ADF Code Corner

70. How-to build dependent list boxes with Web Services Business Services

twitter.com/adfcodecorner

Abstract:

A frequent question asked on the Oracle JDeveloper forum on OTN is how to create dependent select lists using ADF and Web Services as the business service.

In this article, I walk you through building a sample that uses dependent choice lists from a POJO Web Service model.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
21-JAN-2011

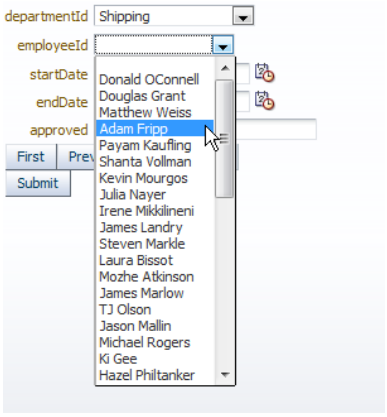
Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

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.

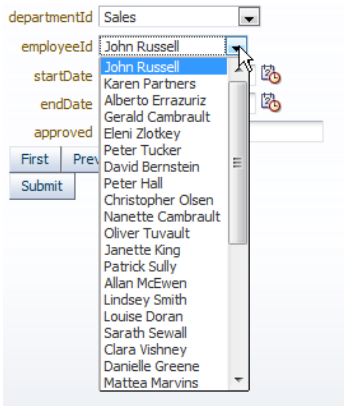
Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: http://forums.oracle.com/forums/forum.jspa?forumID=83

Introduction

The sample application simulates a vacation request form that has dependent lists set up between the **departmentId** and **employeeId** field. When starting the application, two vacation request records are available. Each of them shows an employee and the department this employee works in. Selecting a different department refreshes the dependent employee list with employees name associated with it, as shown in the images below



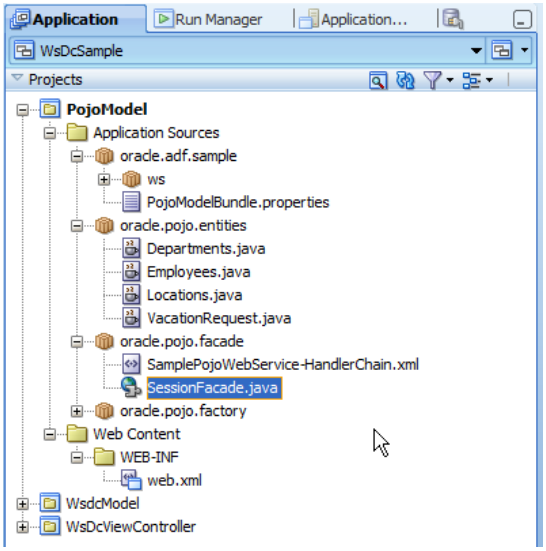
Changing the department changes the detail list



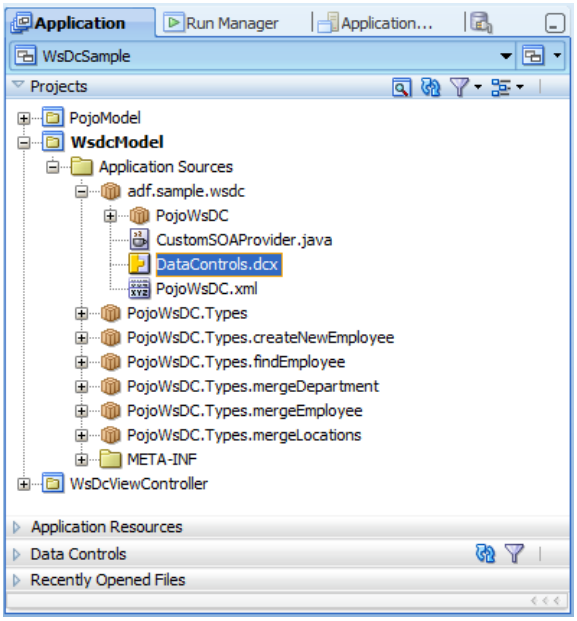
In the following, I step you through the creation of this sample. You may want to download the sample zip file from ADF Code Corner, open it and create a new empty ViewController project to try building the sample yourself.

About the sample

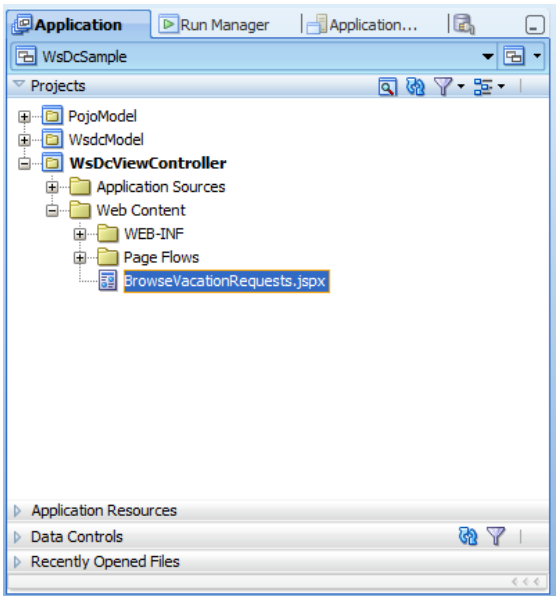
The **PojoModel** project consists of 4 entities and factory classes that provide the sample data. No database setup is required to run the example. The **SessionFacade.java** file is exposed as a JAX-WS Web Service, which basically means that annotations are used to expose the methods in this class in a Web Service.



The **WsdcModel** project contains the Web Service DataControl definition, which I created from the WSDL file reference of the deployed POJO Web Service.

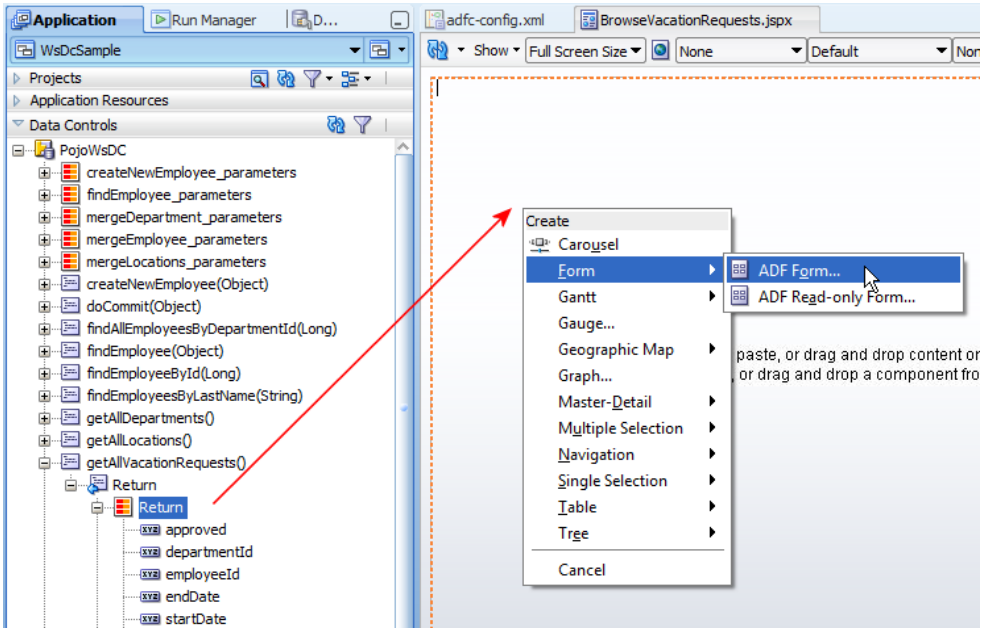


The **WsDcViewController** project uses the DataControls panel populated with the data structure of the Web Service Data Control to build vacation request form. This is the only project that you need to re-build to use this article as a hands-on instead of reading about its implementation.

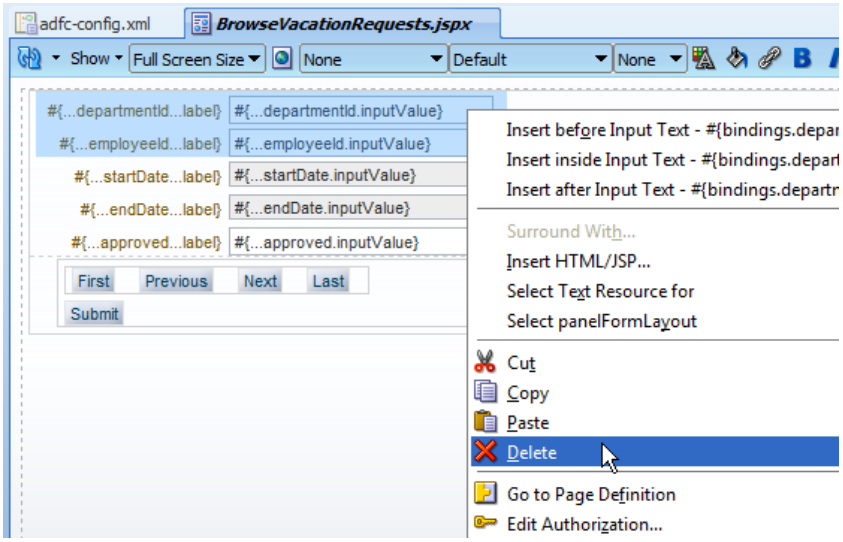


Building dependent list boxes with ADF and Web Services

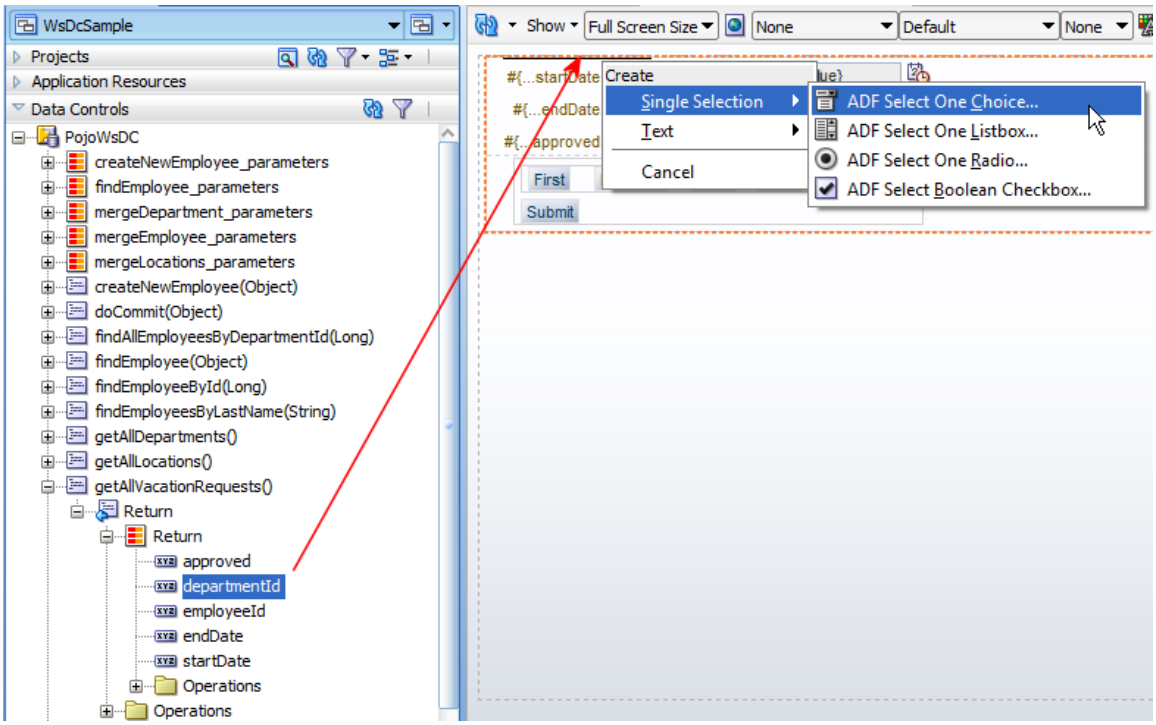
To create the input form, drag and drop the **Return** collection entry of the **getAllVacationRequest** entry as an ADF form onto the JSF page. The **Return** collection is identified by the orange icon in the Data Controls panel.



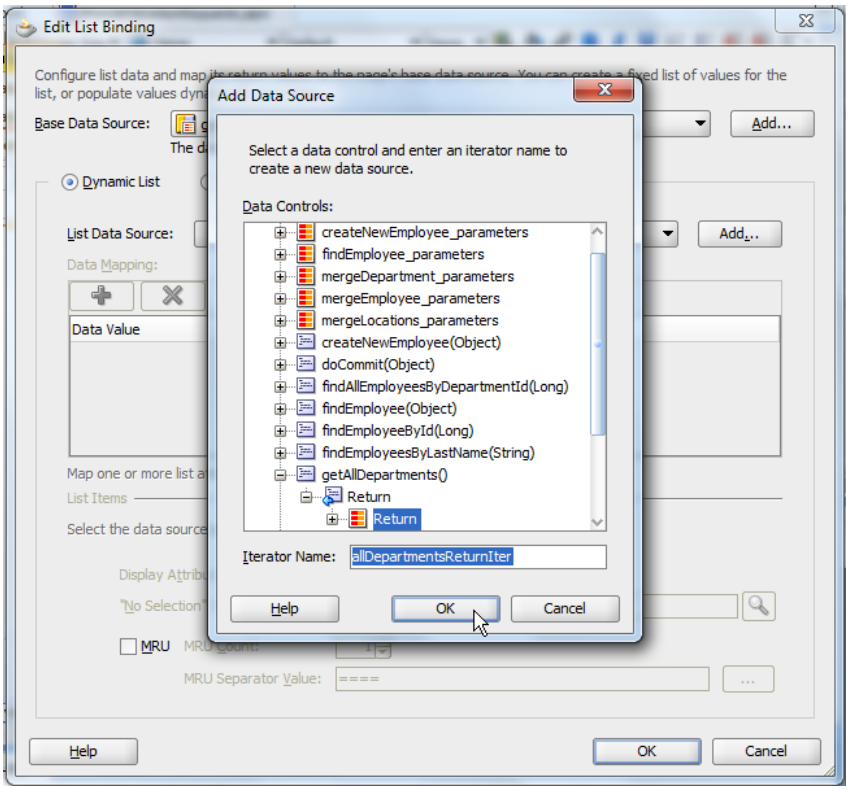
Select the **departmentId** and the **employeeId** fields and delete them by choosing **Delete** from the right mouse context menu. The two fields are re-created as lists in the following.



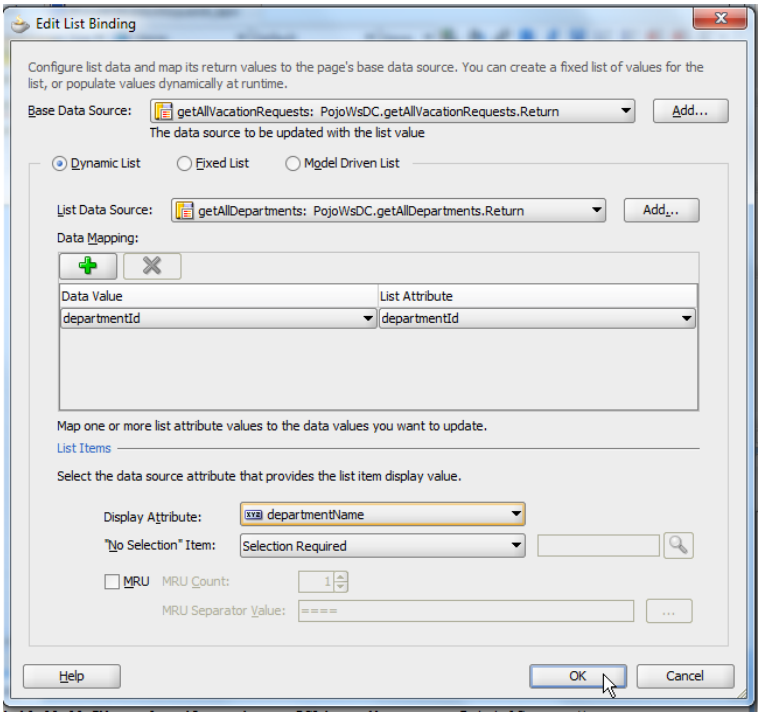
Select the **departmentsId** attribute entry under the **Return** collection node and drag it on top the form, as shown in the image below. In the opened context menu, choose **Single Selection | ADF Select One Choice**



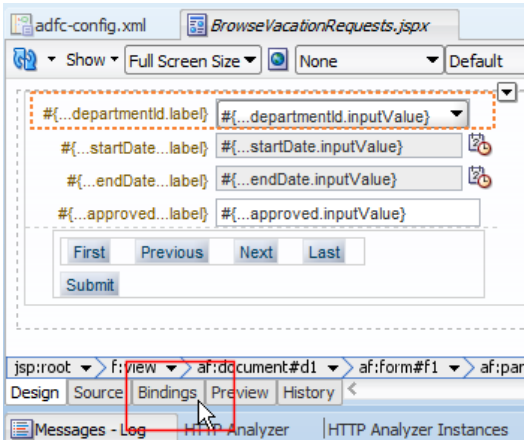
The **Base Data Source** entry points to a list binding that is created in the ADF PageDef file for the **deparetmentId** attribute. Select the **Dynamic List** option and press the **Add** button next to the **List Data Source** option. Select the **Return** collection of the **getAllDepartments** method exposed on the Data Control. Again, the **Return** collection is identified by an orange icon.



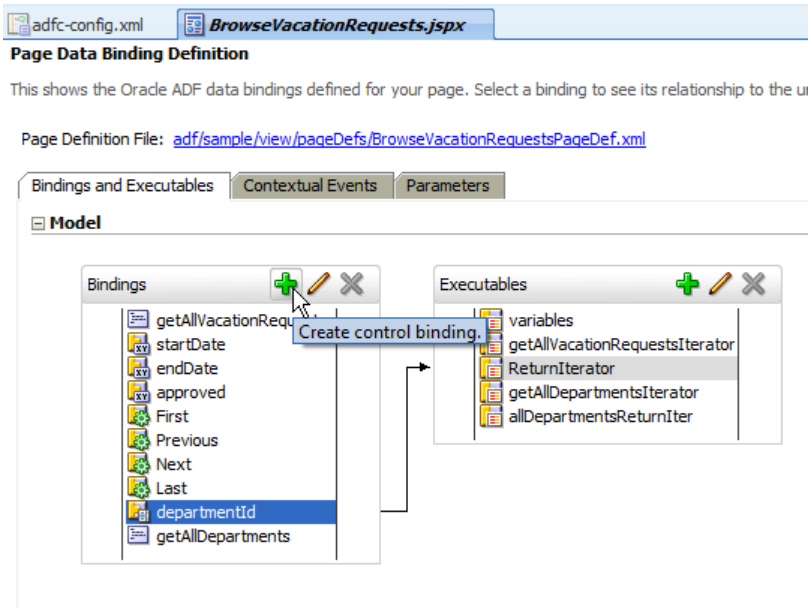
Note: Optional, but sensible, you can change the name of the iterator binding that is created from the collection to something meaningful. This helps later to identify which iterator is referenced from which UI component.



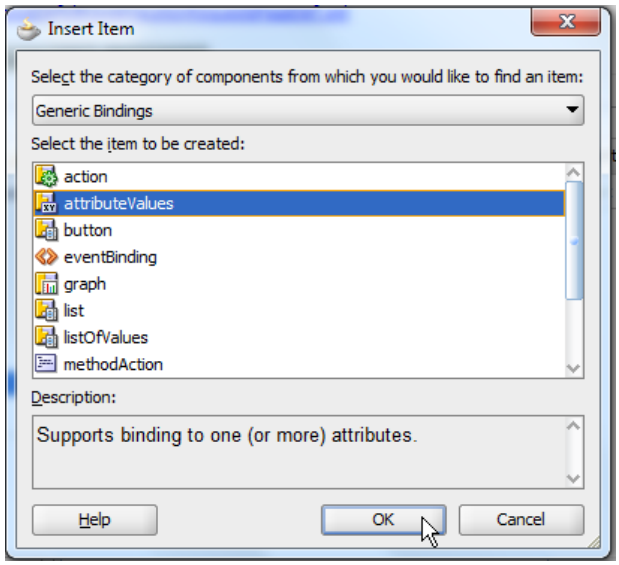
In the **Data Mapping** section of the **Edit List Binding** dialog, select **departmentId** as the attribute on both sides. To change the display value shown in the list box to be different from the ID value, expand the **Display Attribute** list and choose **departmentName.** Press **Ok** to close the dialog for JDeveloper to create and configure the new binding entry.



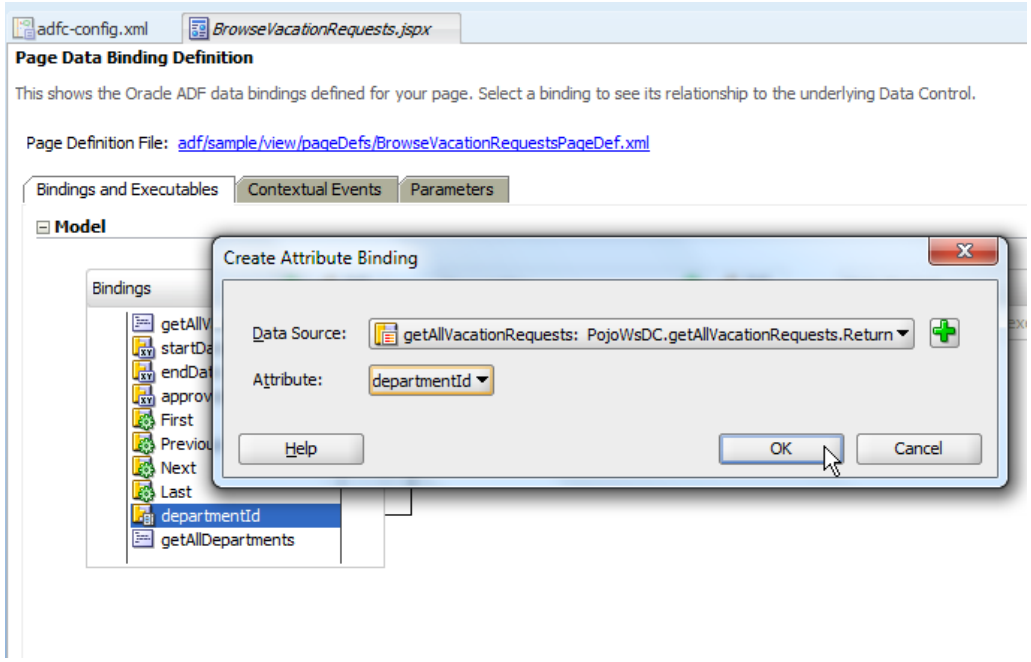
At the bottom of the visual page editor, press the **Bindings** tab to edit the PageDef file. To simplify the dependent list behavior, we create an attribute binding that provides the detail list function call with the selected department Id value.



In the **Page Data Binding Definition** editor, press the **green plus** icon.



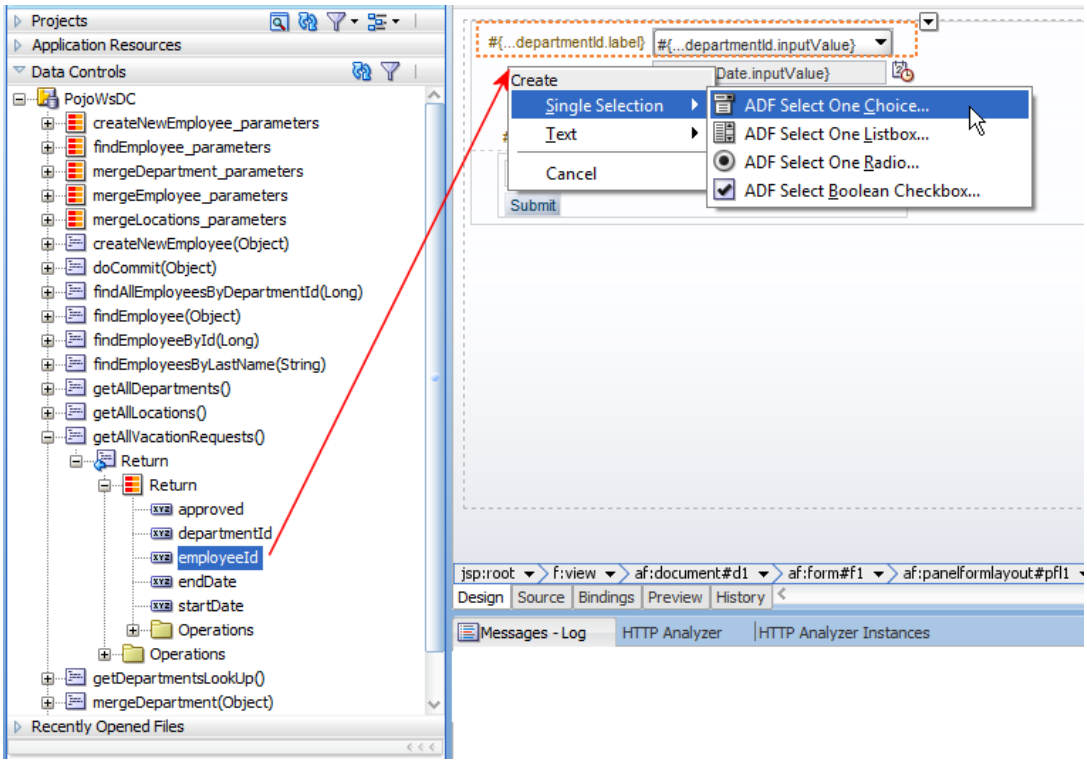
In the **Insert Item** dialog that opens, select the **attributeValues** option to create a new attribute binding.



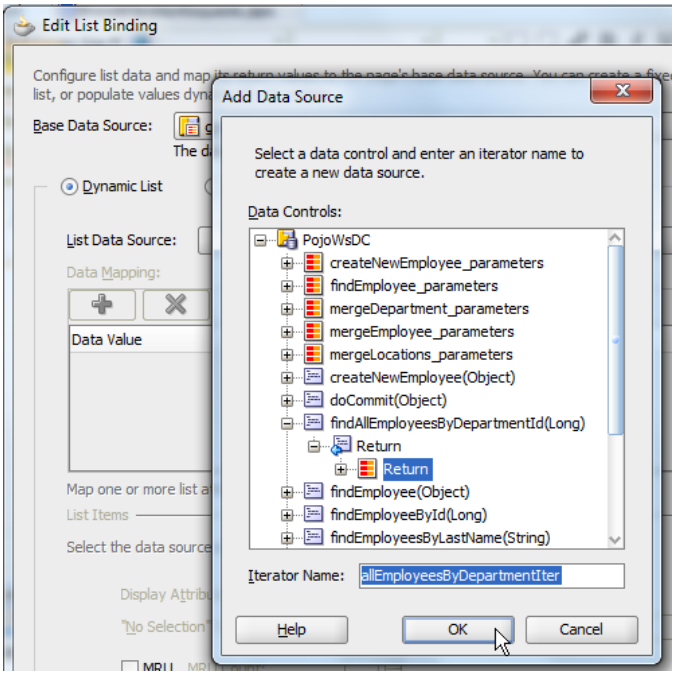
Select the **getAllVacationRequests.Return** entry as the **Data Source** and choose **departmentId** in the **Attribute** selection. The attribute binding that gets created is synchronized with the value users select in the **departmentId** list.

To create the dependent employeeId list, in the Data Controls panel, select the **employeeId** attribute under the **getAllVacationRequestst | Return | Return** collection entry.

Drag the **employeeId** attribute onto the page and drop it as **Single Selection | ADF Select One Choice** below the department Id.

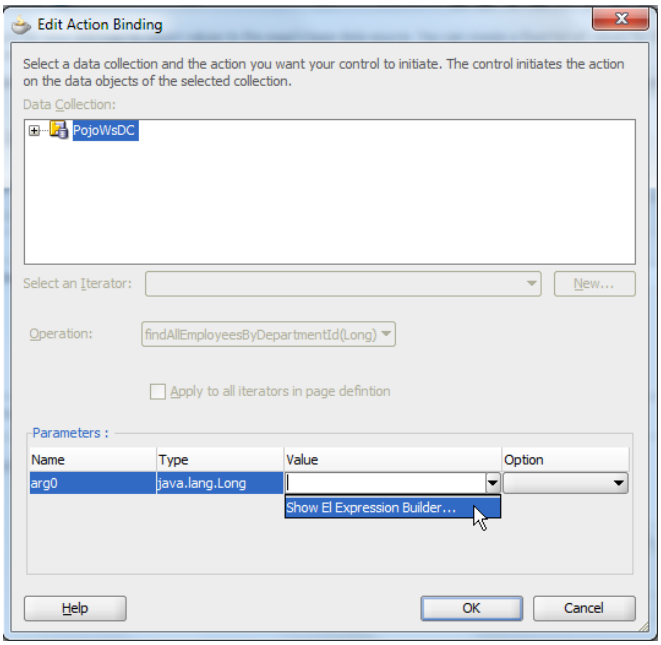


In the opened **Edit List Binding** dialog, press the **Add** button next to the **List Data Source** option and choose the **Return** collection entry of the **findAllEmployeesByDepartmentId(Long) | Return** node. The **findAllEmployeesByDepartmentId** function expect a departmentId value to be passed as an argument.

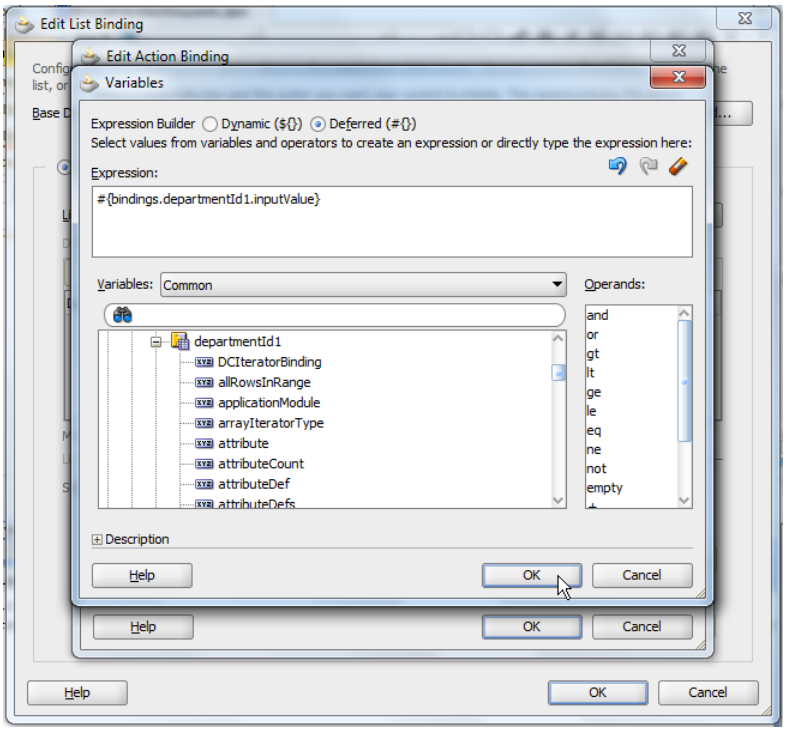


Note: Optionally, but recommended, change the name of the generated iterator binding to something meaningful to you.

In the opened **Edit Action Binding** dialog, expand the **Value** field select box and click the **Show EL Expression Builder** entry to launch the expression builder dialog.

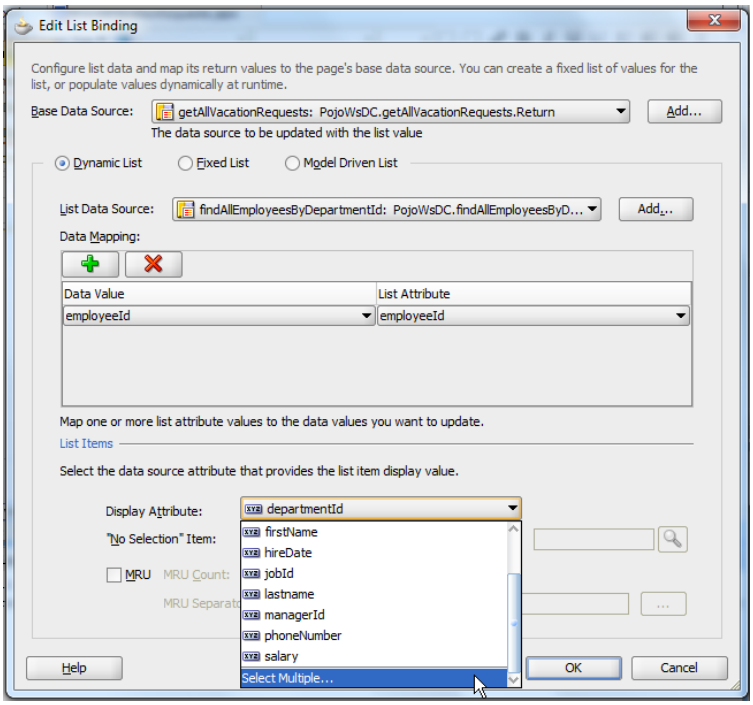


In the expression builder dialog, select the **departemtId1 | inputValue** entry to reference the selected department value.

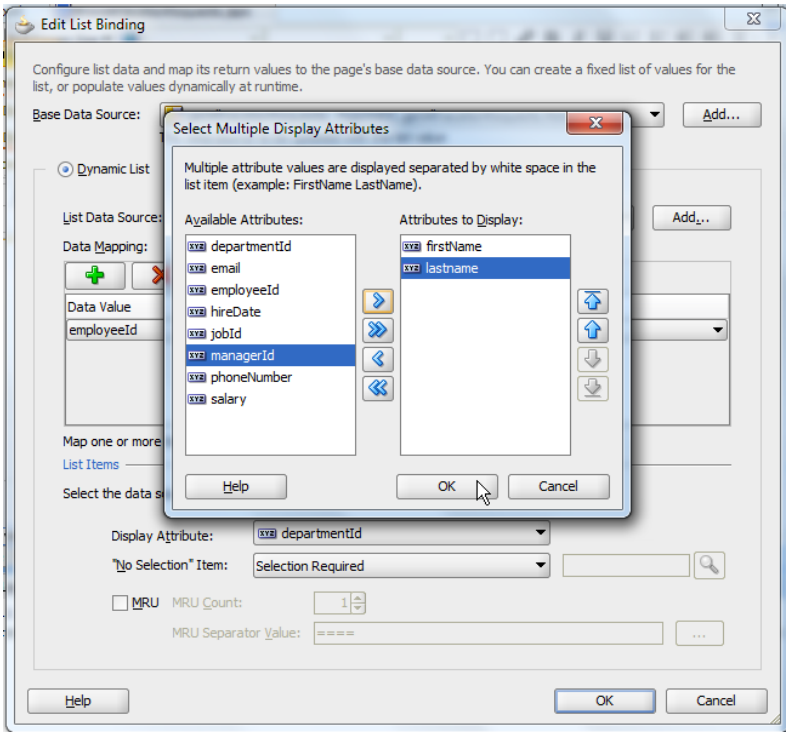


Note: departmentId**1** is the second binding entry for the department Id value. The first one is a list binding that is referenced by the selectOneChoice component.

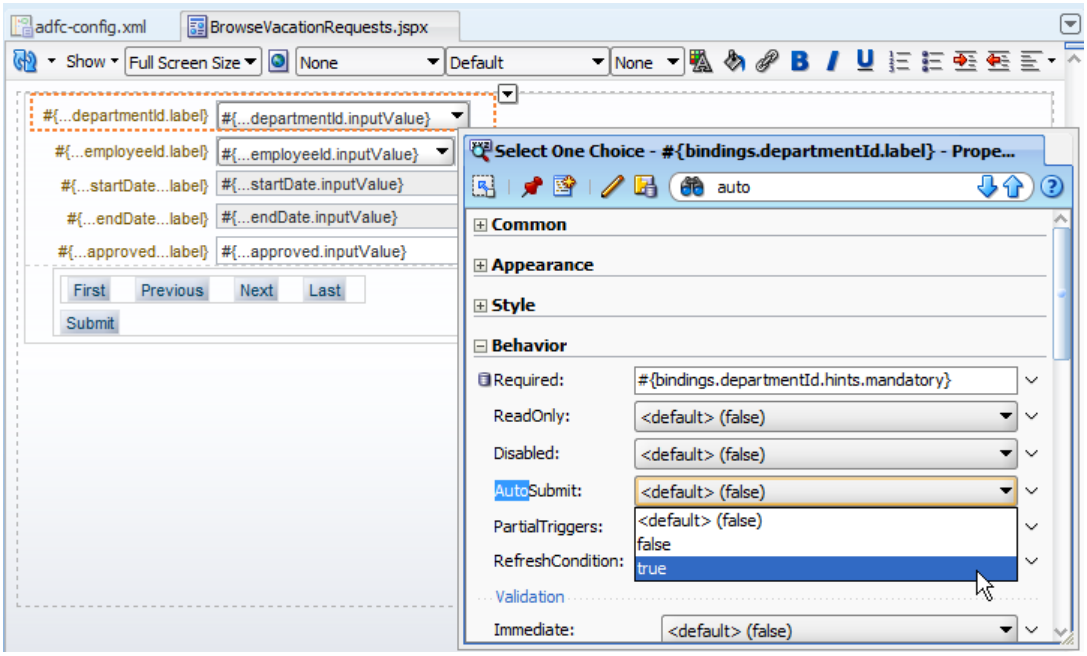
Press **OK** to close the **Variables** dialog and to return to the **Edit List Binding**dialog.



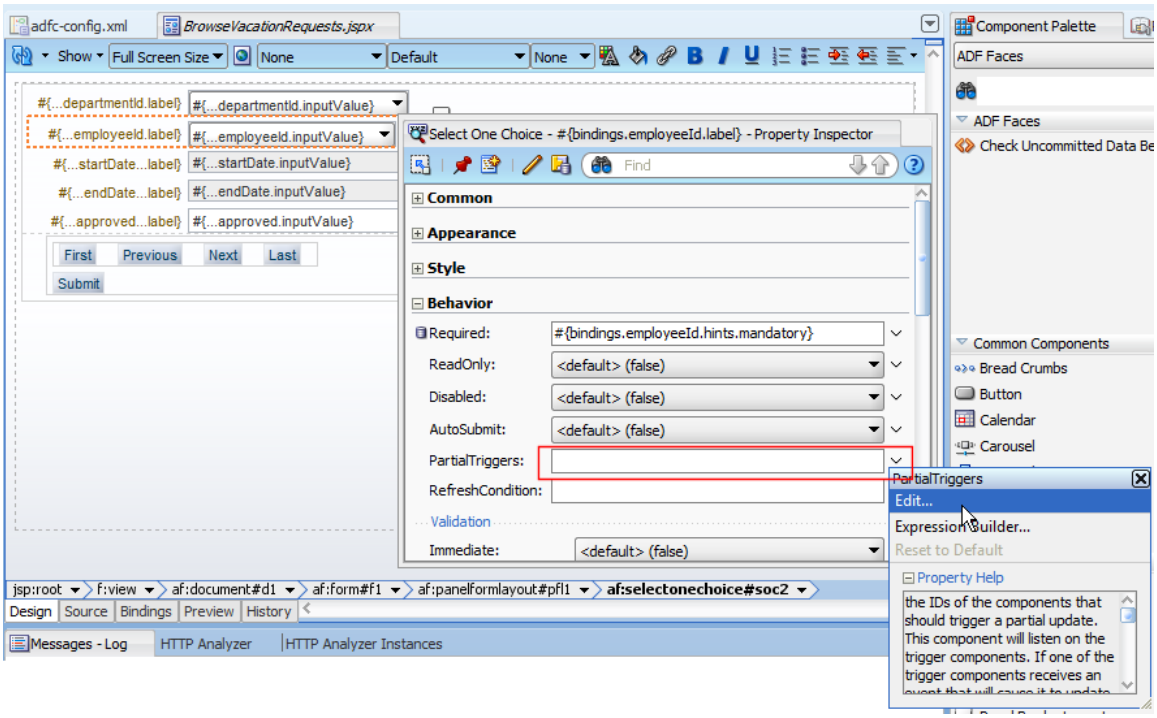
In the **Edit List Binding**dialog, expand the **Display Attribute** list and click onto the **Select Multiple ...** entry to define the list display as "firstName lastname".



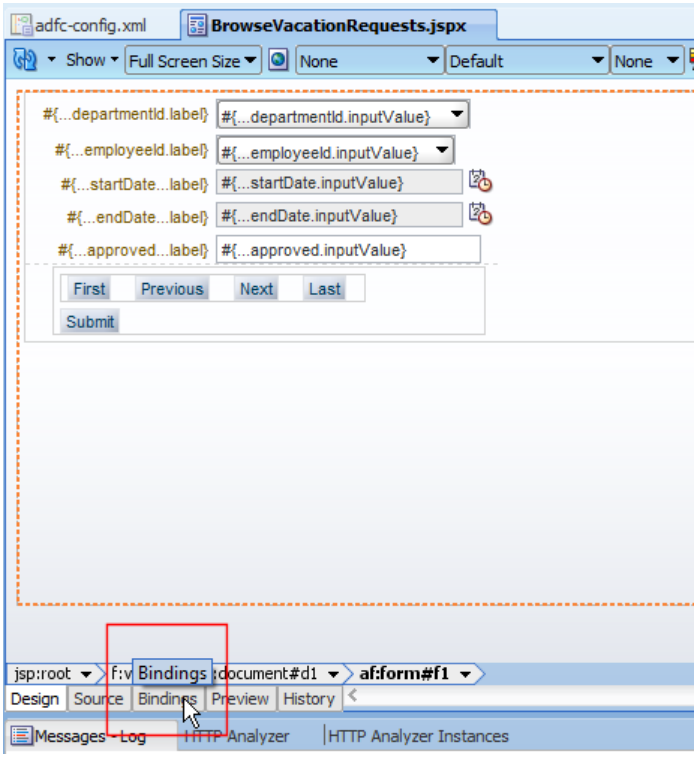
Press **OK** on both opened dialogs to close them and return to the visual page editor.



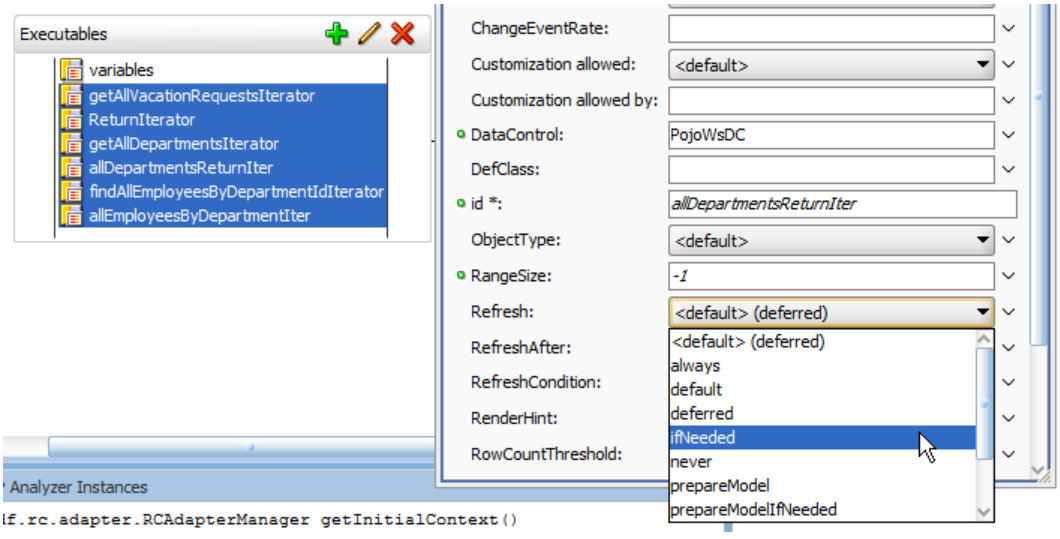
In the visual page editor, select the **departmentId** select one choice component and set its **AutoSubmit** property to **true**. This way, selecting a new department automatically issues a partial submit request.



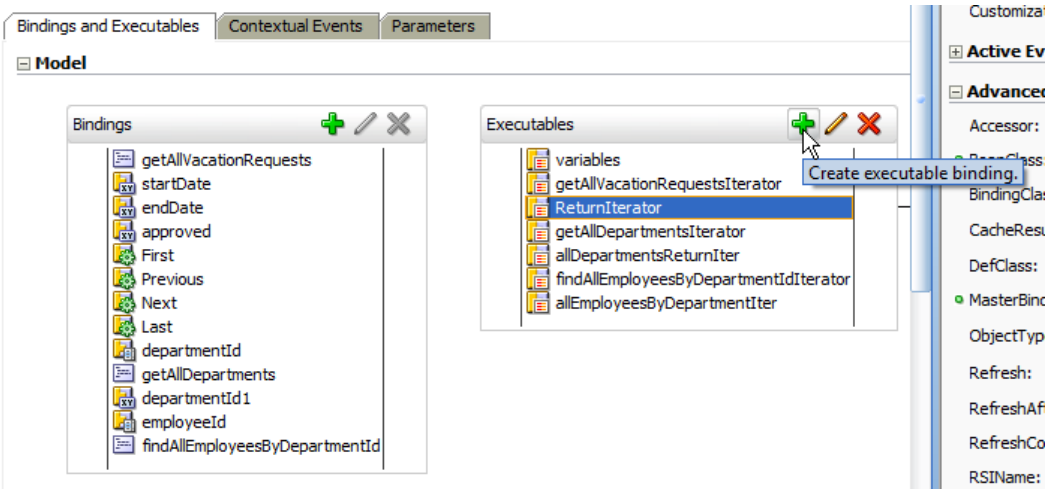
Select the **employeeId** select one choice component and click the arrow icon next to its **PartialTriggers** property. In the opened editor, lookup the departmentId select one choice component and select it. This way, the employeeId select one choice component is refreshed when the departmentId field changes.



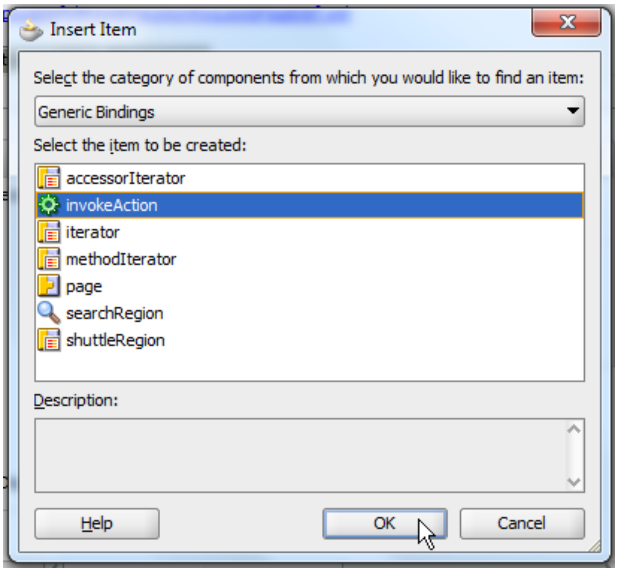
Click onto the **Bindings** tab in the visual page editor to see the Binding editor.



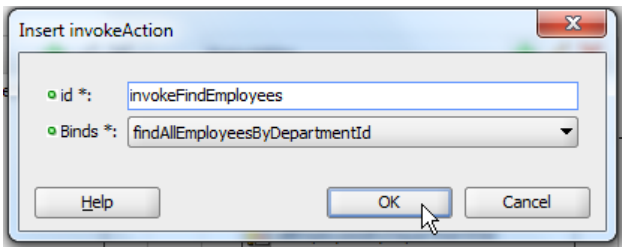
Select all iterators and change their **Refresh** setting from **deferred** to **ifNeeded**. It changes the refresh behavior to refresh during prepare model and – if needed prepare render phase.



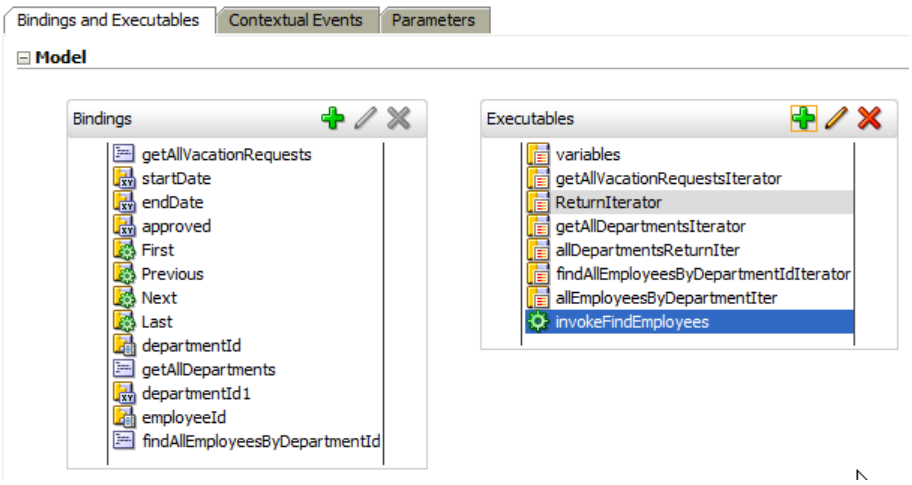
To initially execute the dependent employee method with the current selected **departmentId**, you need to define an **invokeAction** binding in the **Executables** section of the binding.



Point the **invokeAction** to the **findEmployeesByDepartmentId** method binding and provide it a unique ID string



This is it and the dependent list boxes start working.



About Web Services and Oracle ADF

Oracle ADF does not cache data queried from a Web Service across PageDef (binding container) uses. Therefore, for heavy data queries and uses you want to consider a custom caching strategy involving the Web Service client proxy and the ADF POJO Data Control. However, if it is all about accessing Web Services method for a reasonable amount of data, using the Web Services Data Control is an easy and probably better choice.

Sample Download

The sample is developed with Oracle JDeveloper 11.1.1.4 but should work with JDeveloper 11.1.1.3 as well. You can download the zip file as sample #70 from the ADF Code Corner website.

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

No database connection is required as the Web Service reads its data from a POJO containing a copy of the HR schema. The web service endpoint is configured for localhost and need to be changed if you deploy to a remote – non local – server.

RELATED DOCOMENTATION

☒	
☒	
☒	