ADF Code Corner

046. Building a search form that displays the results in a task flow

twitter.com/adfcodecorner

Abstract:

One of the areas that really rocks in JDeveloper 11 is taskflows. To program usecases that involve taskflows can keep you busy for weeks if the goal is to provide a library of code examples.

The use case explained in this blog post is quite simple and is based on frequent requests posted on the Oracle JDeveloper forum on OTN

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
03-Apr-2008

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

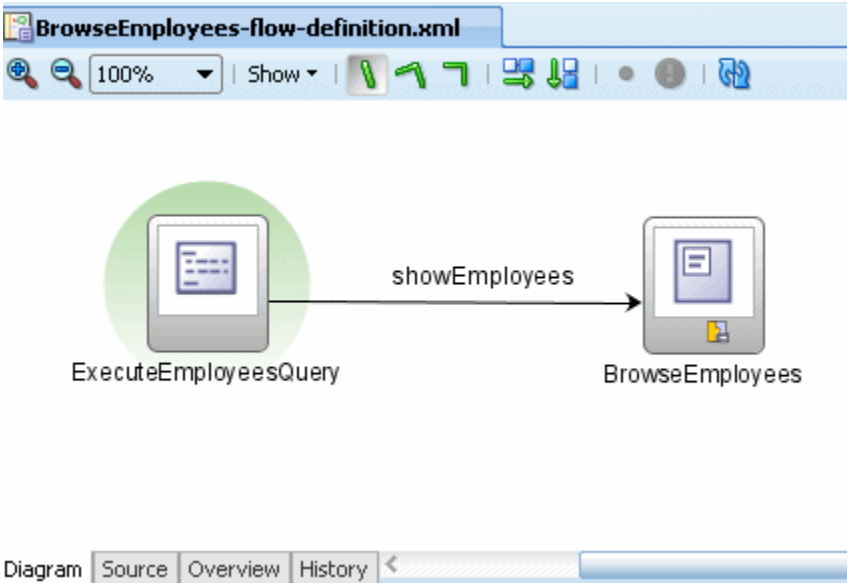
Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

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

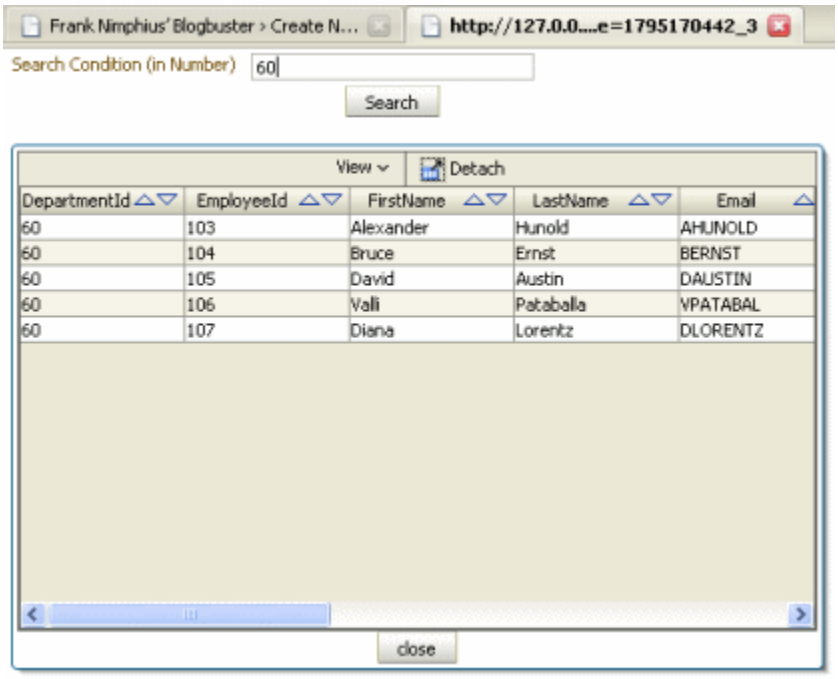
A JSF page has a textfiled and a search button. The user enters a department number and hits the "search" button. The result of this search, which uses ExecuteWithParams in ADF Business Components, shows in a popup dialog on the same page. To enforce reuse - one of the big themes in JDeveloper 11 - the popup content is a taskflow that shows in a region. The challenges solved in this usecase are:

1) The input argument is of type String - as everything on the web starts as a string before it gets converted - whereas the bind variable used within the ExecuteWithParams operation is of type oracle.jbo.domain.Number. Because of this, the result oage cannot be called directly and instead a method needs to be called first to prepare the model before the result page is called



2) The search string needs to be passed to the region as an input parameter. Further, the region needs to be updated whenever a new parameter is passed.

3) The popup needs to be closed programmatically. This part is easy and I am reusing the code from another blog entry of mine At runtime, the application works as shown below



A tiny bit that I haven't yet got working using the af:showPopupBehavior, which works if launching the popup using the af:clientListener with JavaScript, is to proper adjust the position of the popup. However, JDeveloper 11 is not production yet, so no need to worry too much about this yet. Pressing the close button on the popup will dismiss the popup window. Pressing the search button again will re-open it with the query result of the new search input.

How-to

The query is performed within the method activity, which is the default activity in the taskflow and that routes the request to the page fragment that displays the result table once finished.

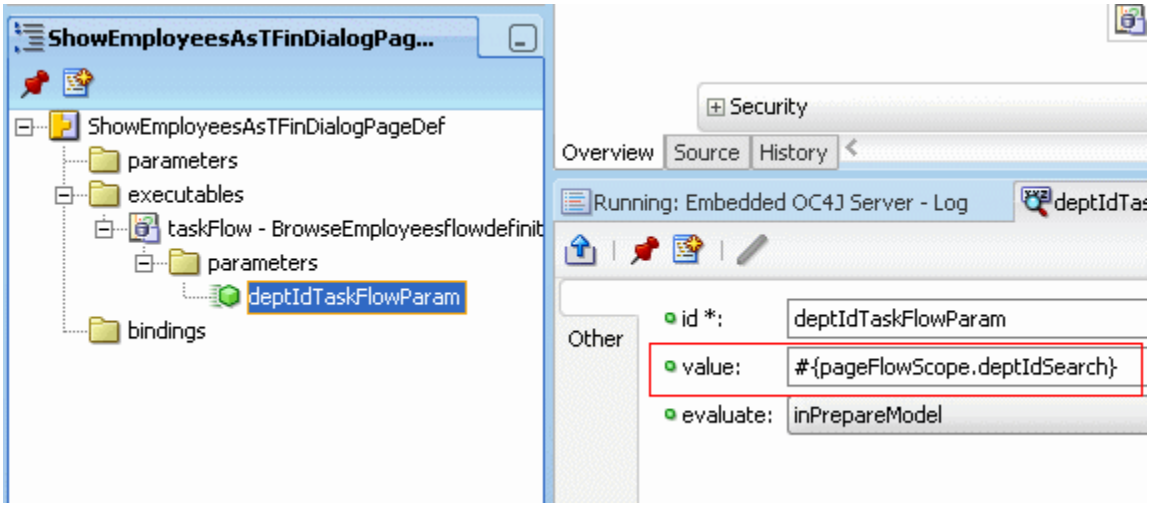
```
public void executeQuery(){

        FacesContext fctx = FacesContext.getCurrentInstance();
        ELContext elctx = fctx.getELContext();
        Application jsfapp = fctx.getApplication();
        ExpressionFactory exprfct = jsfapp.getExpressionFactory();

        ValueExpression adfDataVexpr =
                           exprfct.createValueExpression(elctx,
                                "#{data}",BindingContext.class);
        BindingContext adfdata =
                    (BindingContext) adfDataVexpr.getValue(elctx);
        ValueExpression deptIdDataVexpr =
                  exprfct.createValueExpression(elctx,
                  "#{pageFlowScope.deptIdTaskFlowParam}",Object.class);
        if (deptIdDataVexpr.getValue(elctx) != null){
```

```
            String deptIdStr =
                      (String) deptIdDataVexpr.getValue(elctx);
            try {
              Integer deptIdInt = Integer.parseInt(deptIdStr);
              BindingContainer bindings =
                adfdata.findBindingContainer("BrowseEmployeesPageDef");
                bindings.getOperationBinding(
                   "ExecuteWithParams").getParamsMap().put("deptId",
                    new Number(deptIdInt.intValue()));
                bindings.getOperationBinding("ExecuteWithParams")
                .execute();
            } catch (NumberFormatException ex) {
                ex.printStackTrace();
            }
        }
    return;
    }
```

The initial code lines are there to get a hold of the ExecuteWithParams method that is contained in the pagedef file associated with the page fragment. Theinteresting bits start with line 10. In here, the parameter that is passed from the calling page to the region (taskflow) is accessed and then converted to oracle.jbo.domain.Number before it is used to execute the ExecuteWithParams operation. When you drag a taskflow as a region onto a page, then - if the taskflow requires an input parameter, the pagedef file of the page on which the region is added, is updated with the region definition and input parameter binding



In the image above, you see that the input parameter that is defined on the taskflow, reads its value from #{pageFlowScope.deptIdSearch}, an attribute added to the pageFlowScope of the calling page. The attribute is updated by the search field on the page

```
<af:inputText label="Search Condition (in Number)" id="searchField1"
                        value="#{pageFlowScope.deptIdSearch}"/>
```

So anything that the user types into the search field, gets written to the pageFlowScope attribute when pressing the search button, which performs the form submit. Important to note is that if you want to launch a popup dialog from a button, you must ensure that the button uses a partial submit instead of performing a full page refresh

```
<af:commandButton text="Search" id="searchbutton"
                  partialSubmit="true">
     <af:showPopupBehavior popupId="browseEmployeesPopup"
                           triggerType="click"/>
</af:commandButton>
```

So what do we have until here, and what is missing still ? We have a task flow that starts with a method call to prepare the query for the table. The method takes an input argument that is passed to the taskflow from the region. The region accesses the parameter value from an attribute in the pageFlowScope of the calling page. The remaining job is to make sure the taskflow input parameter writes the input parameter to its own pageFlowScope so the Java method can access it using EL.

```
adfc-config xmlns="http://xmlns.oracle.com/adf/controller"
version="1.2">
  <task-flow-definition id="BrowseEmployees-flow-definition">
    <default-activity>ExecuteEmployeesQuery</default-activity>
    <input-parameter-definition>
      <name>deptIdTaskFlowParam</name>
      <value>#{pageFlowScope.deptIdTaskFlowParam}</value>
      <required/>
    </input-parameter-definition>
    <managed-bean>
      <managed-bean-name>EmployeesBean</managed-bean-name>
      <managed-bean-class>adf.sample.EmployeesBean</managed-bean-class>
      <managed-bean-scope>pageFlow</managed-bean-scope>
    </managed-bean>
    <view id="BrowseEmployees">
      <page>/BrowseEmployees.jsff</page>
    </view>
    <method-call id="ExecuteEmployeesQuery">
      <method>#{pageFlowScope.EmployeesBean.executeQuery}</method>
      <outcome>
        <fixed-outcome>showEmployees</fixed-outcome>
      </outcome>
    </method-call>
    <control-flow-rule>
      <from-activity-id>ExecuteEmployeesQuery</from-activity-id>
      <control-flow-case>
        <from-outcome>showEmployees</from-outcome>
        <to-activity-id>BrowseEmployees</to-activity-id>
      </control-flow-case>
    </control-flow-rule>
    <use-page-fragments/>
  </task-flow-definition>
</adfc-config>
```

The taskflow source shows the following

- The default activity - and thus the first action performed when entering the taskflow - is the method call to a managed bean. <default-activity>ExecuteEmployeesQuery</default-activity>
- The input parameter is defined as "deptIdTaskFlowParam", which is the same name used in the region binding, and writes the obtained value to #{pageFlowScope.deptIdTaskFlowParam} from where it is accessed by the managed bean method

```
<input-parameter-definition>
     <name>deptIdTaskFlowParam</name>
     <value>#{pageFlowScope.deptIdTaskFlowParam}</value>
     <required/>
</input-parameter-definition>
```

The remaining information to give is that I set the "Refresh" property on the region binding in the pagedef file of the calling page to "ifNeeded", which makes sure that the region binding is updated whenever the input parameter changes.

Download

You can download the workspace with the sample - based on the HR schema **from ADF Code Corner**

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

RELATED DOCOMENTATION

☒	
☒	
☒	