ADF Code Corner

107. How-to enforce LOV Query Filtering

twitter.com/adfcodecorner

Abstract:

A question on OTN was about how-to restrict queries in a LOV dialog to avoid unfiltered and expensive queries. For this at least one of the LOV query fields must be provided with a value.

This article explains how the above requirement can be implemented declaratively with the help of a view criteria.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
20-MAR-2013

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

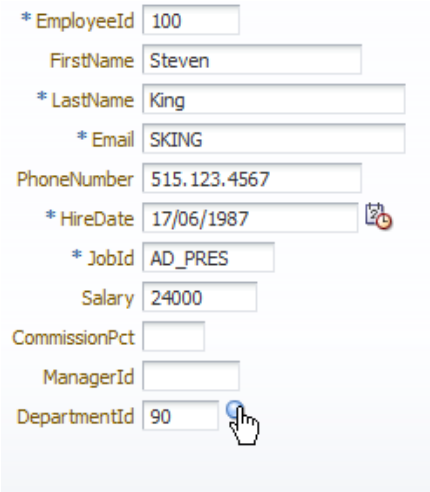
Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

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

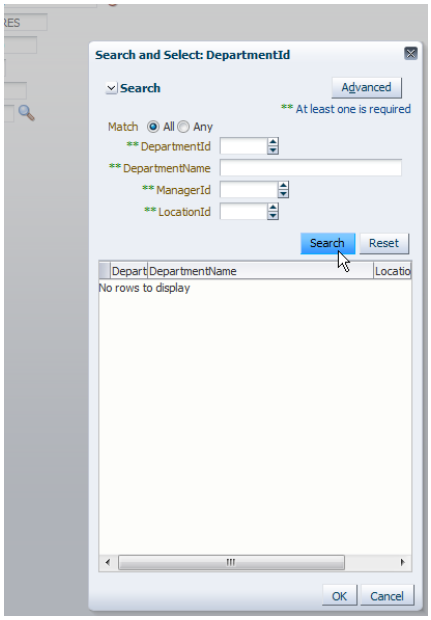
The images below show a LOV dialog that requires users to at least specify search conditions for at least a single required search field.

At runtime, using a LOV input text component, users click the LOV icon to launch the list dialog

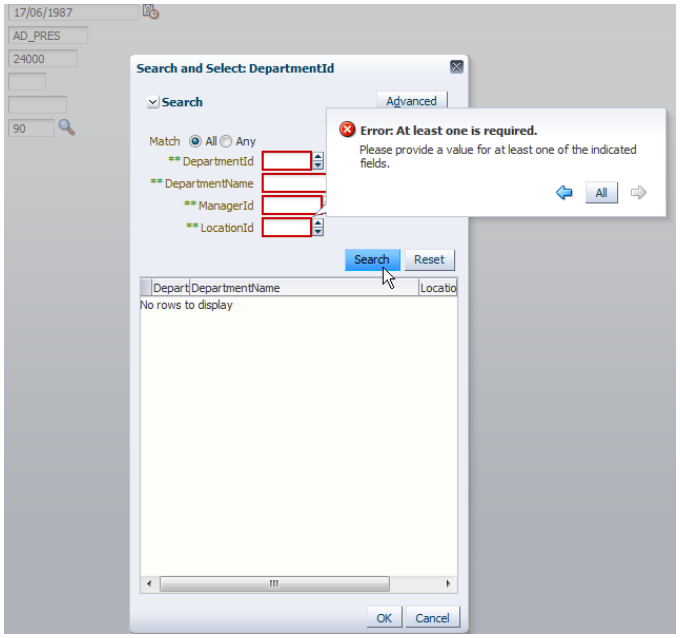


The LOV dialog opens with a search field that indicates search fields as selectively required (see image below). That is that at least one of these fields must be provided with a seach condition.

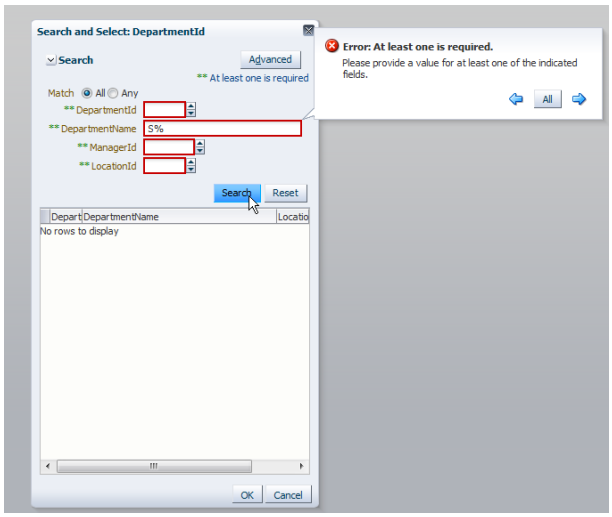
Note: You can use the same declarative approach explained in this article to also allow optional search fields, e.g. only flagging two attributes as selectively requied.



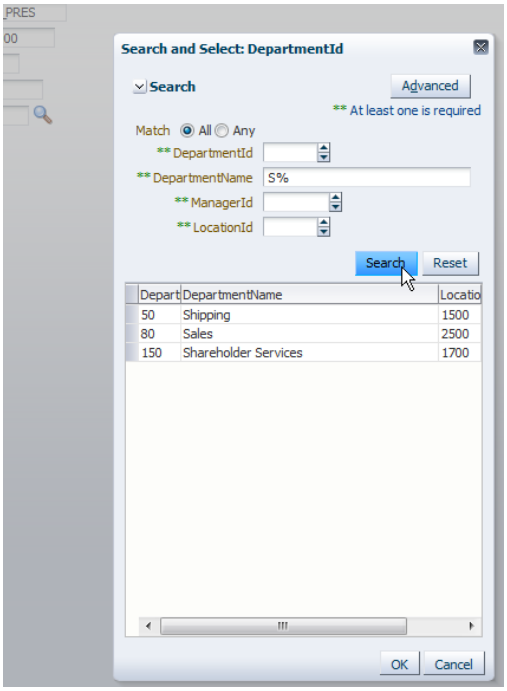
Pressing the **Search** button without providing any search criteria will cause an error to be displayed as shown in the image below.



Correcting the problem and adding a search condition for at least one search field does correct the problem (see image below)



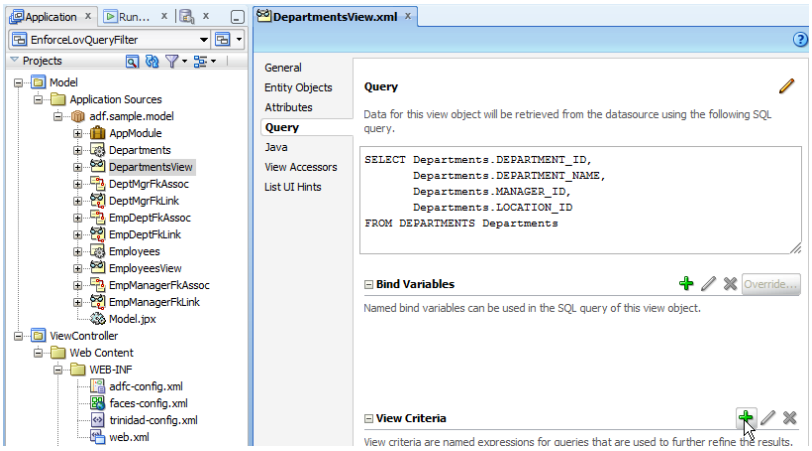
If one of the selective search fields has a value provided, the search will produce a result.



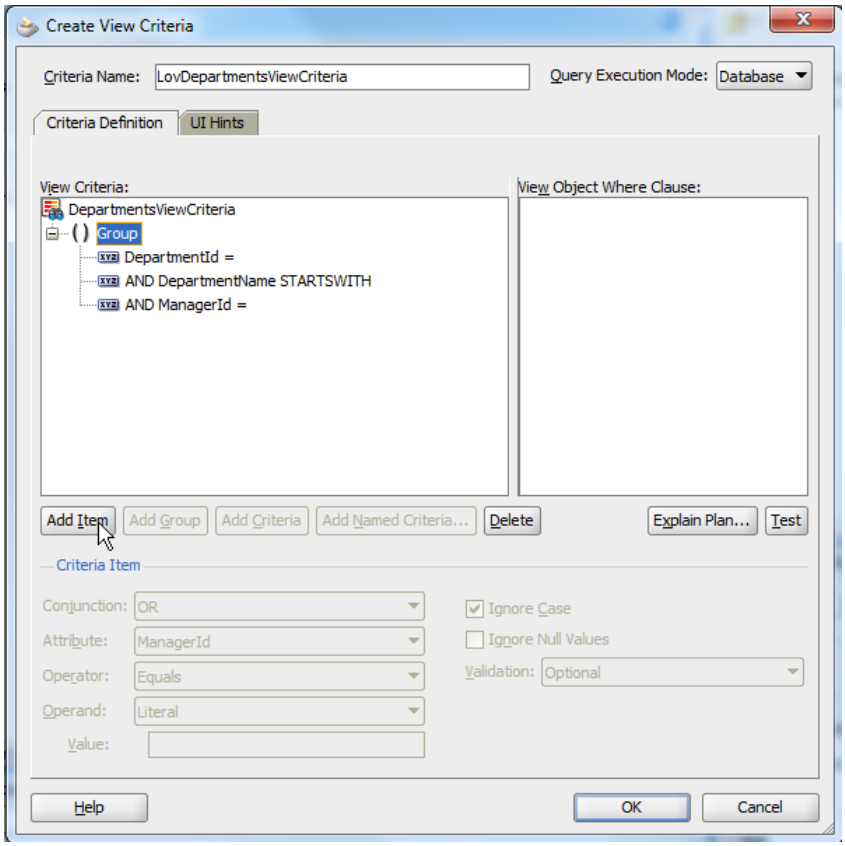
Let's see how to implement this solution declarative with no coding required.

Creating a View Criteria

The recipe for solving this problem is in that List-of-values can be restricted in their query by view criteria and that view criteria can define their query attributes as selectively required. The sample application uses a LOV based on the ADF Business Components **DepartmentsView** object shown in the image below.



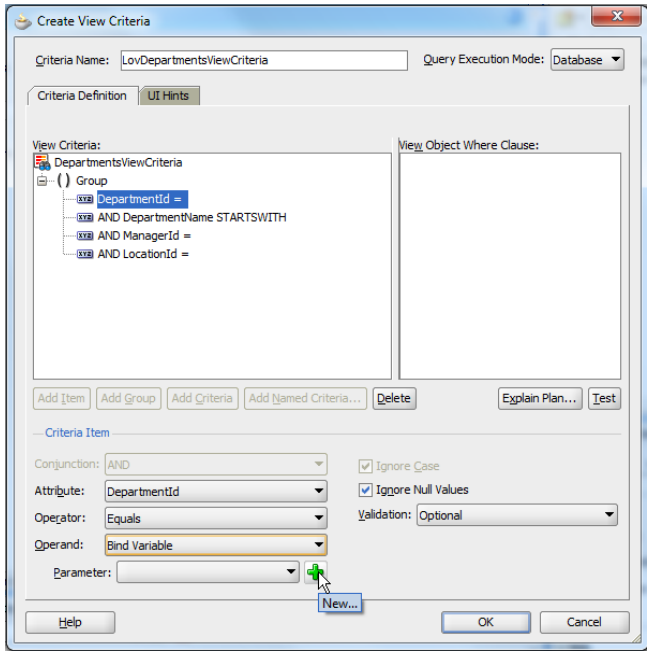
To create the View Criteria, open the DepartmentsView object (or the view object that holds your list of values) and select the **Query** option as shown in the image above. In the **ViewCriteria** section, click the **green plus icon** to launch the View Criteria creation dialog.



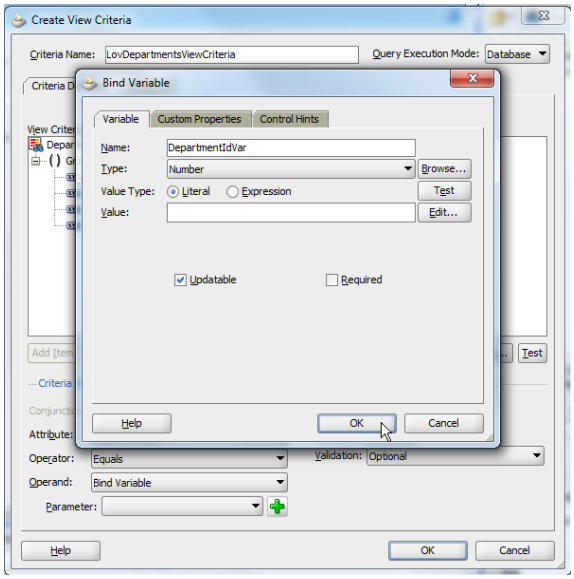
Name the view criteria, like *LovDepartmentsViewCriteria* in the above image, and then, with the **Group** node selected, press the **Add Item** button until all of the attributes you want to be displayed in the LOV query regions are added. Note that associations may also shows, in which case you use the **Delete** button unless you want to filter the LOV by dependent view objects too.

Next (shown in the image below) you create a bind variable for each of the attributes in the view criteria. For this, select the view criteria and choose the **Operand** list to show **Bind Variable**. Then press the

green plus icon (as shown in the image below) to create new bind variables to hold the query values at runtime.

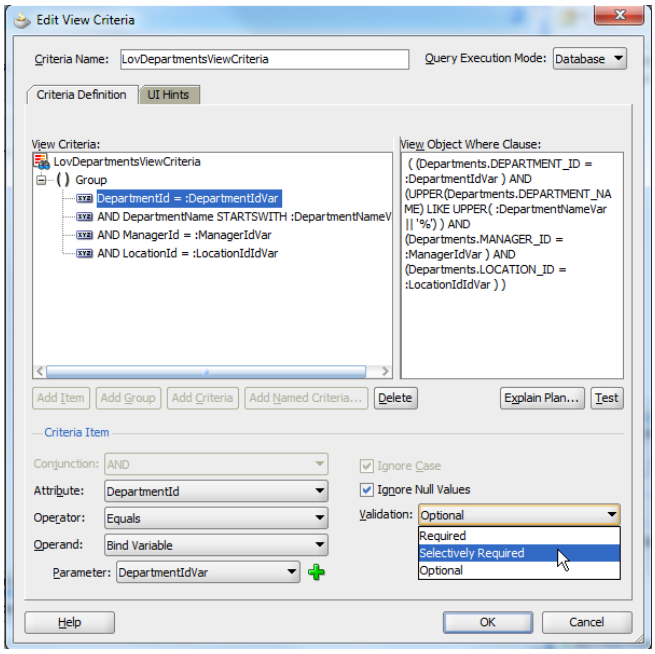


Note: Make sure you create a Bind variable for each attribute



Define a unique name for the bind variables and make sure the **Type** field matches the type of the attribute.

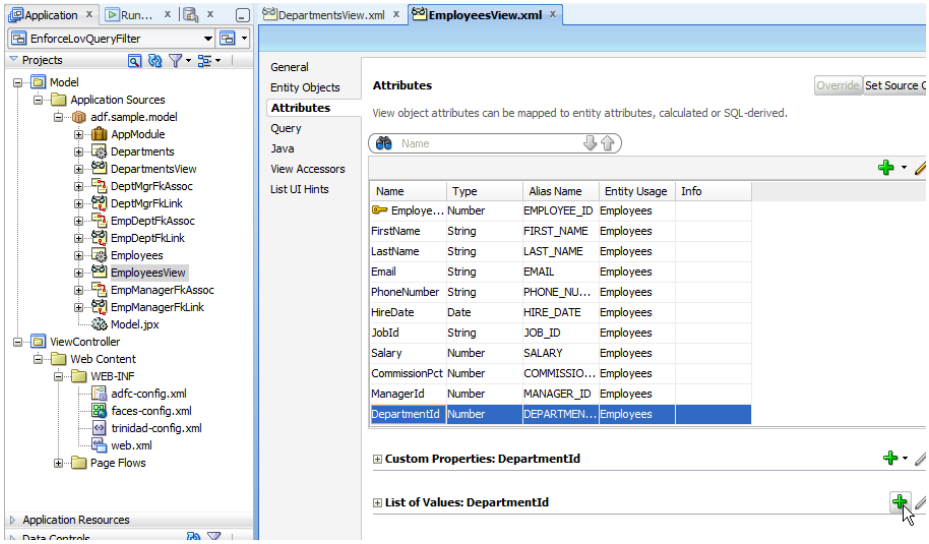
Next, select each attribute in the view criteria and change the **Validation** option to **Selective Required** for all attributes that should have a value unless one of the other selectively required attributes got one assigned. In the sample I set this for all attributes. In your implementation you can only have a few (at least two must be marked as selectively required though)



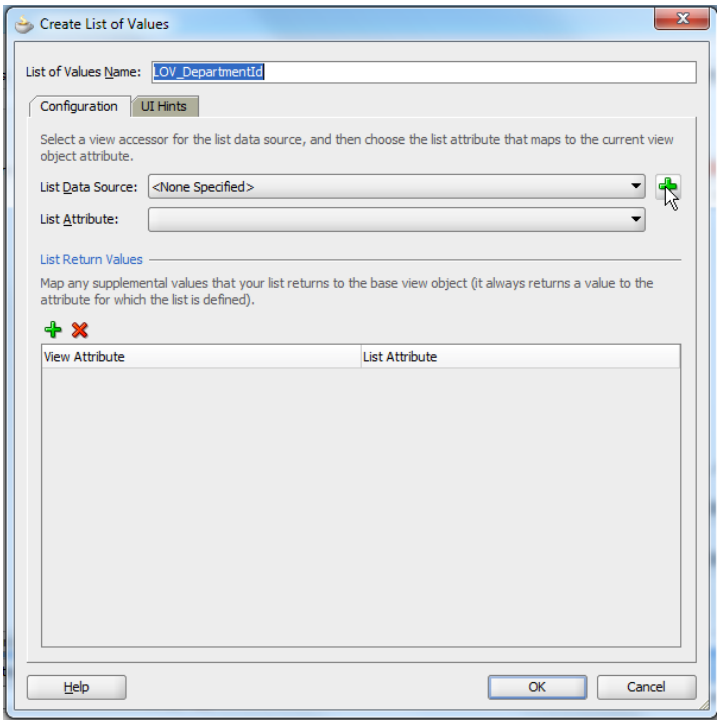
Press **Ok** to close the dialog with the view criteria created.

Building the Model Driven LOV

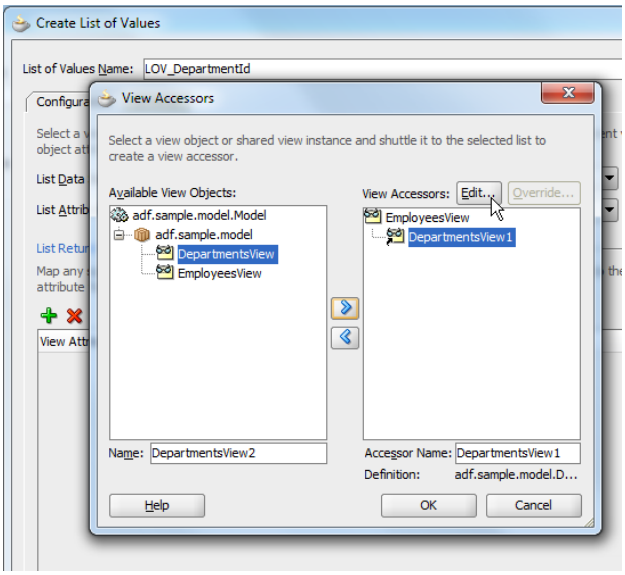
Next you need to build the list-of-values for the attribute that should provide the option. In the image below, this attribute is the **DepartmentId.**



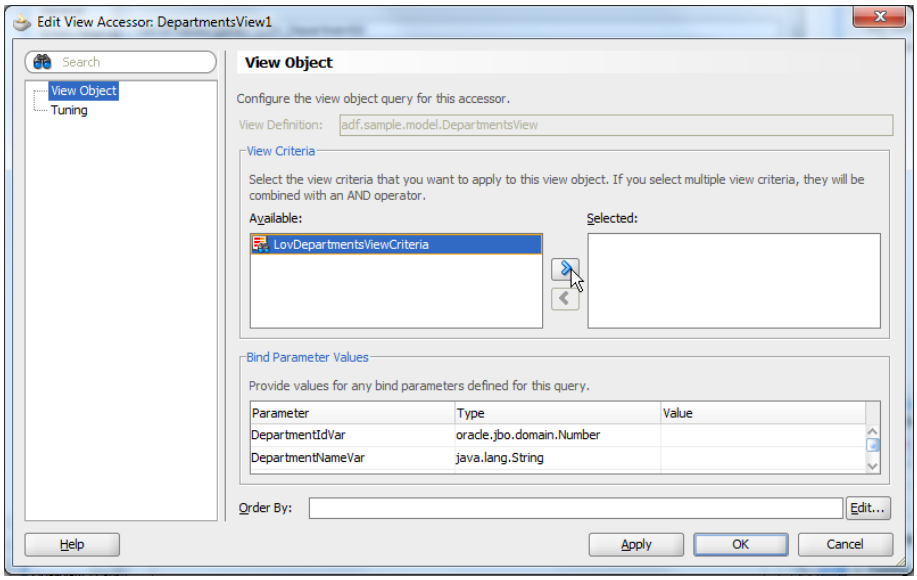
As shown in the image above, press the **green plus icon** next to the **List of Values <attribute name>** header to bring up the List of Values creation dialog.



In the LOV creation dialog, press the green plus icon (shown in the image above) and select the DepartmentsView object (your list view object in your implementation). Move the view object to the list of **View Accessors** as shown in the image below, select it and press the **Edit** button to apply the view criteria created earlier.

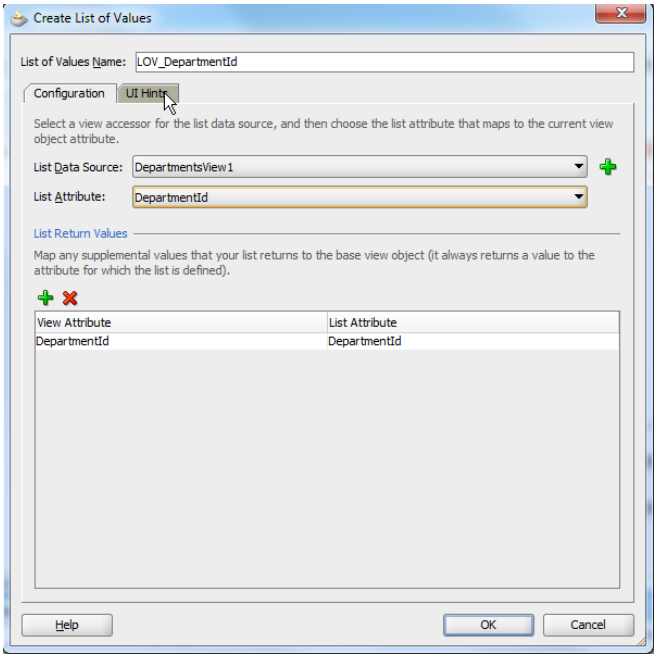


Select the view criteria and shuttle it to the **Selected** list. You don't need to provide values for the bind variables as this is what the LOB search form will add.



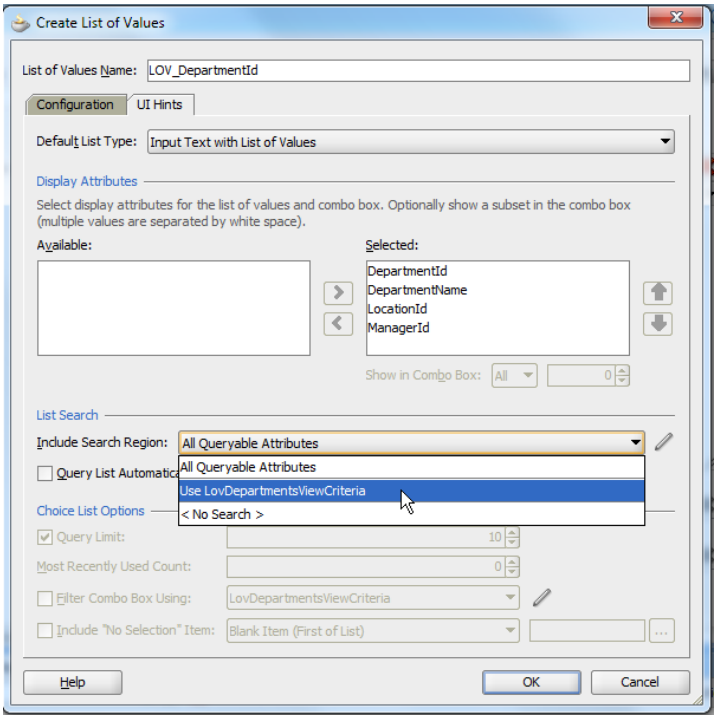
Press **Ok** to close this dialog.

Once you configured the attributes as shown in the image below, press the **UI Hints** tab.



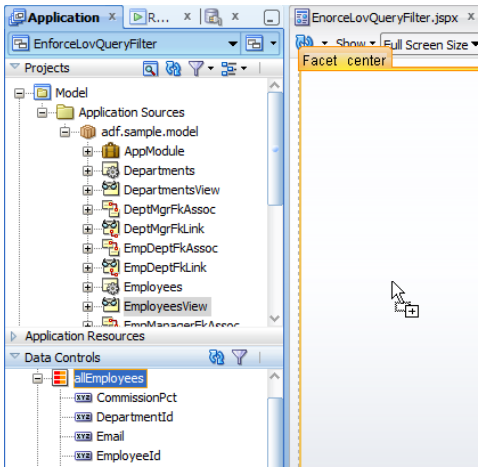
In the **UI Hints** tab pane, select all attributes that should be displayed in the list-of-values result table (in the image below, I selected all attributes to show)

Set the **Default List Type** to **Input Text with List of Values** and choose the name of the view criteria – **LovDepartmentsViewCriteria** in the sample – for the **Include Search Region** field.

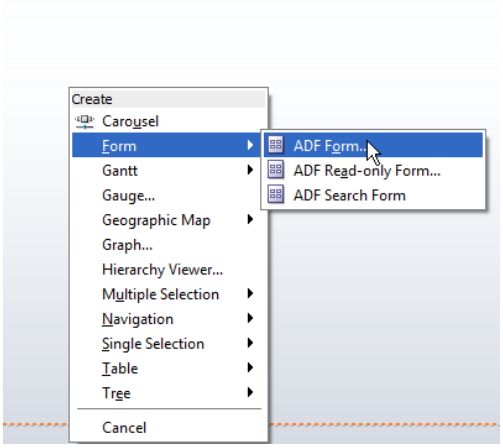


Press **Ok** to close the dialog. This basically is all you needed to do for implementing the use case.

To build the user interface, just drag and drop the **EmployeesView** object (the view object that contains the LOV attribute) to the page and drop it …

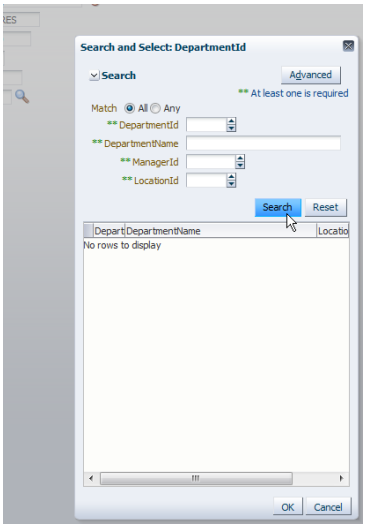


… as an **ADF Form** as shown in the image below.



Press **Ok** to create the form and run the page.

At runtime you should the UI shown below and the behavior shown in the images at the start of this article.



Download

You can download a JDeveloper 11.1.1.6 sample workspace as sample 107 from the ADF Code Corner website: http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html

You need to configure the database connection to point to a local database with the Oracle HR schema unlocked. Then run the JSPX file.

RELATED DOCOMENTATION
