

## ADF Code Corner

### 069. how-to create a custom LOV using bounded task flows

**ORACLE**  
**CODE CORNER**



[twitter.com/adfcodecorner](https://twitter.com/adfcodecorner)

#### **Abstract:**

Model driven list of values in ADF Business Components allow developers to build searchable select lists that open in a lightweight DHTML dialog. However, what if the list of value dialog is a multi step proces or needs a special layout or functionality? In this blog article I show how to create a custom list of value using a bounded task flow that opens in a dialog using the external dialog framework. This article also shows how to globally remove the the close icon from those list fo value dialogs so users must exit the dialog using the command provided in the dialog.

Author:

Frank Nimphius, Oracle Corporation  
[twitter.com/fnimphiu](https://twitter.com/fnimphiu)  
18-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

Recent builds of Oracle JDeveloper 11g provide the option to declaratively configure bounded task flows that use JSPX documents for their views to open in a lightweight dialog using the ADF Faces external dialog framework. This feature allows developers to create custom list of values that may run as multi step processes. The user selected value is passed back to the calling view in a bounded or unbounded task flow to update and refresh an input field. This ADF Code Corner article shows you

- How to create a bounded task flow driven list of value
- How to suppress the default close dialog
- How to refresh an input field if the dialog is opened by a command that has its immediate property set to true
- How to create a custom version of the default "All Queriable Attribute" View Criteria that searches in committed and uncommitted data and also executes its query when the LOV launches

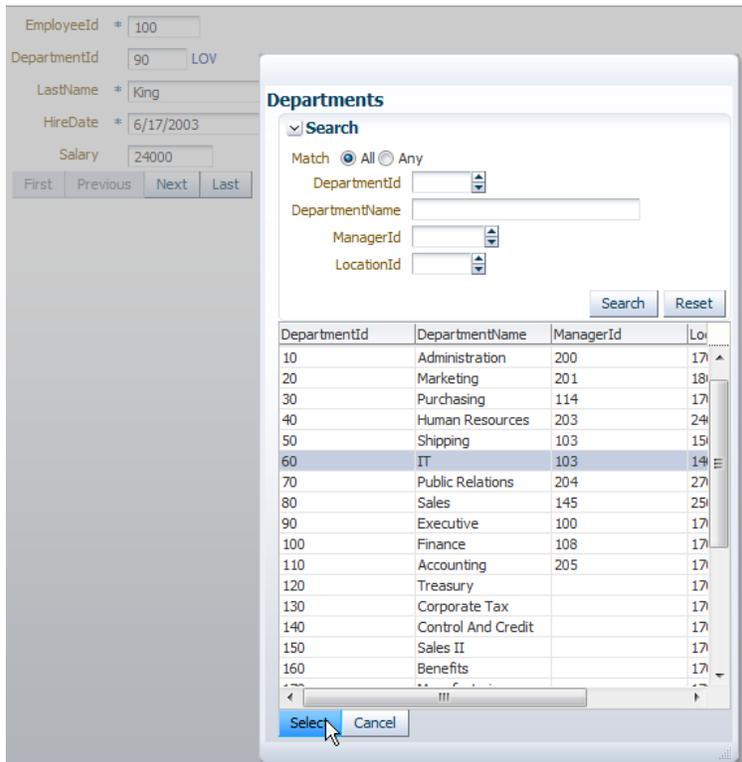
In the example built for this blog article, the Employee form shows a LOV command link next to the **DepartmentId** field. Clicking onto the link opens a dialog displaying a list of departments to choose from.



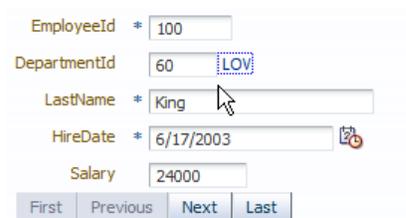
The screenshot shows a form with the following fields and controls:

- EmployeeId: \* 100
- DepartmentId: 90 [LOV](#) (with a mouse cursor pointing to the link)
- LastName: \* King
- HireDate: \* 6/17/2003
- Salary: 24000
- Navigation buttons: First, Previous, Next, Last

The LOV dialog content is defined in a bounded task flow and consists of an af:query form that queries a result table for the user to select a department. Selecting a table row and pressing the **Select** button navigates to the bounded task flow return activity, which returns the **DepartmentId** to the calling page. Though the sample only uses a single view, you can imagine that there is no limit for the number of views you can display in this LOV.



The update of the employee form is handled in the command link that may run as multi step processes. The user selected value is passed back to the calling view **ReturnListener** handler.

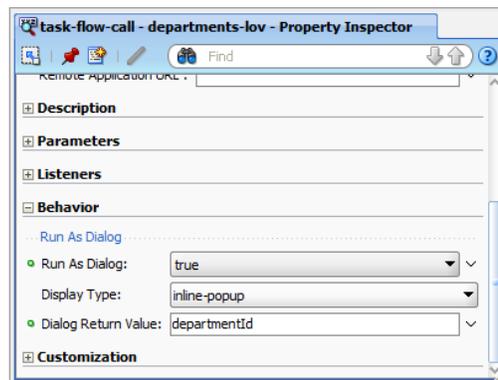
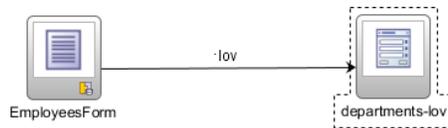


## How-to build LOV from bounded task-flows

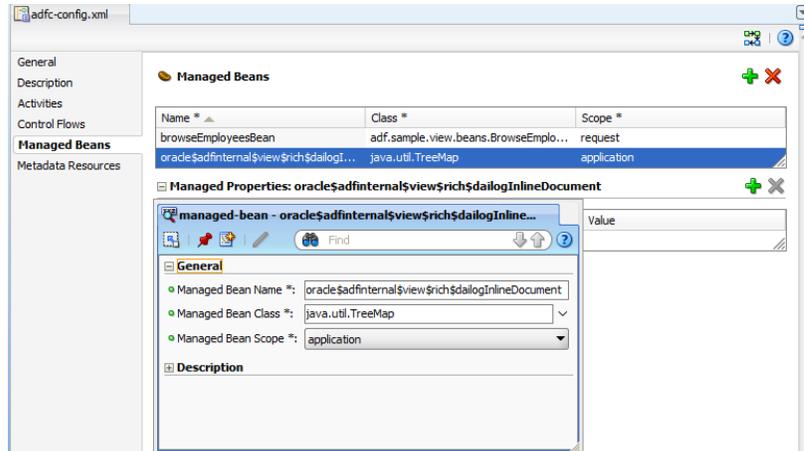
Bounded task flows that are exposed in lightweight dialogs must use JSPX documents for rendering the views. The dialog configuration is defined on the task flow call activity that is created when dragging the

bounded task flow into the calling task flow diagram. The control flow case – **lov** in the sample – is called from a command link and points to the task flow call activity.

Unbounded Task Flow



On task flow return, the dialog is closed and the return listener on the "LOV" link component is called for the developer to read the returned data and update the employee **DepartmentId** field. Note that the **Dialog Return Value** references the bounded task flow return parameter that should be delivered within the return value listener defined on the command link launching the dialog.



By default the LOV dialog has a close icon in the header for users to close the dialog. However, the close icon does not invoke the return listener, which is why developers want to hide it. To hide the close icon, a managed bean configuration in the `adfc-config.xml` file can be used as shown below.

```
<managed-bean>
  <managed-bean-name>
    oracle$ad$internal$view$rich$dialo$inlineDocument
  </managed-bean-name>
  <managed-bean-class>java.util.TreeMap</managed-bean-class>
  <managed-bean-scope>application</managed-bean-scope>
</managed-bean>
```

```

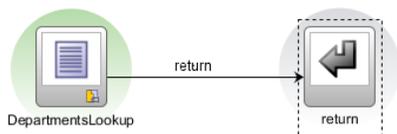
<map-entries>
  <key-class>java.lang.String</key-class>
  <value-class>java.lang.String</value-class>
  <map-entry>
    <key>MODE</key>
    <value>withoutCancel</value>
  </map-entry>
</map-entries>
</managed-bean>

```

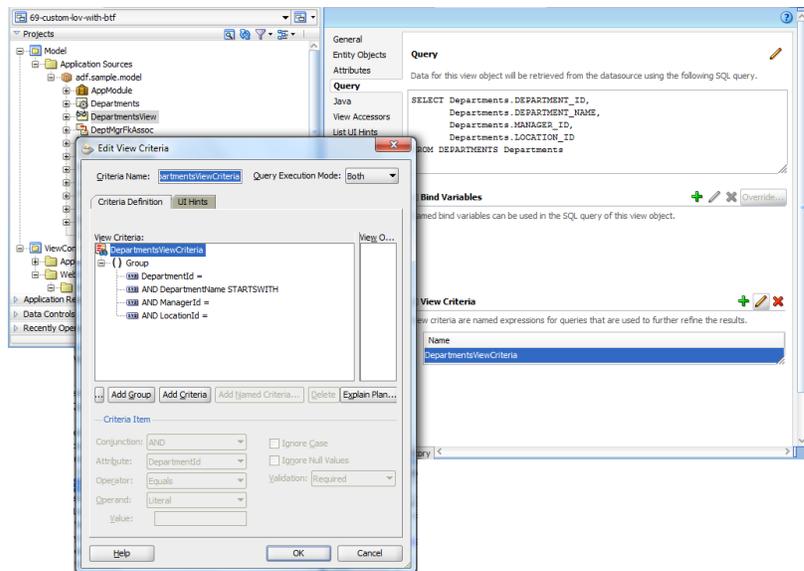
**Note:** The managed bean name has a misspelling "**dailog**InlineDocument", which Oracle is aware of. This is not a type in this how-to document.

**Note:** The above configuration is an Oracle JDeveloper 11.1.1.4 feature

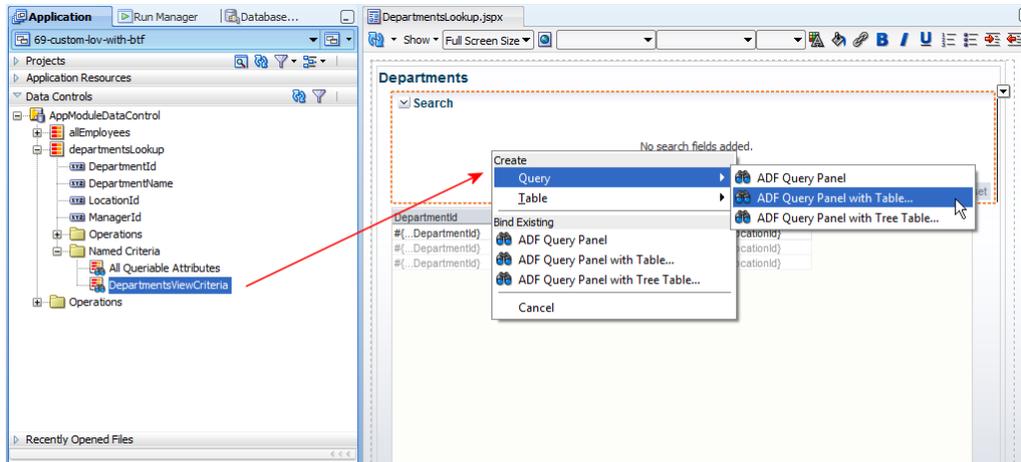
The image below shows the content of the bounded task flow that is exposed in the LOV dialog. The bounded task flow default activity is a view displaying the search form and result table. Selecting a table row and pressing the **Select** button stores the department id of the selected row to a page flow scope attribute that is referenced from the bounded task flow output parameters.



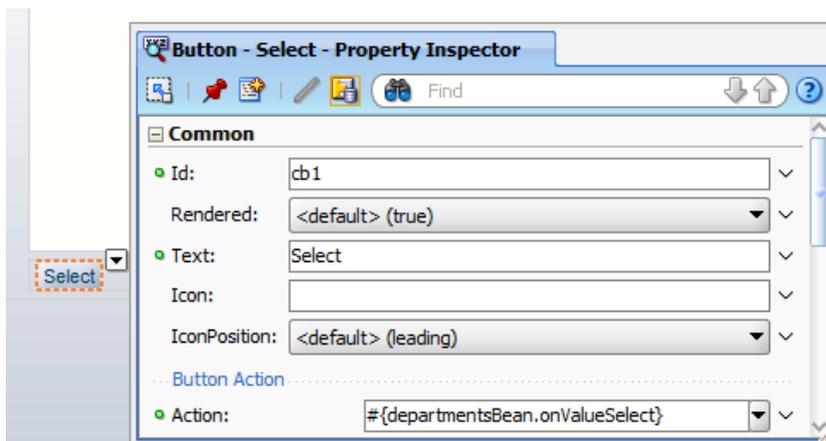
The query form is built from a View Criteria in ADF Business Components. In the provided example, the LOV is immediately executed when the LOV dialog opens. For this the query form is built from a custom View Criteria definition shown below.



In the **UI Hints** tab the View Criteria is configured to execute the query when the LOV opens and to read the data from memory and the database, which also is not what the default View Criteria does.



To create the LOV query form, the custom View Criteria is dragged to the dialog page and dropped as an **ADF Query Panel with Table**. To close the dialog, two buttons are added below the table. Both buttons navigate to the bounded task flow return activity.



The managed bean method referenced from the **Select** button is shown below. It accesses the result table **DCIteratorBinding** to read the selected row's **DepartmentId** value.

The value is written to the page flow memory scope where it is looked up from the bounded task flow **Return Value Definition**.

```
public class DepartmentsLovBean {
    private RichTable employeeTable;

    public DepartmentsLovBean() {
        super();
    }

    public String onValueSelect() {
        //get collection model from table JSF component binding reference.
        //This also grants access to the DCIteratorBinding that is
        //synchronized with the selected table row
        CollectionModel model = (CollectionModel) employeeTable.getValue();
        JUCtrlHierBinding tableBinding =
            (JUCtrlHierBinding) model.getWrappedData();
```

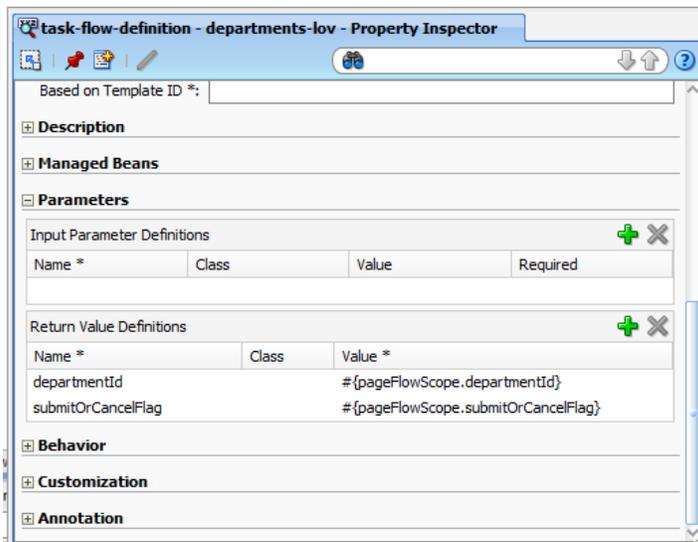
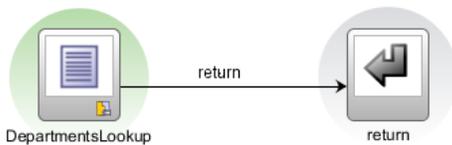
```

//table synchronizes row selection with current binding row
DCIteratorBinding tableIterator
    tableBinding.getDCIteratorBinding();
if (tableIterator.getCurrentRow() != null) {
    Object departmentIdValue =
        tableIterator.getCurrentRow().getAttribute("DepartmentId");
    //copy value into the pageFlowScope, which is returned in an task
    //flow param output
    ADFContext adfCtx = ADFContext.getCurrent();
    Map pageFlowScope = adfCtx.getPageFlowScope();
    pageFlowScope.put("departmentId", departmentIdValue);
}
return "return";
}

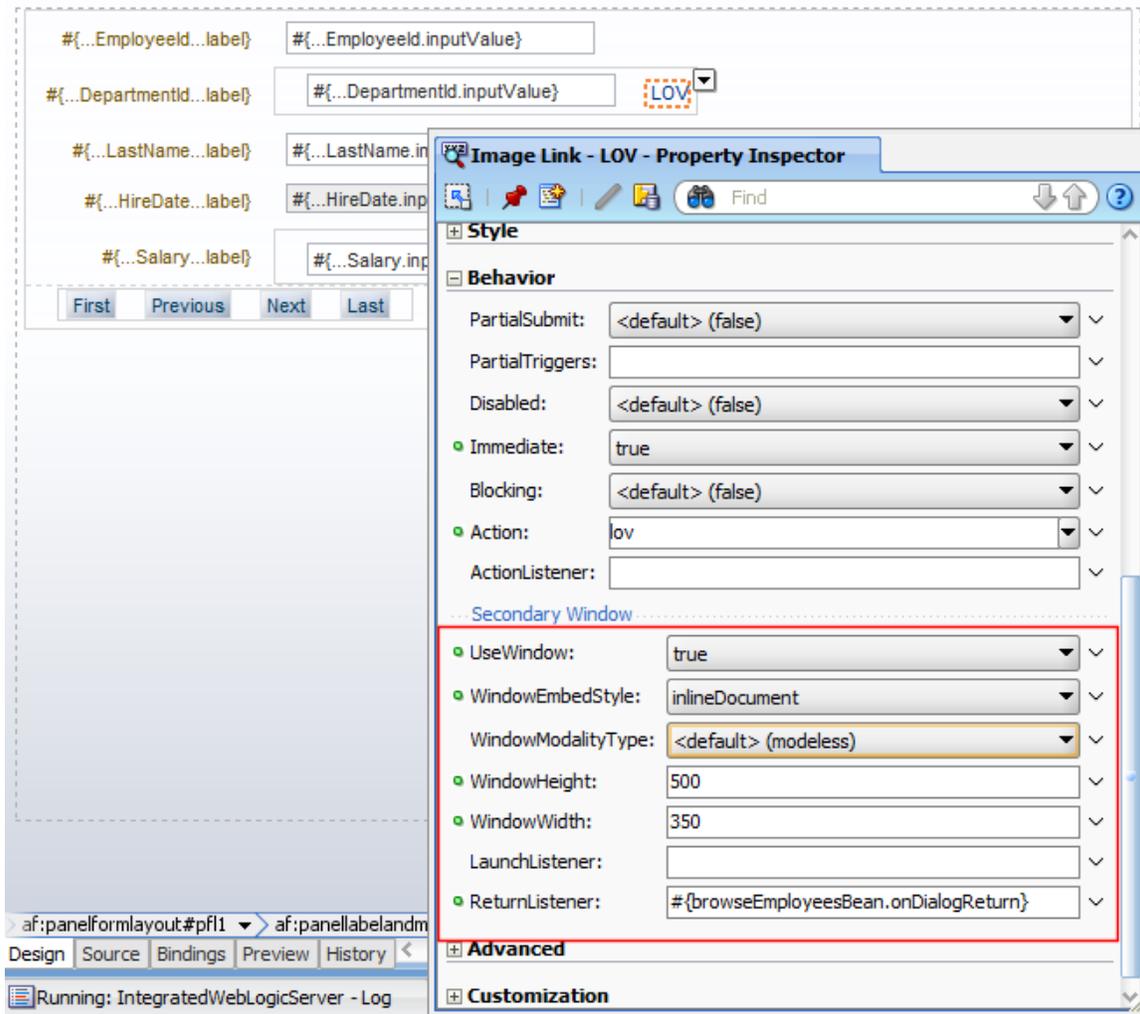
public void setEmployeeTable(RichTable employeeTable) {
    this.employeeTable = employeeTable;
}

public RichTable getEmployeeTable() {
    return employeeTable;
}
}

```



The LOV bounded task flow returns two values, the selected **DepartmentId** and a flag indicating whether the LOV dialog was closed pressing the **Select** or the **Cancel** dialog.



The **LOV** link that launches the dialog is configured to open it in an **inlineDocument**. The LOV width and height is defined on the command link. Upon dialog return, the **ReturnListener** is invoked to read the returned **DepartmentId** and update the DepartmentId field.

```
public class BrowseEmployeesBean {
    private RichInputText departmentIdToUpdate;
    public BrowseEmployeesBean() {}

    public void onDialogReturn(ReturnEvent returnEvent) {
        ADFContext adfCtx = ADFContext.getCurrent();
        Map pageFlowScope = adfCtx.getPageFlowScope();
        //only update the field if the LOV select option was used,
        //ignore for cancel
        Object cancelFlag = pageFlowScope.get("submitOrCancelFlag");
        if(cancelFlag!=null){
            if (((String)cancelFlag).equalsIgnoreCase("submit")) {
                Object departmentId = returnEvent.getReturnValue();
                //call reset value on field that you update with the
                //LOV return value if dialog is launched by a command
            }
        }
    }
}
```

```

        //item having immediate=true set. This enforces the
        //input component to re-read its value
        departmentIdToUpdate.resetValue();
        departmentIdToUpdate.setValue(departmentId);
        AdfFacesContext adfFacesContext =
            AdfFacesContext.getCurrentInstance();
        adfFacesContext.addPartialTarget(departmentIdToUpdate);
    }
}

//input field JSF component binding
public void setDepartmentIdToUpdate(
    RichInputText departmentIdToUpdate) {
    this.departmentIdToUpdate = departmentIdToUpdate;
}

public RichInputText getDepartmentIdToUpdate() {
    return departmentIdToUpdate;
}
}

```

Note that because the LOV command link opens the LOV dialog having its **immediate** property set to **true**, the DepartmentId field must be reset – calling `resetValue` – before update. This also is an important information in this how-to document,

## Sample Download

The Sample can be downloaded as sample #069 on the ADF Code Corner website at

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

Configure the sample database connection to point to a database with the HR schema installed and unlocked. Select the **EmployeesForm.jspx** page and run the sample. Press the **LOV** link next to the **DepartmentId** field and search and select a new value. Press the select or cancel button to copy the selected value or dismiss the selection. Note that the LOV dialog does not show the native close icon.

---

### RELATED DOCUMENTATION

---