ADF Code Corner

054. How-to build a search form using the Oracle ADF Web Service Data Control and Complex input types

twitter.com/adfcodecorner

Abstract:

Web Service operations may use complex input types in the arguments. For example, querying a list of employees may expect an Employee object as the search argument. The Oracle ADF Web Service Data Control resolves the complex argument if its structure is contained in the Web Service definition (WSDL) , which usually is the case when using JAX-WS services. The resolved object structure however is not directly exposed with the WS operation and thus a bit - though declarative - of extra work is required to build an input form from it. This blog article explains how to build a search form from an operation defined in a POJO Web Service.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
18-JUN-2010

Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

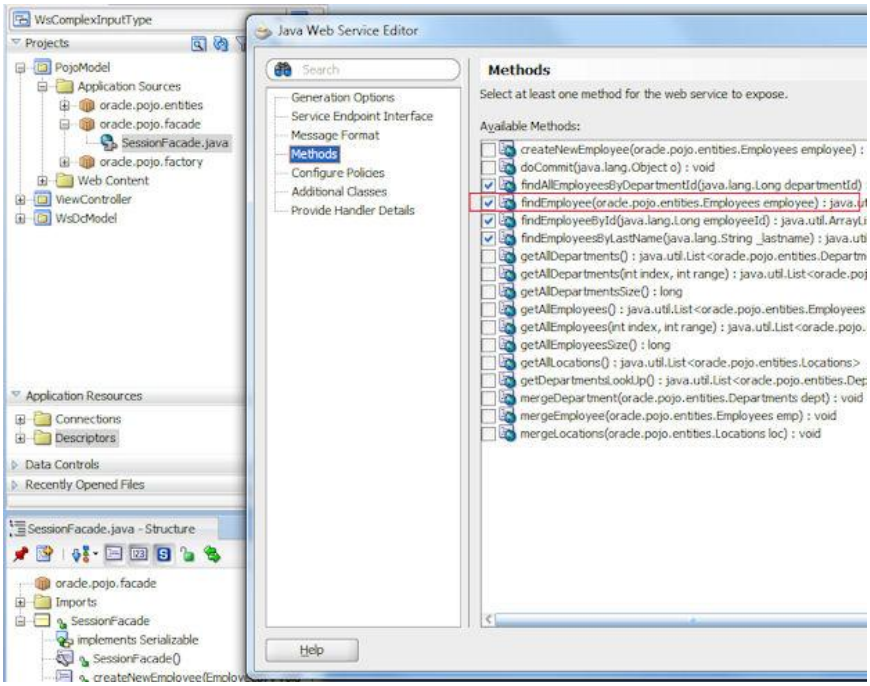
Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: http://forums.oracle.com/forums/forum.jspa?forumID=83

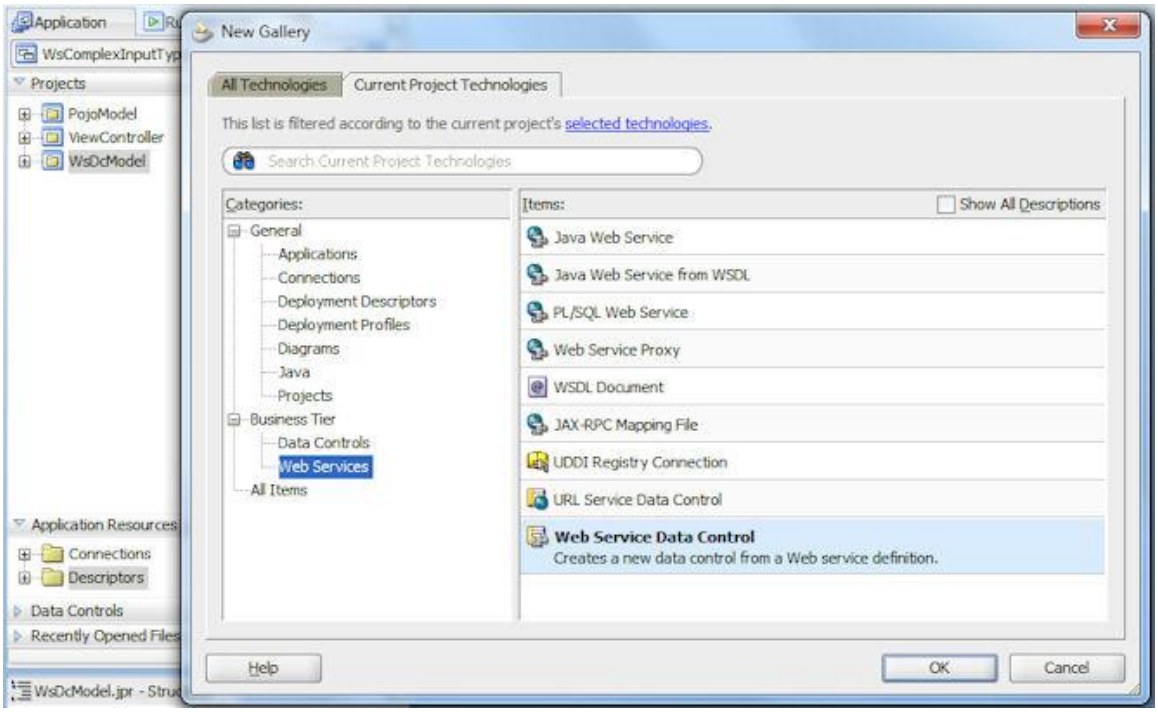
Introduction

In this blog article, I am stepping through the development of a search form that is based on a POJO Web Service operation that requires a complex object type as an input parameter. The use case is a form to query employees, which in this example are coded within the POJO itself, based on employee properties that are exposed in a form. The screen shot below shows the Web Service definition and the operations it exposes.

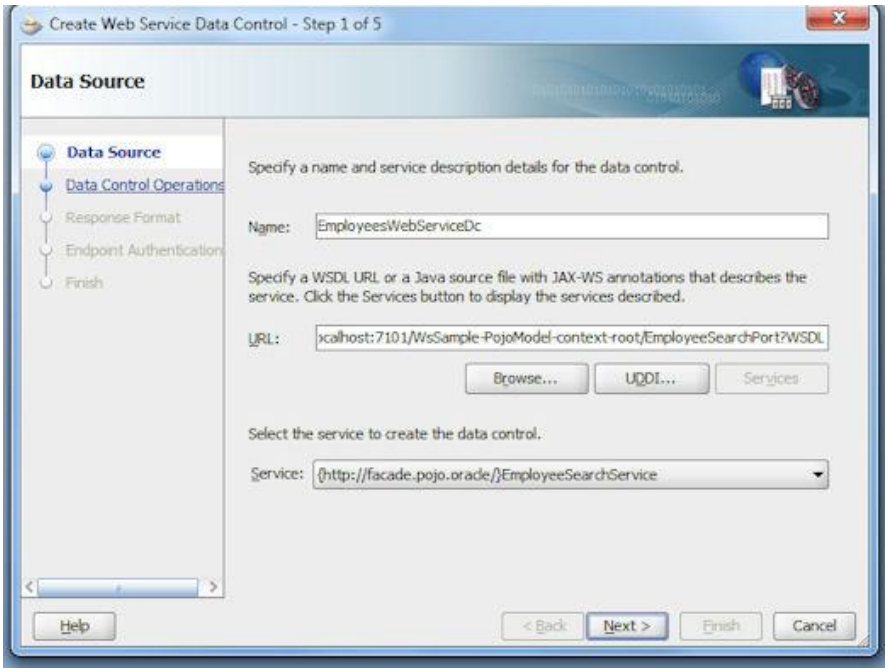
Creating the WebService Data Control



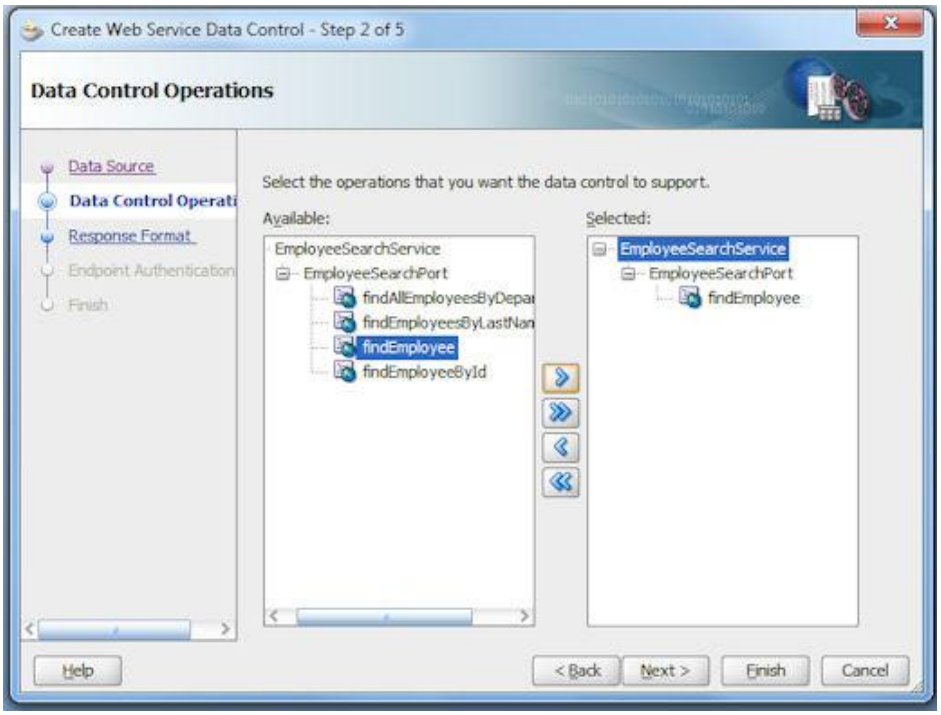
After deploying the Web Service, which you can do by running the Web Service tester using the integrated Weblogic Server, the Web Service Data Control is created from a WSDL reference. To launch the Web Service Data Control wizard, choose New | Business Tier | Web Services | Web Service Data Control from the context menu displayed when clicking the project with the right mouse button.



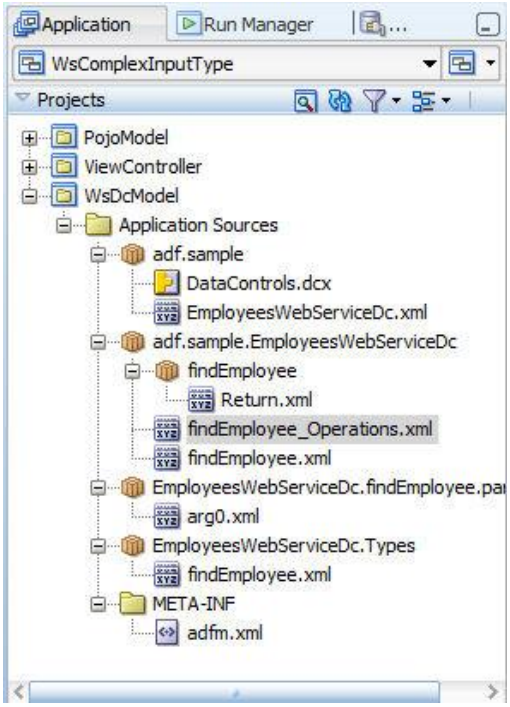
In the Web Service Data Control wizard, specify the name of the Data Control as it should appear in the Data Controls panel and provide the WSDL reference to the deployed Web Service,



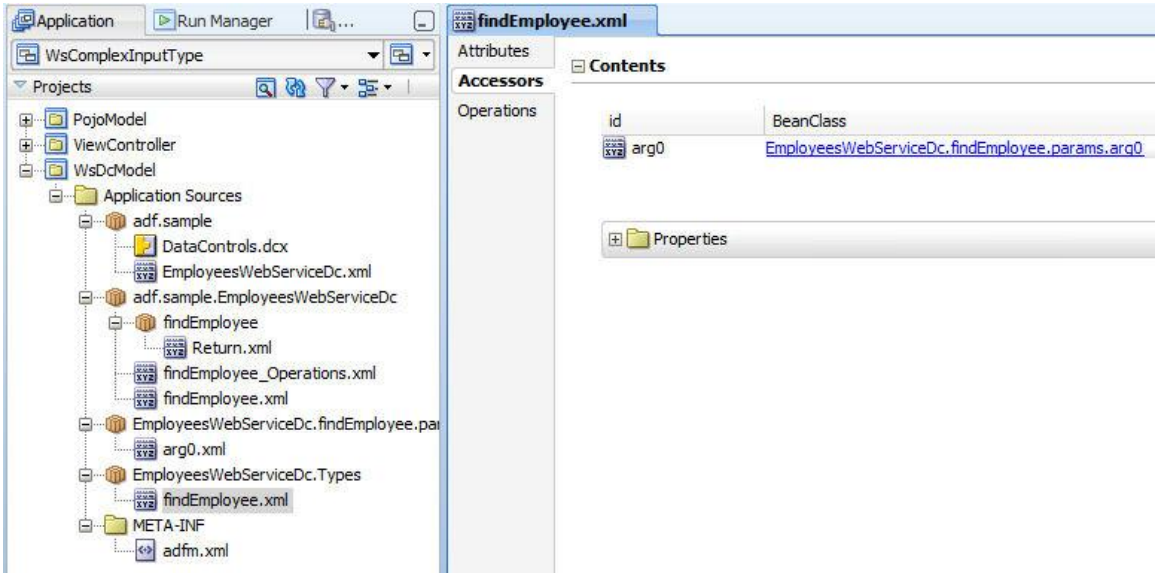
The Web Service Data Control Wizard allows developers to further refine, which Web Service operations should be exposed to the ADF application developer. In this example, only one of the four possible search operations shall be exposed in the Data Controls panel.



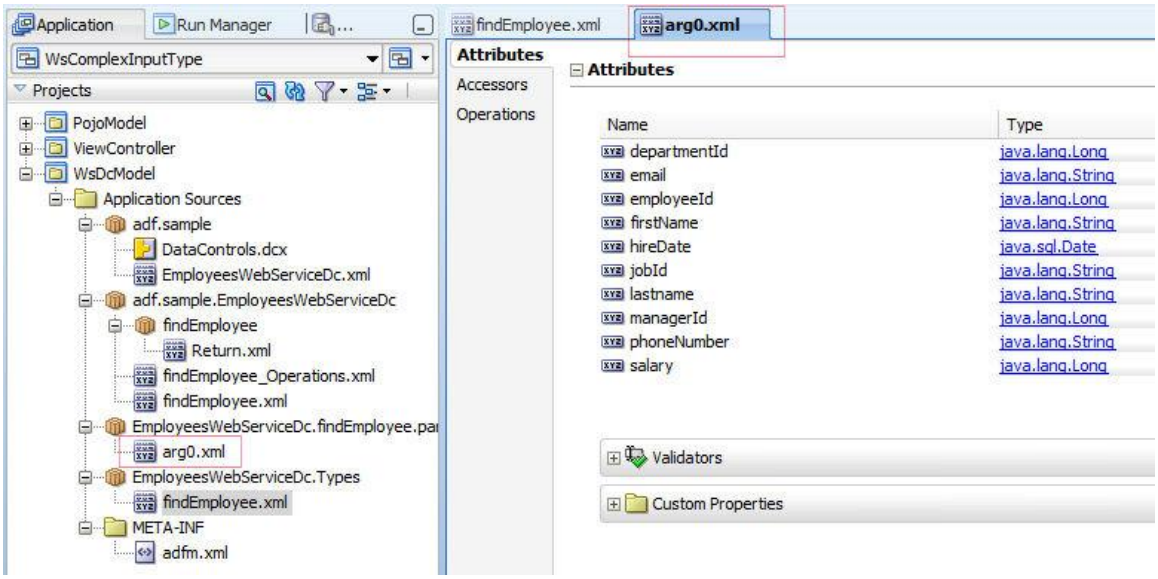
The Web Services Data Control generates metadata files that describe the operation and the service to call. The findEmployee_Operation.xml file defines the search method used in this example. Double click this file to open the visual editor.



The visual editor shows the operation argument as arg0, which is not what you would have expected. Instead, you would want to see the attributes that make the input object. But don't worry, its all there. Click the link shown below the Bean Class header and JDeveloper navigates to another XML file, the arg0.xml file, which now shows the flattened object the Web Service operation expects as an input argument.

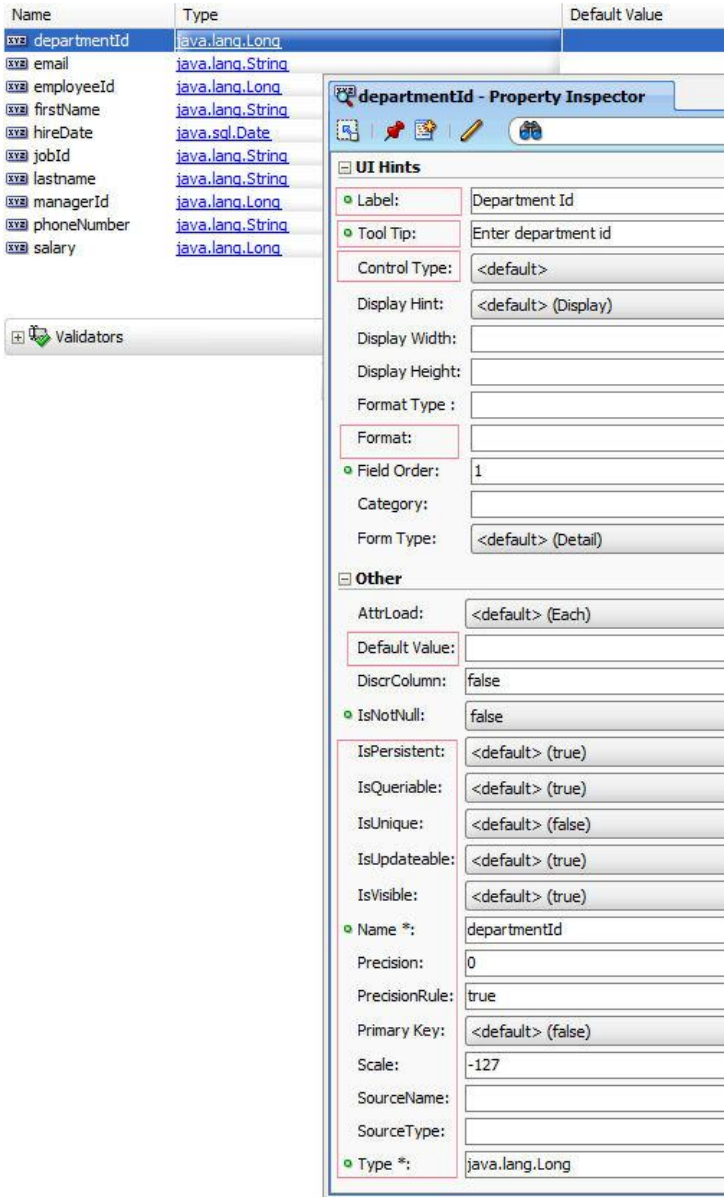


Each of the attributes - or properties if you prefer - can be configured with UI hints using the Property Inspector, or a dialog that opens after selecting an attribute and clicking the pencil icon (not shown in this image).



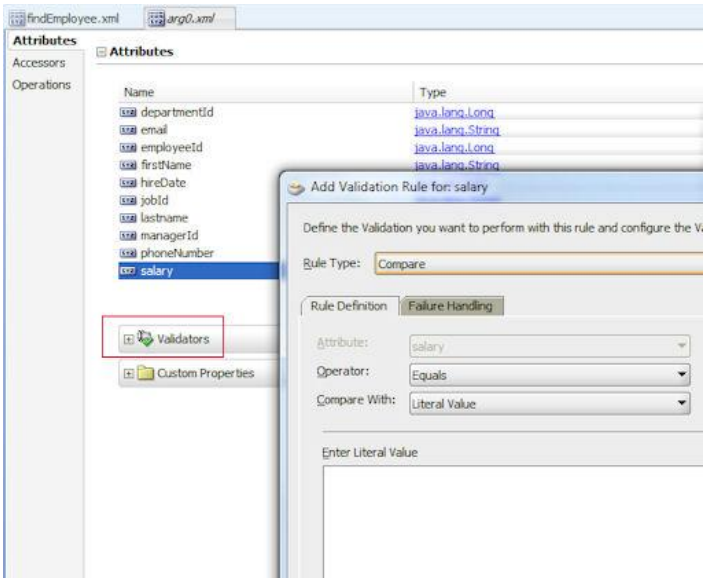
The possible configuration includes translatable labels and tooltips that are automatically displayed when creating UI components from the attribute.

Also, you can specify the default UI component used to render the attribute, when dragging the argument object as a form onto an ADF Faces page. This allows you to define which attribute for example should be rendered as a list item and which should be rendered as an input text field. Also pay attention to other properties, like a default value that can be defined to consistently show for all usages of the data control. If the object is used in an input form for inserting and updating data, you may want to set the primary key and updateable properties. Attributes that are not set to updateable will render read only on the ADF Faces page (and I've seen many developers getting confused by this read only rendering). The benefit of defining attribute properties is consistency.

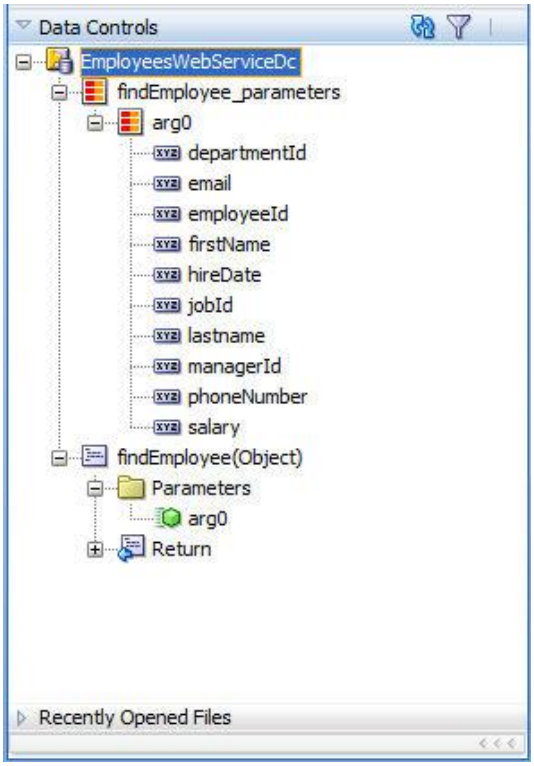


In addition to the attribute properties, you can define validation rule, which are enforced by the ADF binding layer. For example, Web Services attributes may expect a specific format or value range that can be defined in this central location. To open the validation dialog for a selected attribute (salary in the

screen shot), you press the green plus icon (not shown in the image) next to the validators" header. It makes sense - and I recommend it - to spend some time with customizing the meta data like this as otherwise you risk to leave a key strength of the ADF binding layer unused.



The image below shows the Web Service in the Data Controls panel. Collections, which you can use to build form and tables from are indicated by a yellow/orange icon. Methods are indicated by a code sheet icon. As you can see, the arg0 argument is displayed as a collection, listing all its attributes. With this view, we are ready to build the search page.

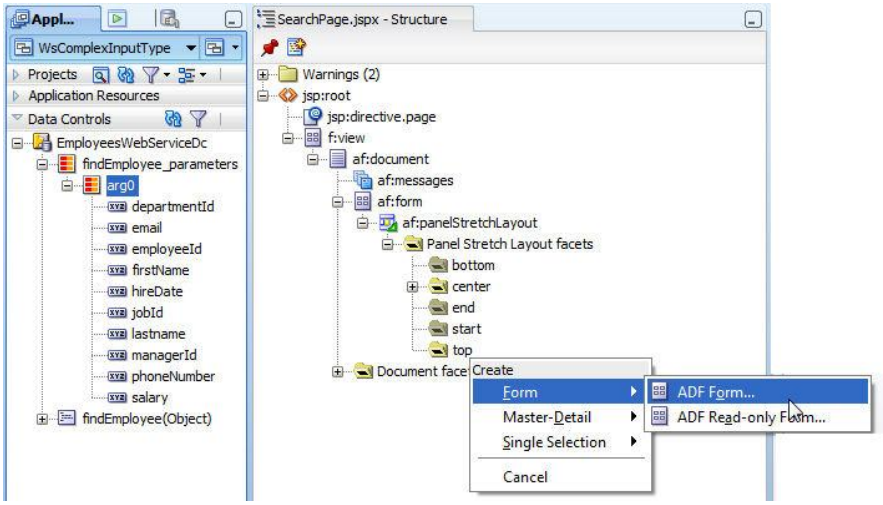


Building the ADF Faces View

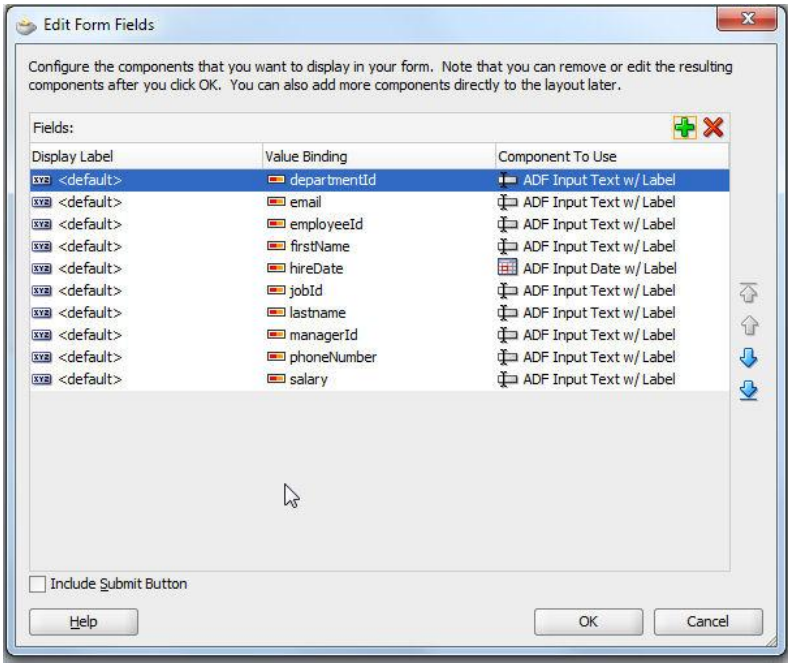
Using the ADF Faces quick layout feature, I created a page consisting of a panelStretchLayout component and an af:form tag. To create a new JSF page, use the right mouse button to click onto the ViewLayer project. Then choose the JSF entry.

The image below shows the Data Controls panel and the Structure Window. The Structure Window represents a hierarchical view of the page, which at the same time may be displayed in the visual WYSIWYG editor. There are different options available to construct the search form, but lets go for easy:

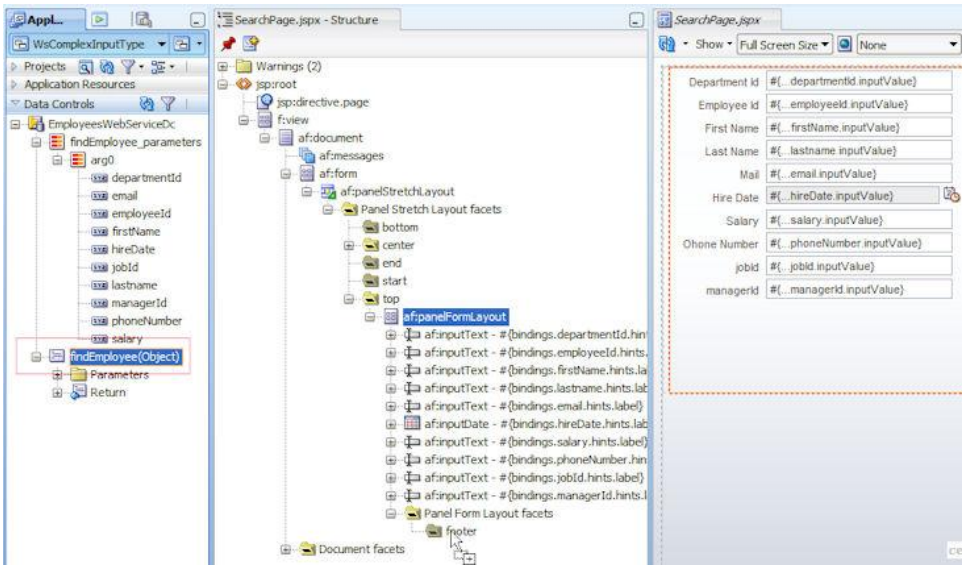
- Select the arg0 node and drag the collection onto the page. In this example, the drop is into the "top" facet of the panelStretchLayout component so the search fields show above the table displaying the findings. After dropping the collection, a context menu opens to create the form



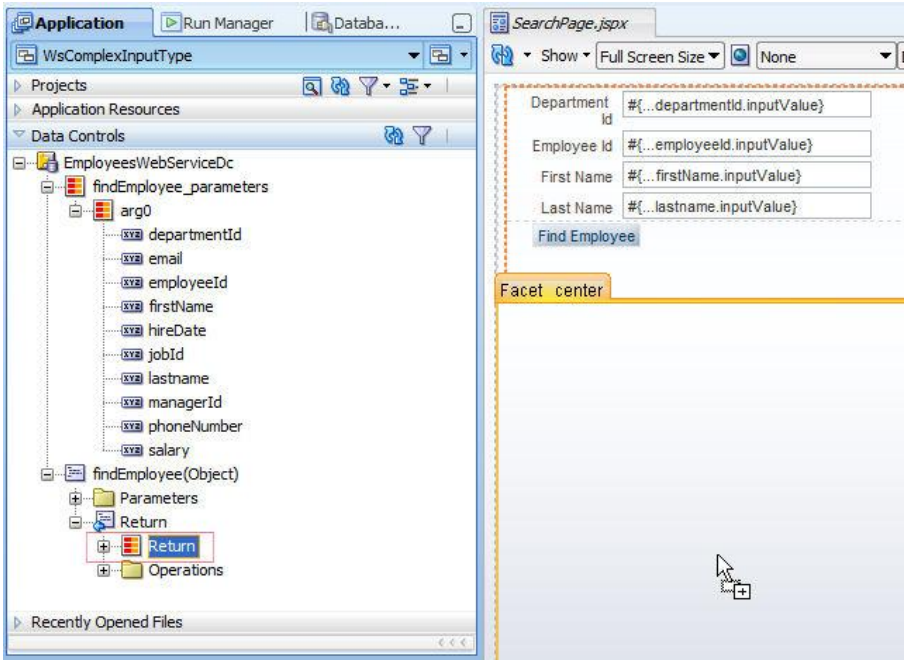
- In the opened form dialog, you can re-order the attributes to best meet the user expectation



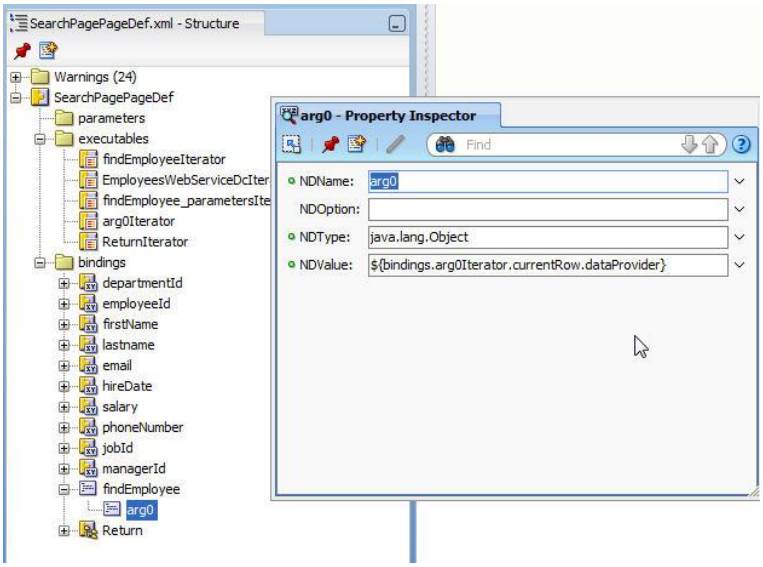
- Once the form is created, select the panelFormLayout component and open the Property Inspector. Set the row property to 4 and the maximum columns property to 3 to arrange the search form into columns.
- Drag and drop the findEmployee method from the Data Controls panel onto the page. This time, drag the button into the "footer" facet of the panelFormLayout component.



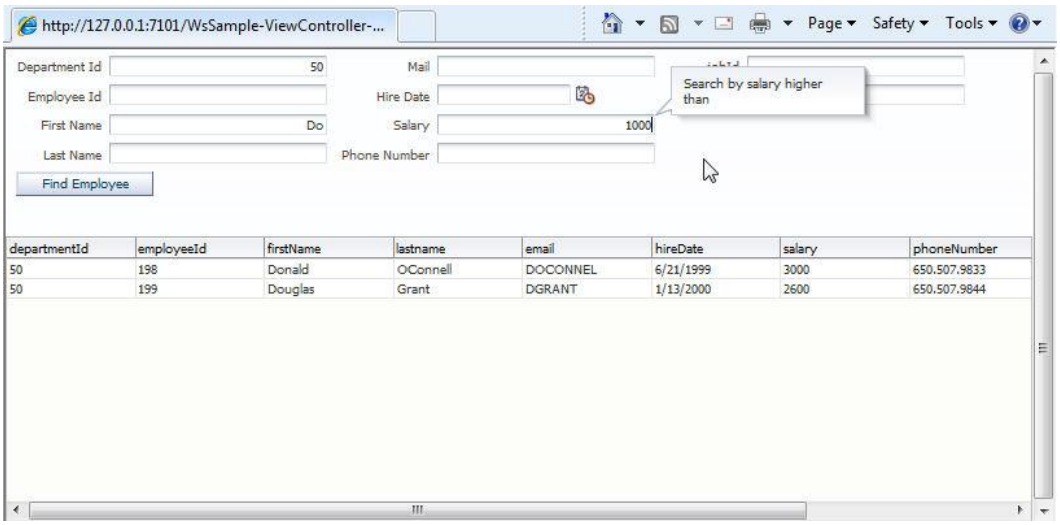
To show the search results, drag and drop the "Return" collection of the findEmployee method as a table into the "center" facet of the panelStretchLayout component. To optimize the user experience, you may want to set the search button's partialSubmit property to true and reference the search button Id from the table's partialTriggers property. This way you avoid a full page refresh when submitting the search.



Implicitly, the pageDef file - which represents the binding container at runtime - is configured with the attribute bindings and iterators. The findEmployee method is contained in the bindings section and is referenced from the actionListener property of search command button. The method binding argument references the arg0Iterator, which takes the search arguments from the input components. The iterator currentRow,dataProvider is the Employee object, which - surprise, surprise - is exactly the object type the Web Service operation expects.



Running the JSPX file in the sample, the search form shows as in the screen shot below. Note that by default all table data is displayed, which has to do with the way I coded the search function in the Web Service. The search function treats null as the search condition of an attribute in the search argument as a wild card. Therefore all data is queried initially. If you don't want this, change the search functionality, or change the refresh condition of the iterator in the PageDef file.



Hint: To enforce the right alignments in the input fields, I used stylesheet added to the contentStyle property of the input text components. For a sample, this is a valid approach. In a real application development project, this would better be added into a skin file using the af|inputText::content selector

Download Sample

The sample shown in the screenshots can be downloaded from ADF Code Corner

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

It is a JDeveloper 11.1.1.3 workspace that uses a POJO model for the Web Service. The Search in the input text fields is defined such that it checks if the provided search string is contained in the value. Number values must be exact, or in the case of salary be below the salary to search for. If you need to change the search functionality (after all, it is a demo) you can do this by changing the findEmployee method in the EmployeeList.java class.

RELATED DOCOMENTATION

☒	
☒	
☒	