplicate the file and define the name of the duplicate as WSDcModelBundle_<lang>.properties, where <lang> should be replaced with "de" for German, "fr" for French etc.



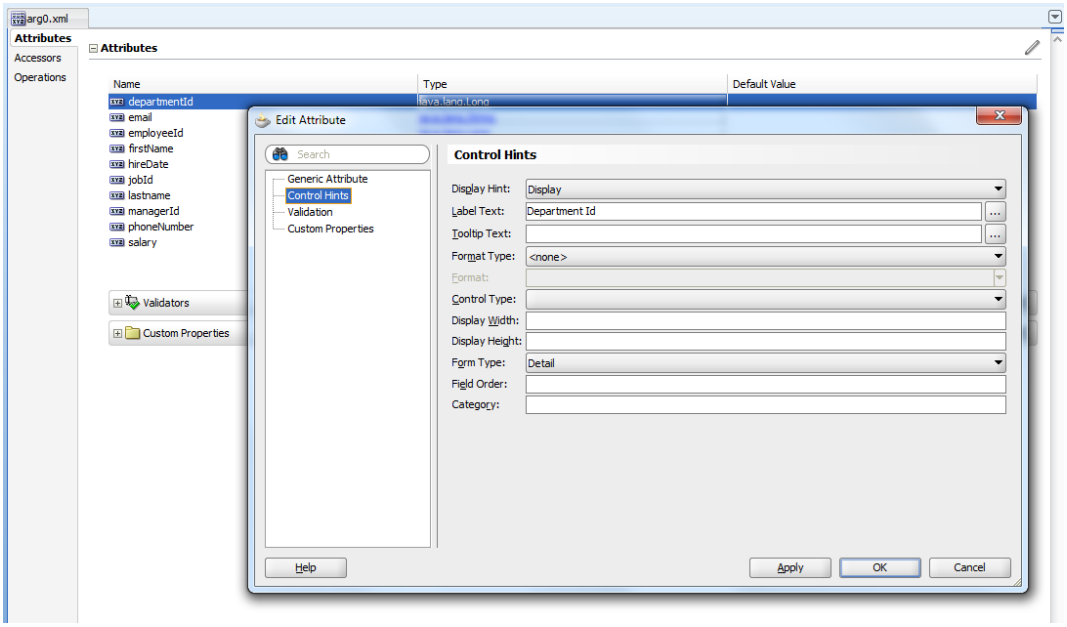
15. To do the same for the input arguments expected by the findEmployee, function, select the "arg0.xml"entry in the "WsSampleBase.findEmployee.params" package

16. Double click the arg0.xml entry to open the edit dialog. Note that this is a second option to define attribute of the WS DC



17. Select an attribute, e.g. "departmentId", and press the "pencil icon" to bring up the edit dialog

Note: the labels you define for the "attributes will later show in the search form. The previously edited labels are those that will show in the result table



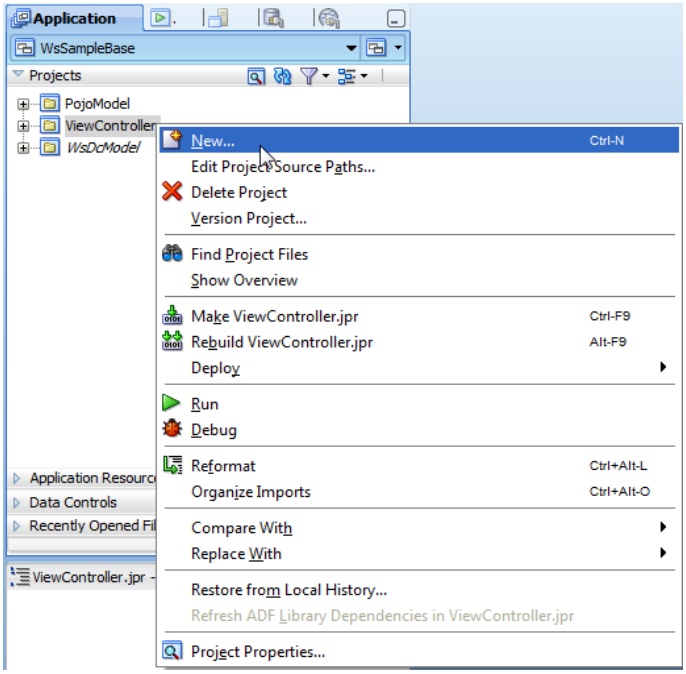
18. Edit the "Label Text" and "Tool Tip" text.

Note that there is a "Validation" section in this dialog that allows you to define validation rules for attributes that are then enforced by the ADF binding layer. We have an example of how to use it at the end of this hands-on.

19. Save the JDeveloper workspace when you are done.

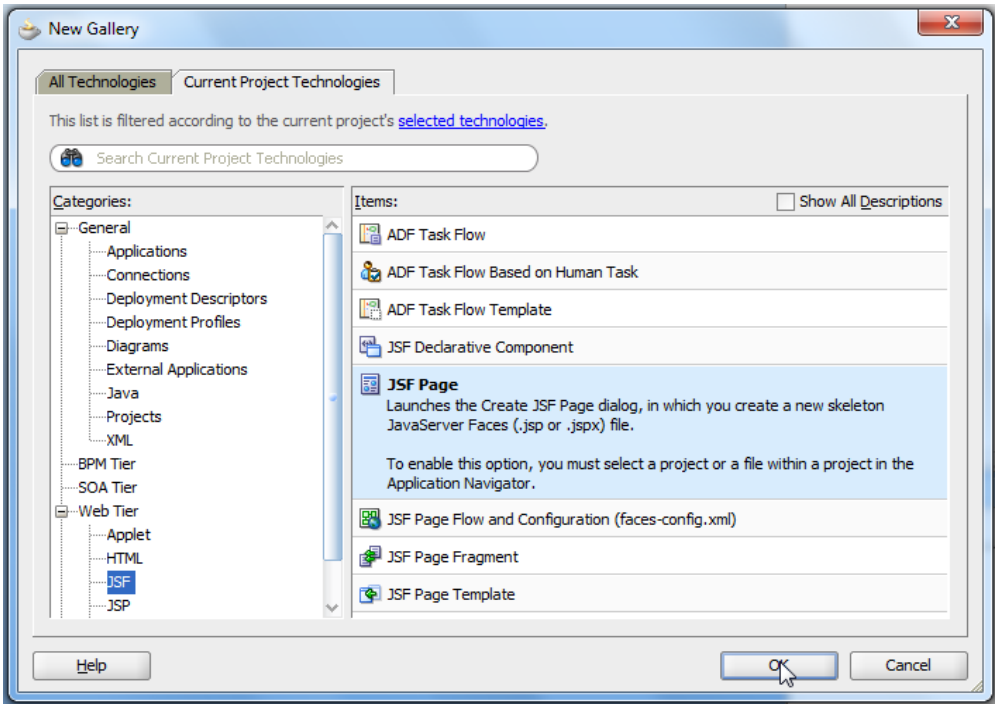
Creating the Search Form

1. Select the "ViewController" project and choose "New" from the right mouse context menu.



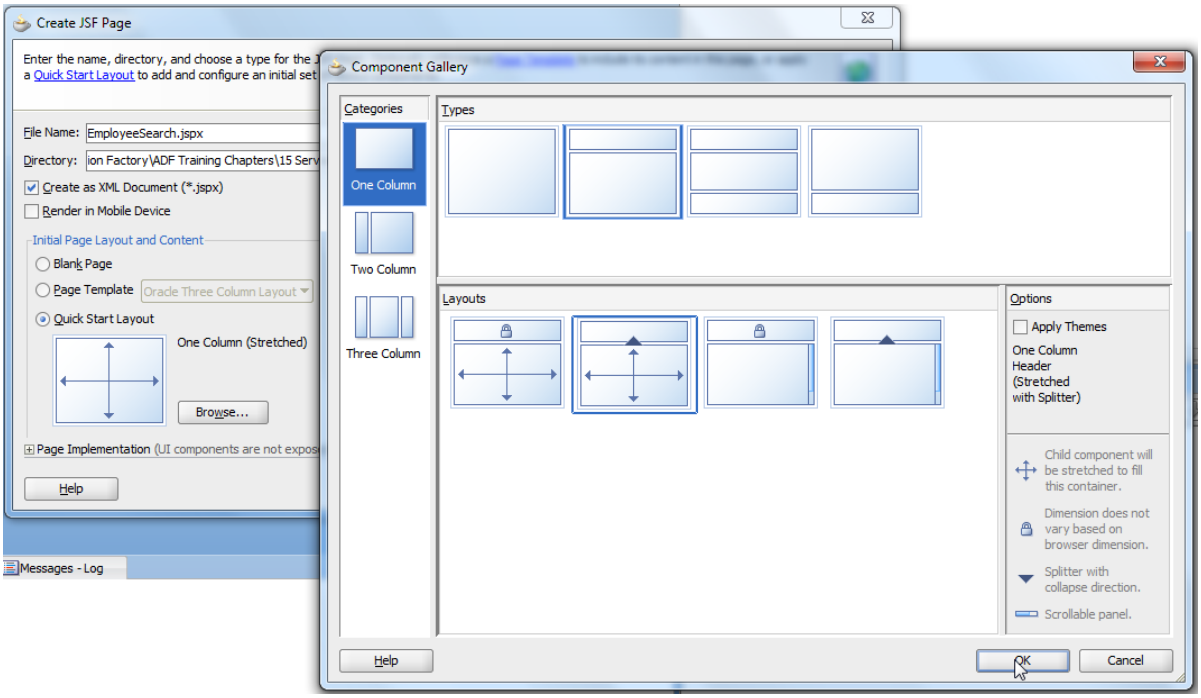
2. In the New Gallery, select the Web Tier | JSF category and choose the "JSF Page" option.

3. Press "Ok"



4. In the Create JSF Page dialog define the "File Name" as "EmployeeSearch.jspx"

5. Select the "Quick Start Layout" Radio Button and press the "Browse" button



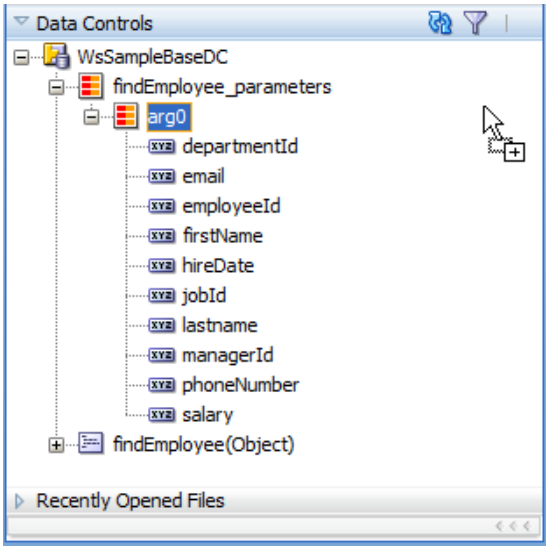
6. Select a one column layout as shown in the image above. This will create a page with a panel splitter component to separate out the two view areas

7. Press "Ok" on both dialogs to render the empty page

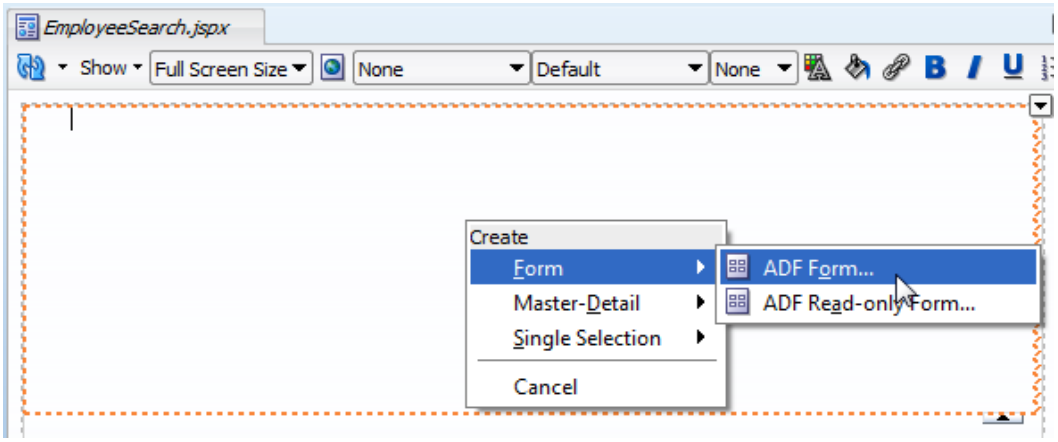
8. Expand the "Data Controls" panel

9. Select the "arg0" entry under the findEmployee_parameters node. This node resolves the attribute structure of the complex input type – and instance of the Employee object.

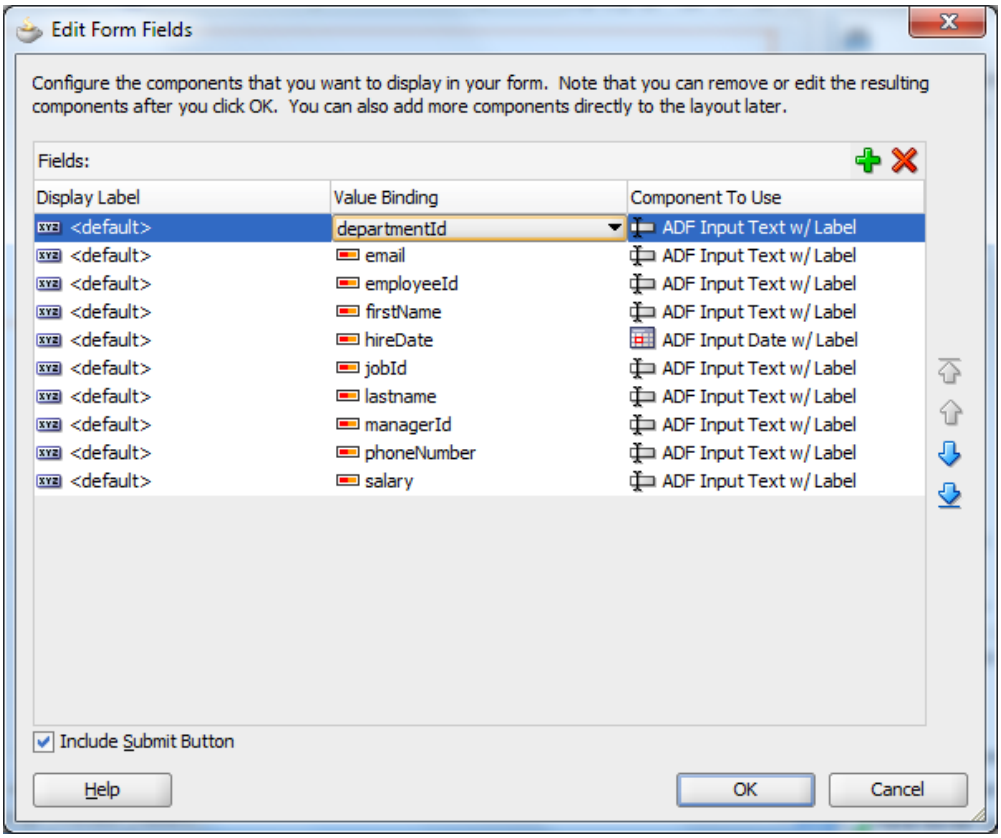
10. Drag the "arg0" node to the top area of the JSF page



11. Drop the "arg0" node as "Form | ADF Form" to generate the search form

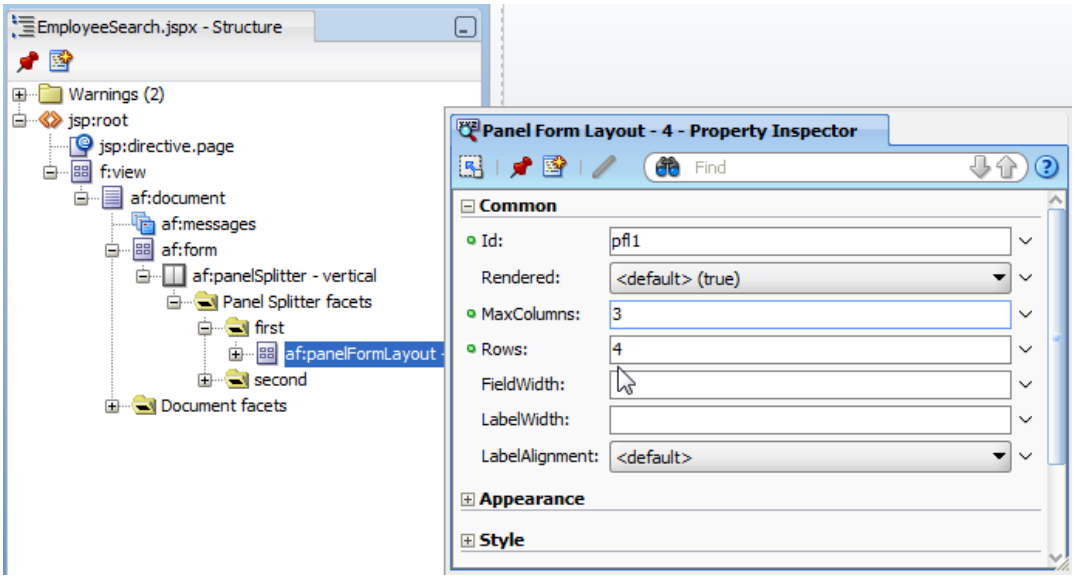


12. In the opened dialog, check the "Include Submit Button" option and "Ok" the dialog

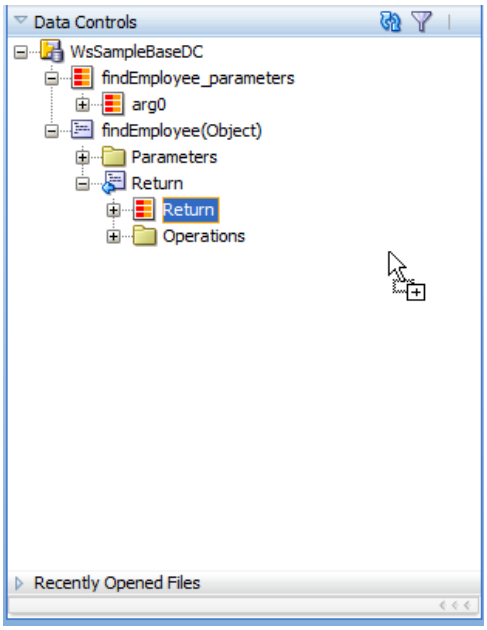


13. To create a multi column form layout, select the af:panelFormLayout component in the Structure Window. The Panel Form Layout component is contained in the first facet of the af:panelSplitter component.

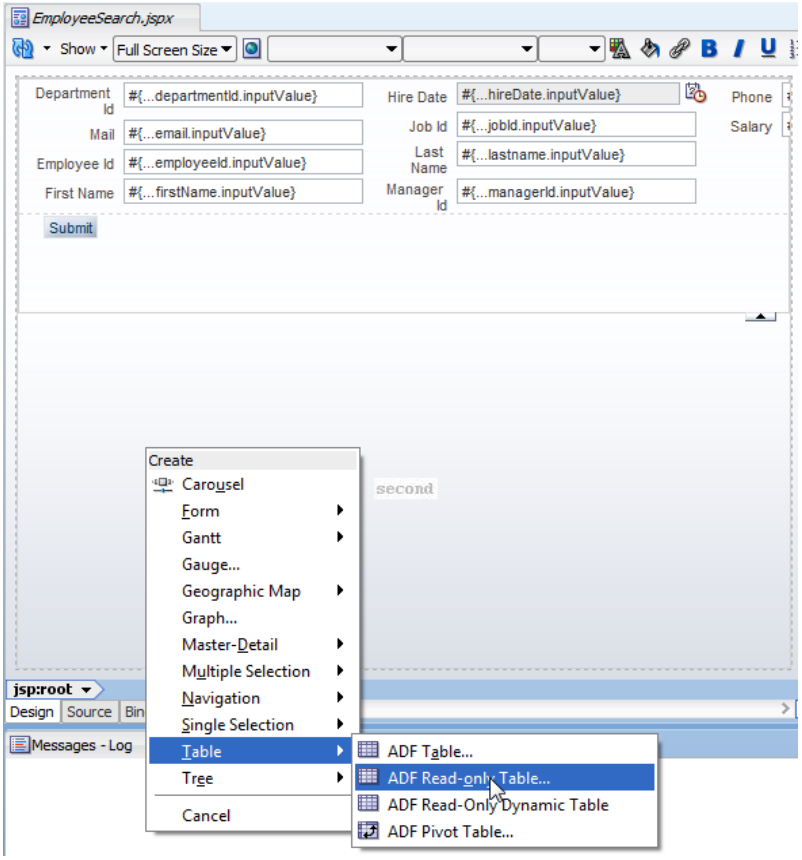
14. Open the Property Inspector and set the "MaxColumns" property to 3 and the "Rows" property to 4. The visual editor immediately changes the input form layout accordingly



15. In the Data Controls panel, select the "Return" collection under the findEmployee(Object) | Return node. The collection object describes the structure or the Web Service response to display in the table.

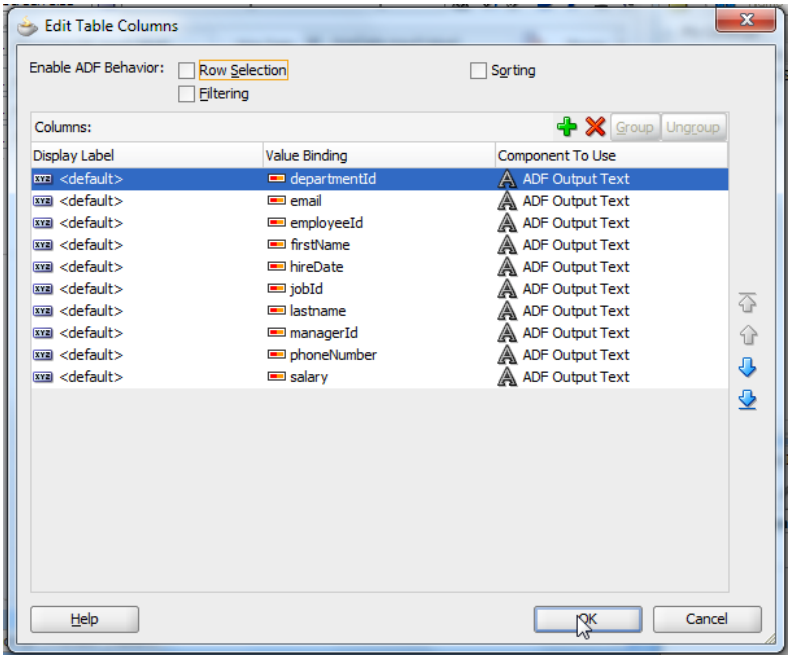


16. Drop the "Return" collection as a table to the second facet of the af:panelSplitter component



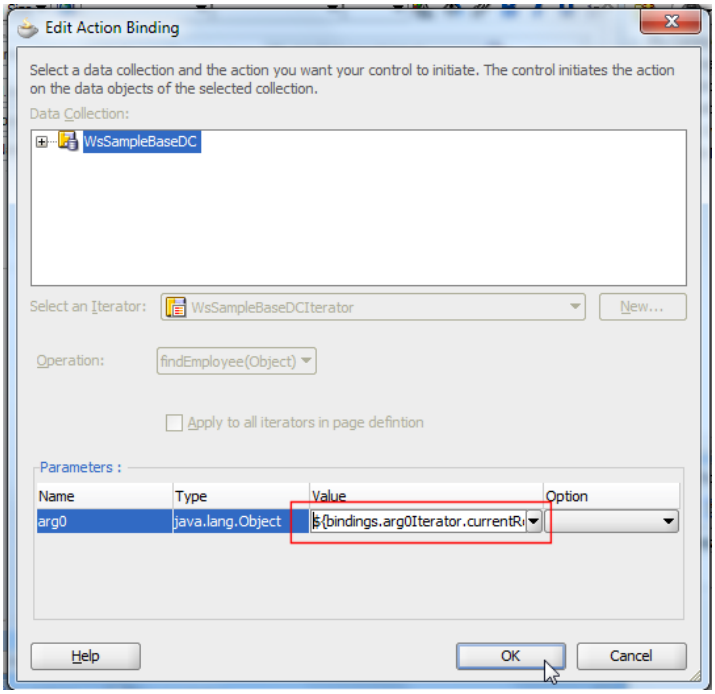
17. Optionally, re-order the attributes in the table display. Ignore row selection, filtering and sorting for now.

18. Ok the dialog

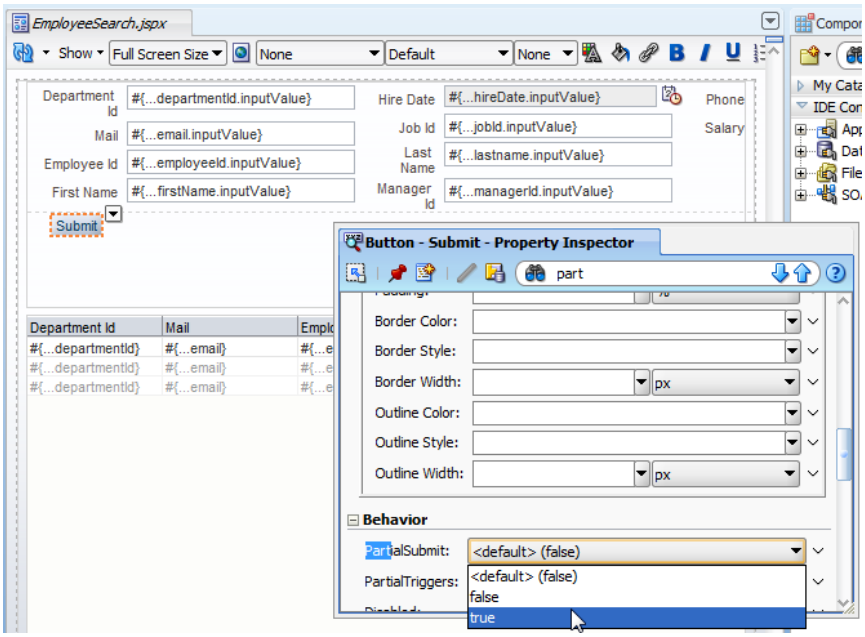


19. In the "Edit Action Binding" dialog there is nothing for you to do. Note the "Value" field for the "arg0" argument of the findEmployee(Object) method, which points to the variable iterator created when building the search form.

20. Close the dialog by clicking Ok.

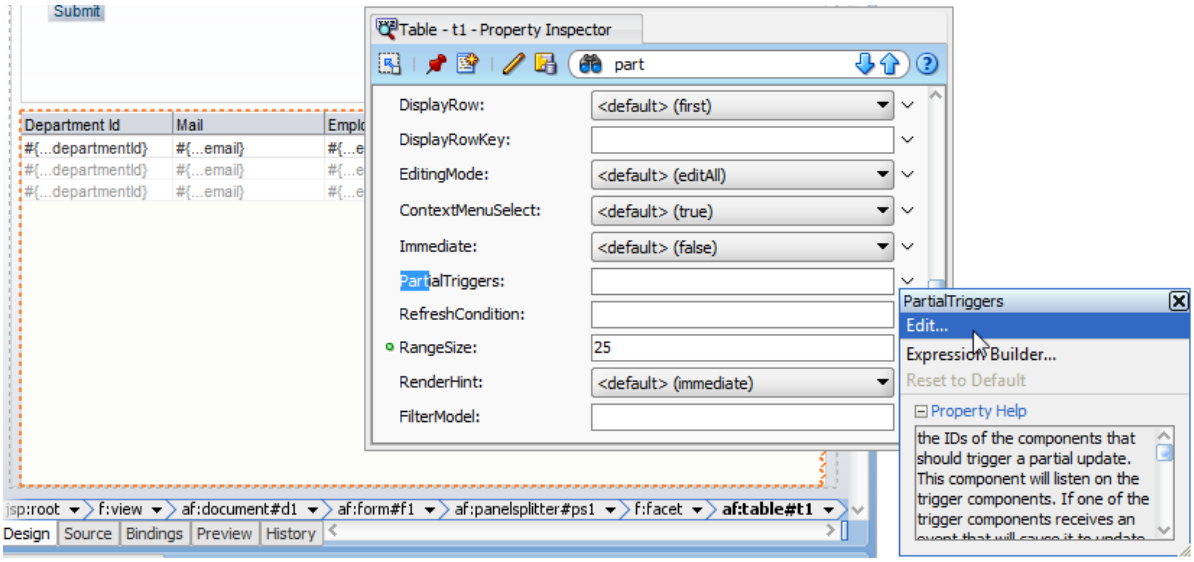


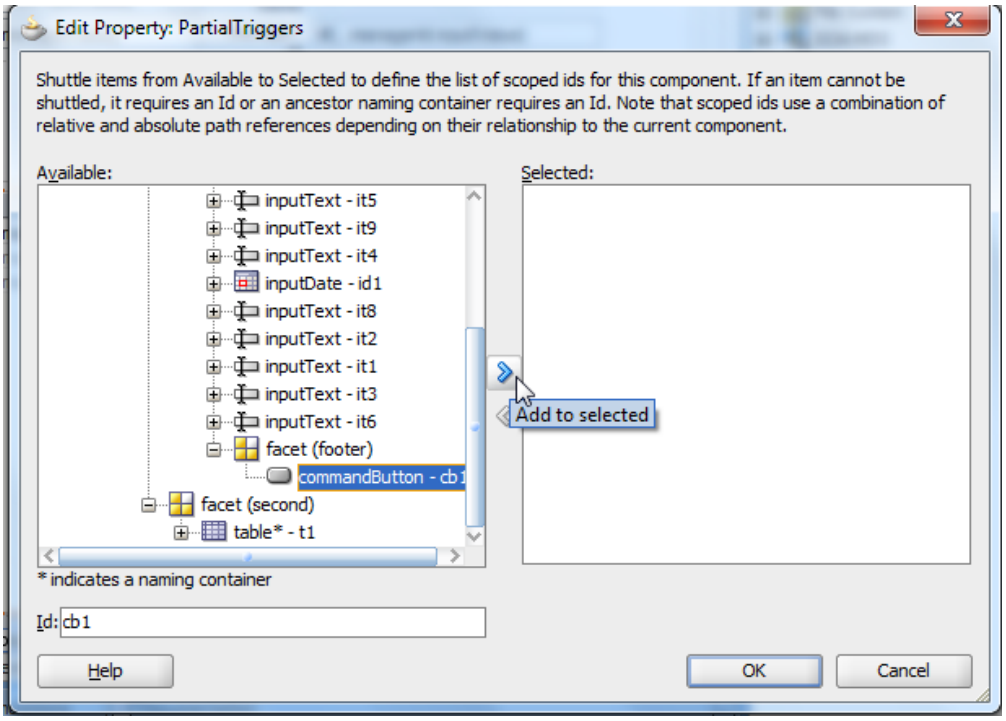
21. To avoid full page refreshes when searching for employees, you need to configure Partial Page Refresh (PPR) behavior. For this, select the "Submit" button of the search form and set its PartialSubmit property to "true".



22. Select the table for the result set and search for its "PartialTriggers" property. The PartialTriggers property allows an ADF Faces component to listen for a change on another component (for example a button press).

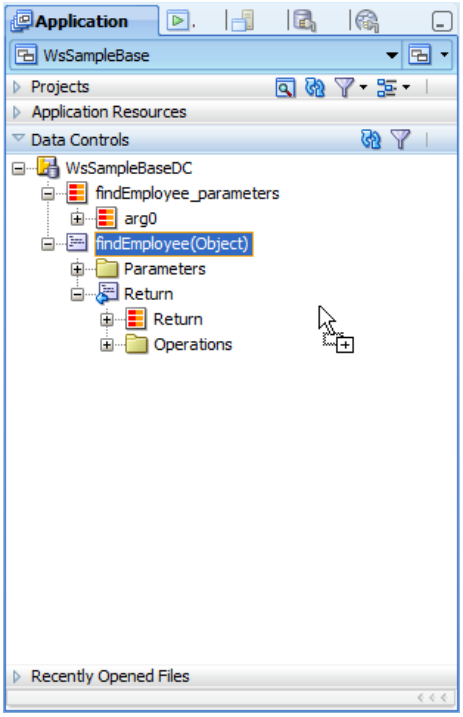
23. Click the "arrow down" icon at the end of the PartialTriggers property and select the Edit option in the opened context menu. Oracle JDeveloper supports you in referencing the UI component to listen for, avoiding errors that may occur with components contained in naming containers.



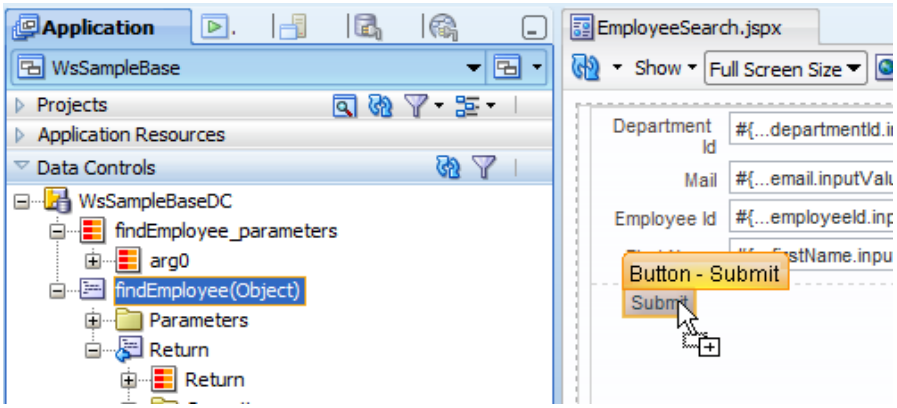


24. In the "Edit Property"dialog, find the submit button. The submit button is not shown by its label "submit" in the editor, but its component Id. You are able to find the button by its location, which is the first PanelSplitter facet, Panel Form Layout footer facet

25. Drag the findEmployee(Object) method from the Data Controls panel to the JSF page

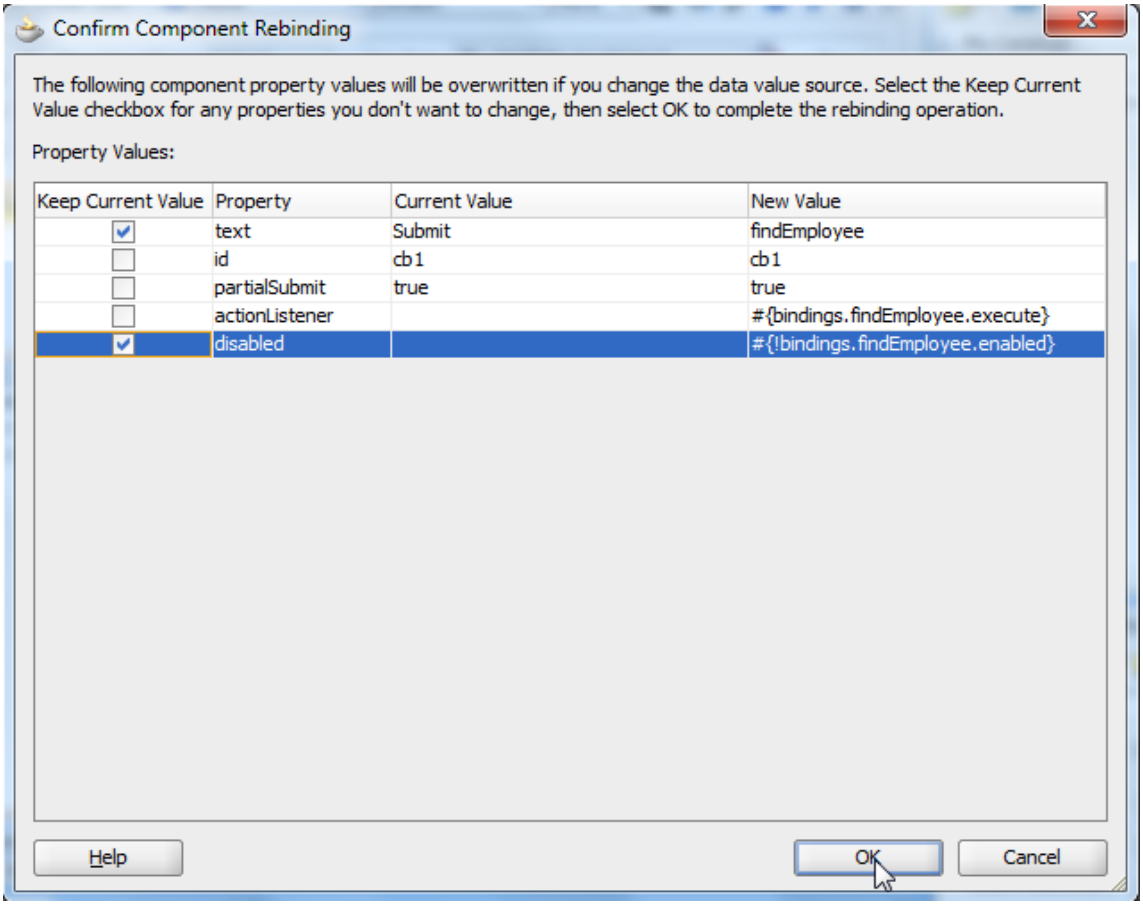


26. Drop the findEmployee(Object) method onto the submit button. The submit button so far would only submit the search form to the server, but would not yet invoke the findEmployee method to query the list of employees from the Web Service. Dropping the findEmployee(Object) method onto the submit button creates a method binding entry in the PageDef file and also configures the submit button ActionListener property to reference this binding.



27. In the "Component Rebinding" dialog, select the "text" property and the "disabled" property as you want to keep the current configuration for these.

28. Ok the dialog

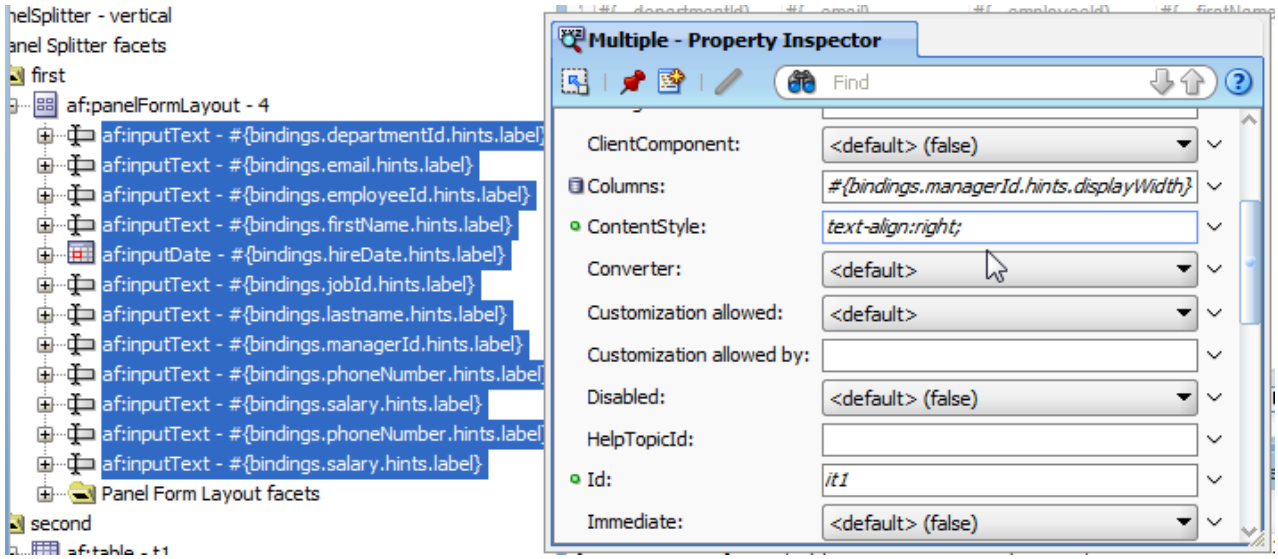


29. To ensure all search criteria is right aligned in the input text fields, a bit of CSS is required. Keep the CTRL-Key pressed and click on all components contained in the Panel Form Layout. The best place to do this is the Structure Window

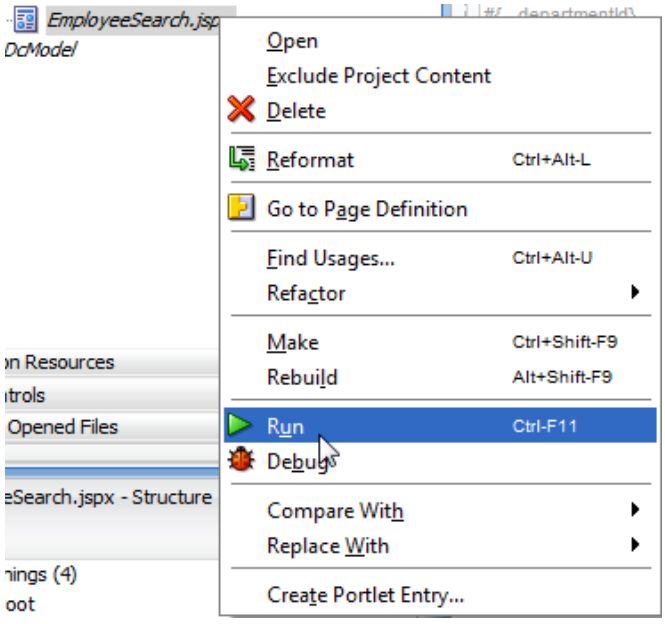
30. Open the Property Inspector, navigate to the "ContentStyle" property and add

text-align:right;

The "ContentStyle" property ensures the CSS is directly applied to value area of the input components.

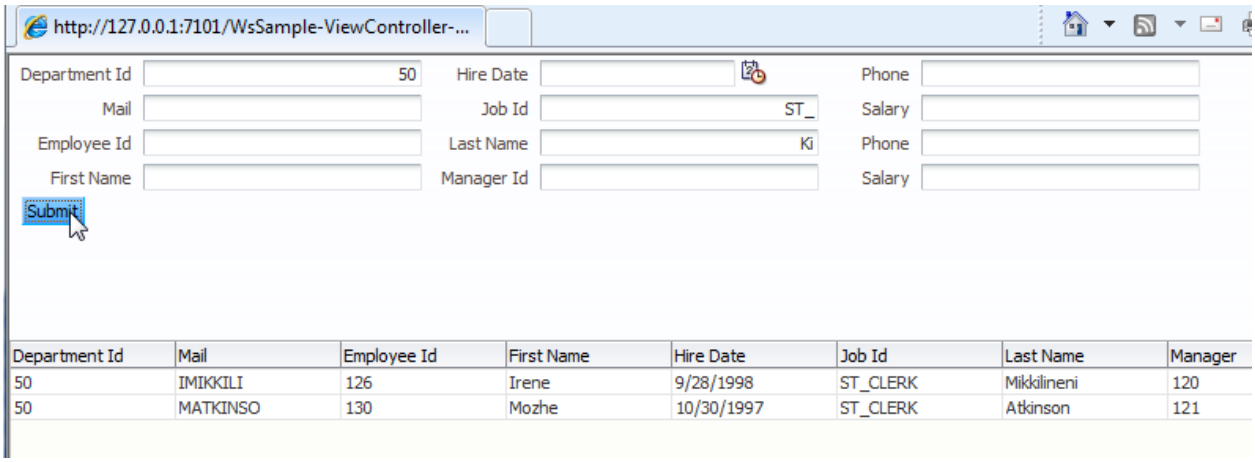


31. Select the EmployeeSearch.jspx page and choose "Run" from the context menu



32. Note the search field and table header labels, which are read from the Data Control UI Hints you created before.

33. Move the mouse over a search field and wait for the tool tip to show – which is also read from the DataControl configuration

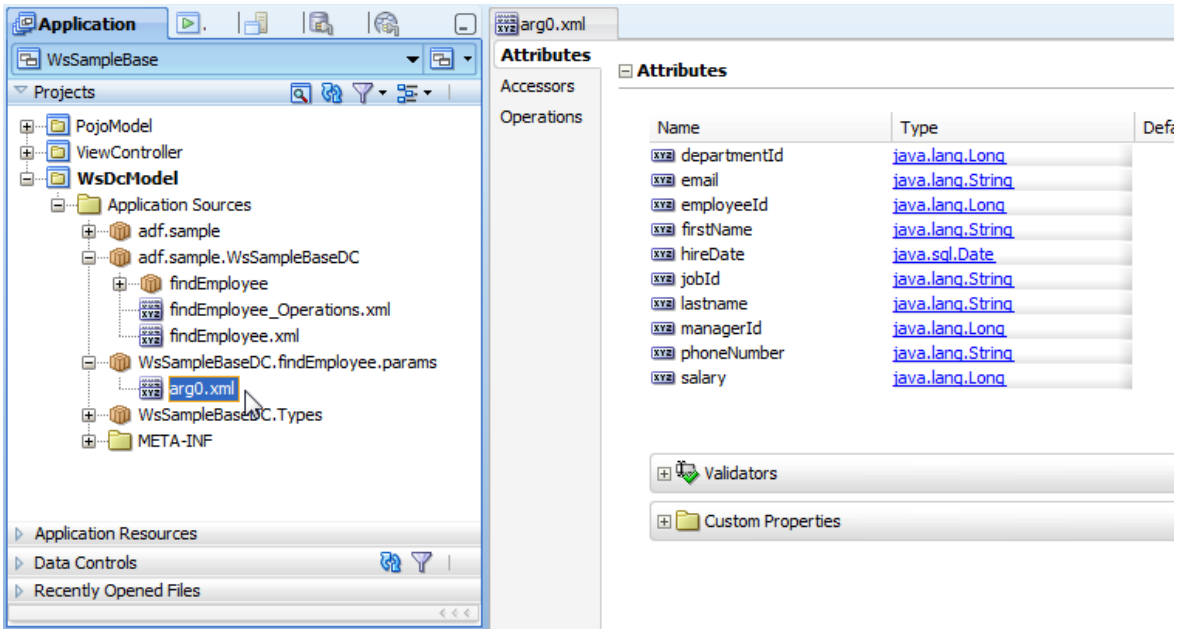


34. Type valuesinto the search form field, as shown above, and press "Submit". The table should refresh with the data queried from the WS

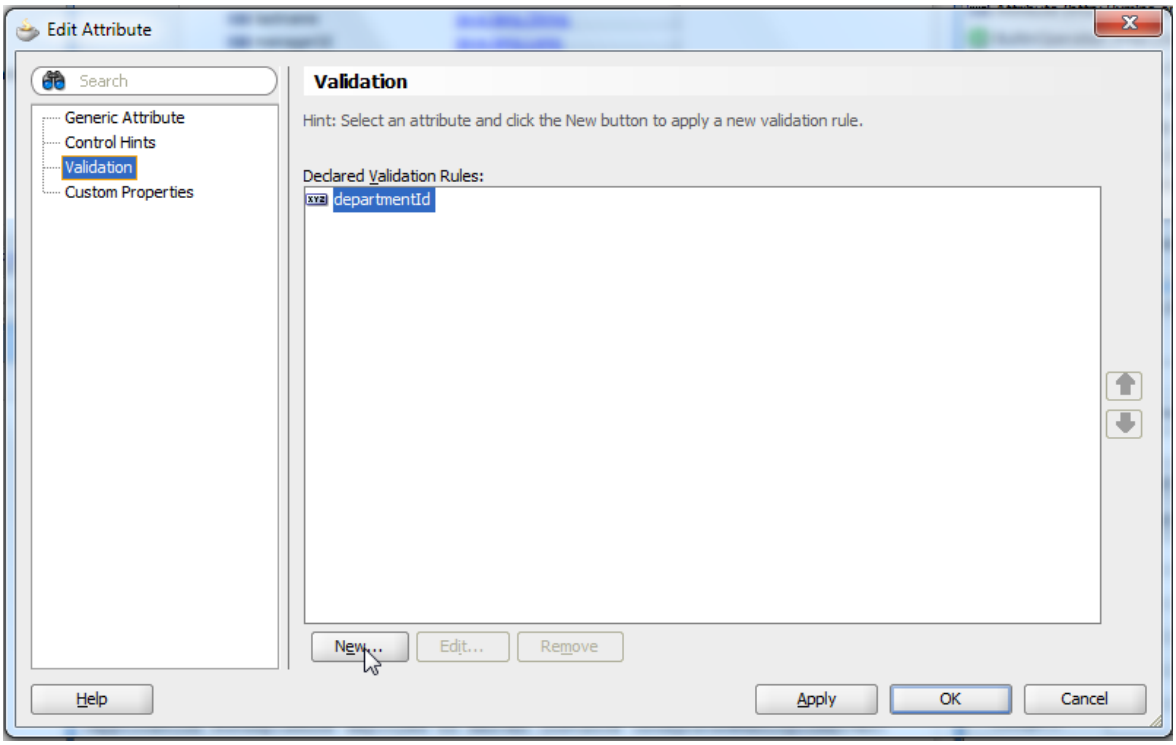
Adding Model Validation

An advantage of using the WS Data Control is that it allows you to define validation rules, like range validation or pattern validation, like mail address verification. So let's add validation to the project. What about a pattern validation that ensures the entered departmentId to search for is between 10 and 999 ?

1. In the WsDcModel project, select the "arg0" entry contained in the WsSampleBaceDC.findEmployees.params package

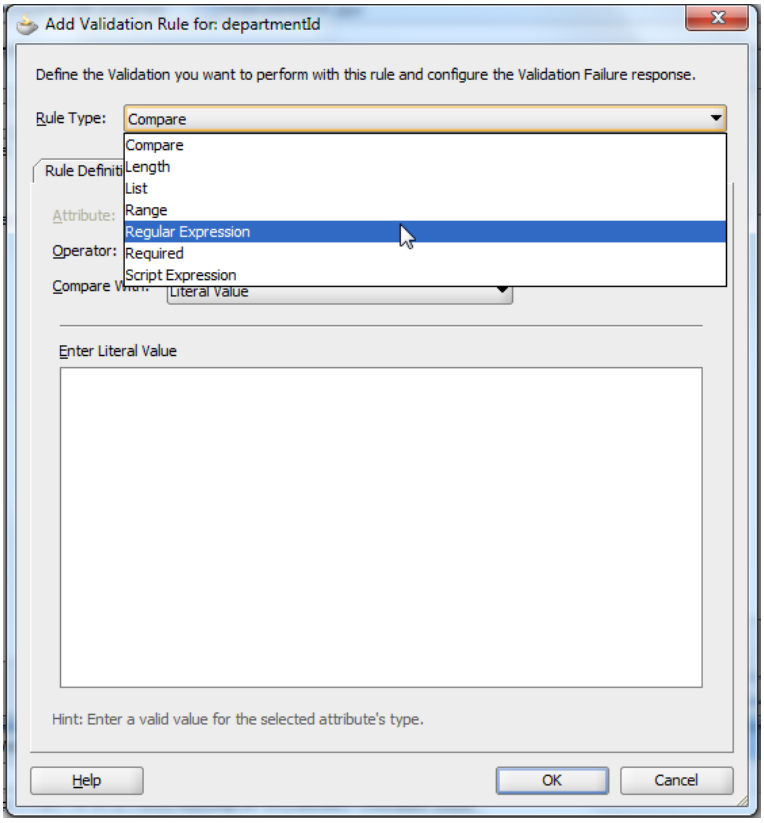


2. Double click the "arg0" entry to open the attribute editor.

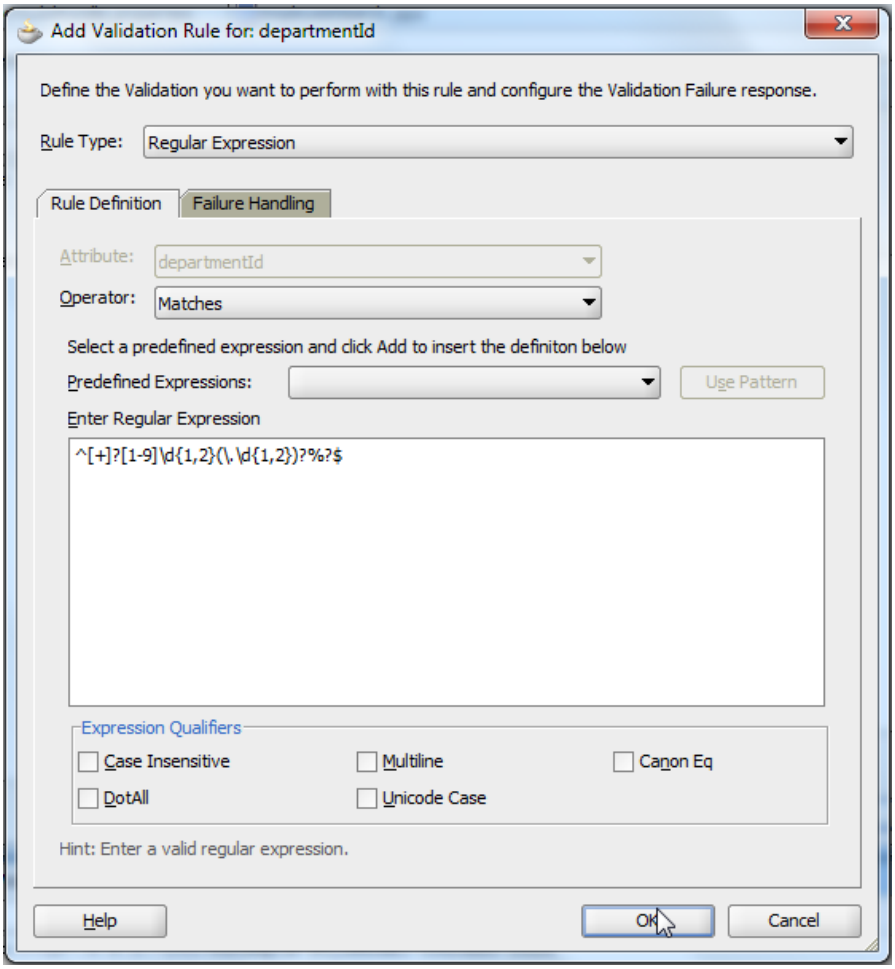


3. Select the departmentId entry and click the edit icon (pencil icon)

4. In the "Edit Attribute" dialog, select the "Validation" entry and click the "New" button



5. Select "Regular Expression" as the "Rule Type"

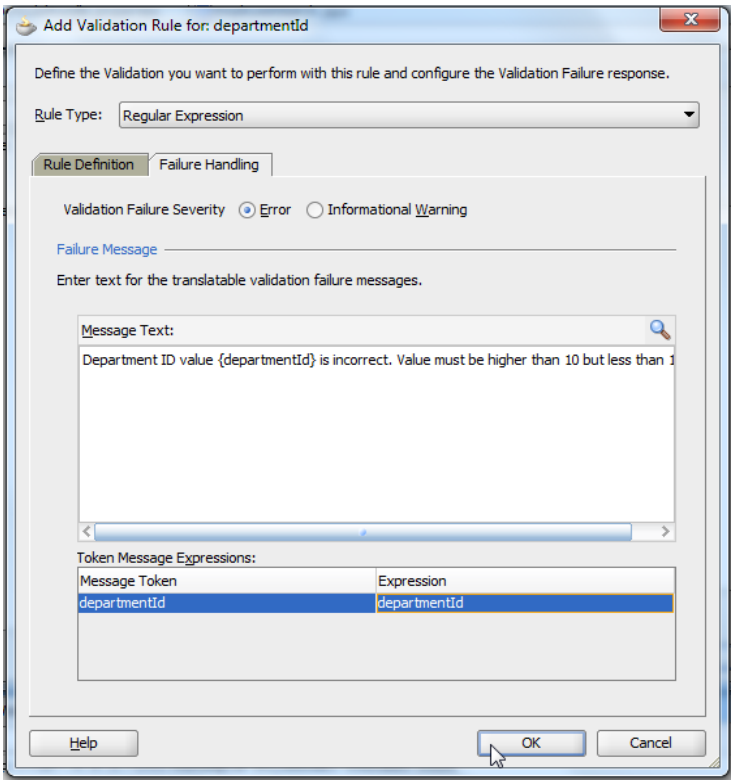


6. In the "Enter Regular Expression", add

^[+]?[1-9]\d{1,2}(\.\d{1,2})?%?$

Note: You can copy and paste this String from above

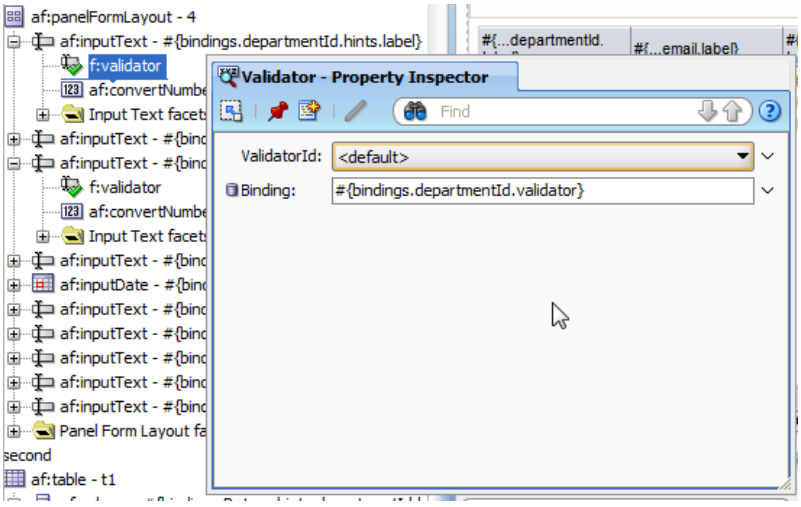
7. Select the "Failure Handling" tab Here we specify an error message, which also can be translated into different languages



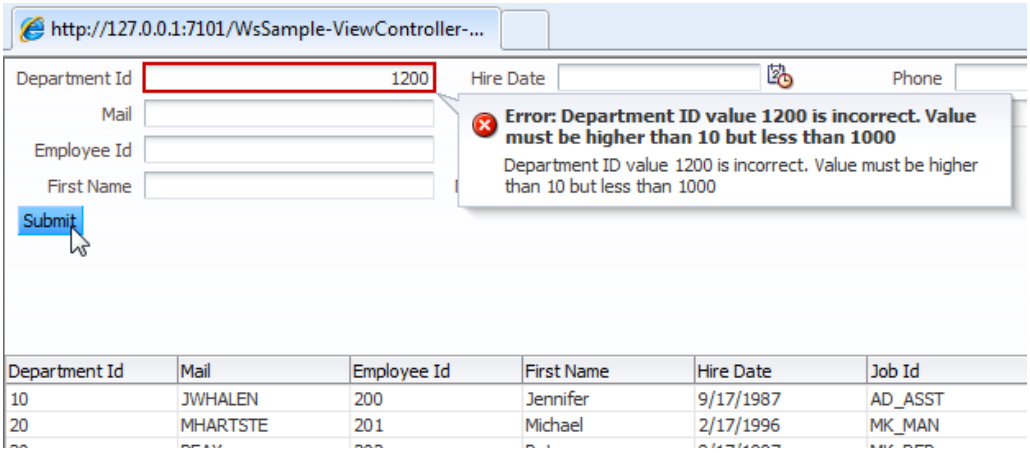
8. Add a message string like above saying that the provided value must be between 10 and 999. Make sure you add {departmentId} somewhere in your text.

9. The {departmentId} flag becomes a variable that allows you to add information from the attribute into your message. In the example we want to tell the user that the value he/she entered is not valid. For this add "departmentId" to the Expression field.

10. To see why and how this works, go to the ViewController project and select the departmentId input text component "f:validator" tag in the Structure Window and open the Property Inspector. Note that the "Binding" points to the ADF binding's department Id attribute binding, and here the "validator" attribute

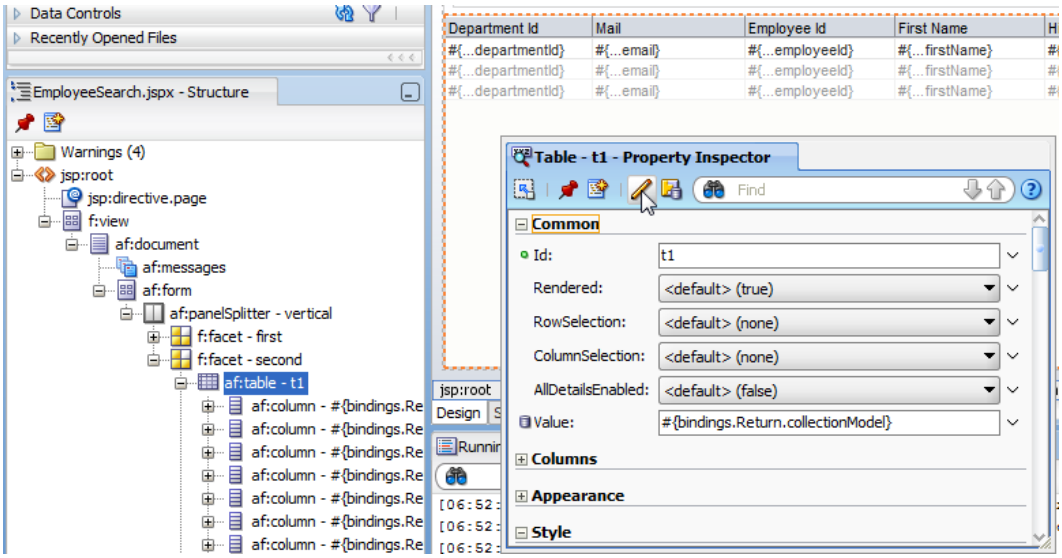


11. Rerun the search page and type 1200 into the departmentId field. Press submit and see the error message thrown by the binding layer.



Enable Table Filtering and Sorting

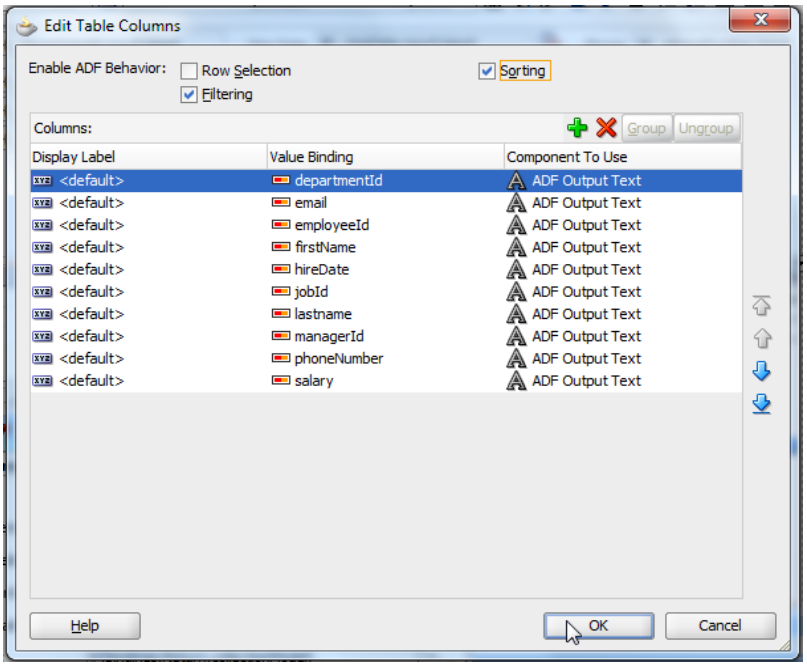
As a last exercise in this hands-on, you add table filter and sorting functionality to the result table.



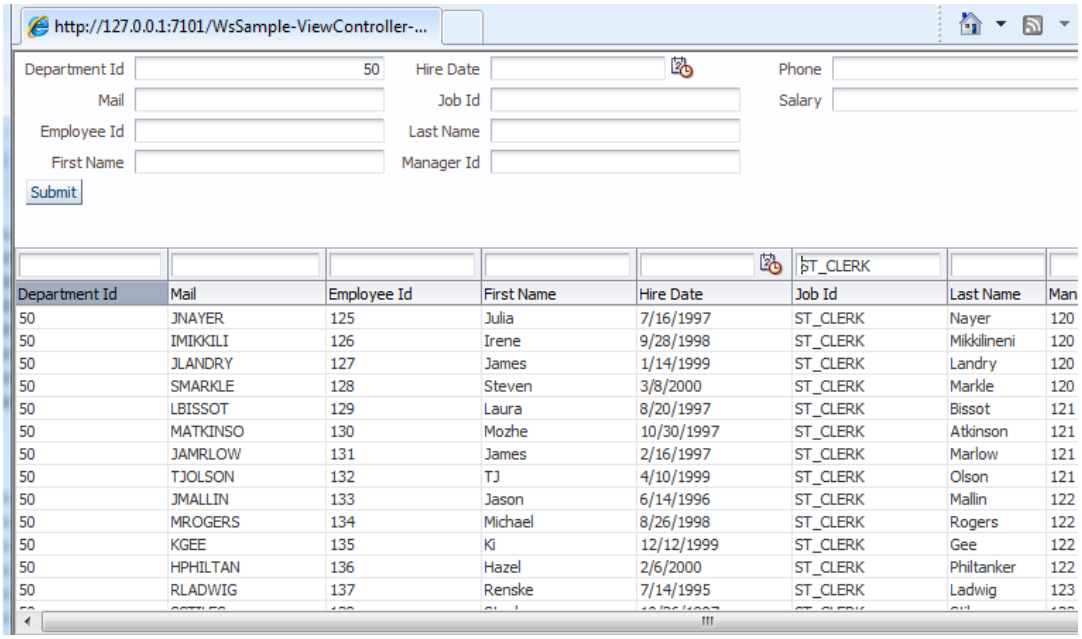
1. Select the table component in the Structure Window and open the Property Inspector

2. In the Property Inspector, click the "pencil" icon to launch the component configuration

3. In the "Edit Table Column" dialog, check the "Sorting" and "Filtering" option. The "Filtering" option adds a search binding in the PageDef configuration file and references it from the table's QueryListsner property



4. Click OK and run the search page from within JDeveloper

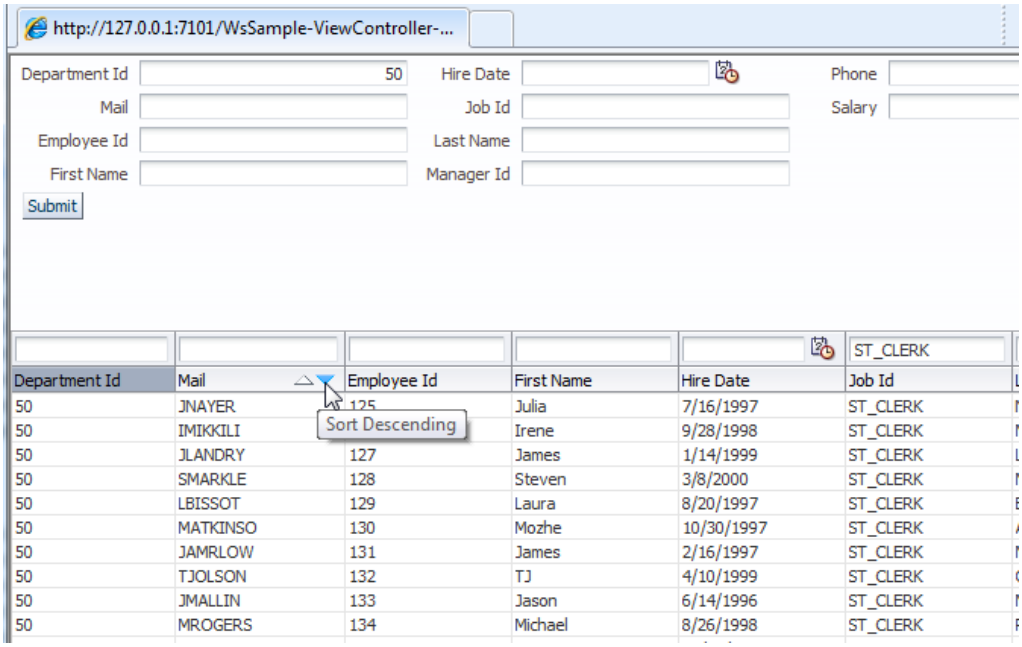


5. In the search form, enter 50 for the departmentId and press the Submit button

6. Type ST_CLERK into the JOB_ID filter of the table.

7. Hit the ENTER key to filter the table

8. Click the Sort icon in the "MAIL" column header to sort the queried and filtered result set



Downloads

The Oracle JDevloper starter workspace can be downloaded in a ZIP file as sample #73 from the ADF Code Corner website at

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

RELATED DOCOMENTATION

☒	
☒	
☒	