052. How-to deploy bounded task flows in an ADF library

Abstract:

Bounded task flow is the reusable process definition in Oracle ADF Controller (ADFc) that extends the JavaServer Faces page navigation handler to perform navigation in ADF Faces applications. ADFc allows developers to build navigation definitions for immediately use in an application, for remote deployment and access and for reuse when deployed in ADF libraries. ADF libraries is a Java Archive (JAR) file concept in ADF that contains a specific manifest file that allow Oracle JDeveloper to detect and import reusable components so they show in the ADF Component palette. One of the artifacts that can be deployed in ADF library files are bounded task flows - as mentioned - which developers deploy with the dependend model definition if required. This how-to article explains how to create and deploy a bounded task flow that contains ADF bound page fragments in ADF libraries and how to import and use this library in an ADF application. It also shows how to customize the name of bounded task flows in an ADF library to show user friendly names instead of the technical source file names. The latter is a hidden, yet undocumented nugget that I am sure developers will enjoy.
How-to deploy bounded task flows in an ADF library

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 image below shows the runtime view of the use case explored in this article. The area surrounded by the red rectangle is a task flow added from an ADF library. Using a task flow input parameter, the department ID is passed to the bounded task flow to query the detail rows. In this article, I explain the following topics:

- Creating View Criteria
- Configuring View Criteria for a View Object instance
- Setting bind variables through a method call activity
- Creating an ADF Library
- Importing an ADF library to the IDE
- Adding an ADF Library to a project
- Configuring the ADF task flow database connection
- Customizing the bounded task flow names in the ADF library
The Bounded Task Flow to reuse

The reusable bounded task flow to deploy in the ADF library consists of a method call activity to set the bind variable of the Employees View Object and a page fragment to show the queried employees. You may argue that this use case doesn't justify a reusable task flow and that it would be much better if the task flow also allows to edit the queried employees records. And right you are - but samples should be simple and for what I want to explain this simple use case is just fine. However, keep your idea in mind when extending this sample ;-) 

**Note:** Before you start. Any content that you plan to deploy in an ADF library is subject to naming conflicts, which is why you should carefully consider a naming strategy for the package structure, component names and class names of the objects added to a library.

Query by examples in ADF Business Components can be created in one of two way: i) adding a where clause to the View Object query and use a bind variable to pass dynamic values in and ii) create and apply a named View Criteria that also uses a bind variable for the dynamic by-Example value passed in. In this article I use the latter approach. To create the View Criteria, double click the View Object and select the View Criteria option in the Query panel. Define a bind variable input the department ID from external.
To set the bind variable from the ADF binding layer, you need to expose a setter method for the bind variable on the client interface. Open the View Object and select the Java option. Click the pencil icon to open the Java option dialog. Check the "Generate View Object Class" option and ensure the "Include bind variable accessors" is selected too. This generates a setter and getter method for the bind variable.

For this article, I edited the setter method and added `this.executeQuery()` to make sure the View Object is re-queried with the new bind variable value immediately after it is set.

Select the client interface option of the Java panel and shuffle the setter method of the bind variable to the list of selected client methods. This exposes the method as a child operation of the View Object in the Data Controls palette.
To add a View Criteria to a View Object instance, open the Data Model by a double click onto the Application Module. Select the View Object instance to apply the View Criteria and press the Edit button. In the opened dialog, select the View Criteria in the "Available" list and shuffle it to the "Selected" list.

This instance of the View Object will always have the View Criteria applied. With this, the ADF Business Component model is ready and the next step is to build the bounded task flow, which is the reusable
How-to deploy bounded task flows in an ADF library

To create the bounded task flow, choose New | JSF | ADF Task Flow in Oracle JDeveloper. From the Data Controls palette, expand the Employees View Object and drag and drop the set method of the bind variable as a method call activity to the bounded task flow.

Then, drag and drop the View Object as a read only table and create a control flow case between the two activities. To pass the department ID, create an input parameter for the task flow that writes the parameter value to a pageFlowScope attribute, for example #{pageFlowScope.deptId}.

When you drag and drop the binding setter method, a dialog is shown for you to enter a value for the bind variable. Add the reference to the task flow input parameter as the value: #{pageFlowScope.deptId}. Because the method call activity is the default activity, refreshing the bounded task flow re-executes the Employees View Object.

Creating the ADF Library

To create the ADF library, double click the View project of the reusable component application. Select the deployment entry and press the New button to create a new deployment profile. Choose ADF Library as the archive type and edit the "Connections" entry. The "Connections" configuration defines how the database connection is added to the ADF library. You can add the connection with all the connection details - except the password - or just the name. In both cases, the database connection needs to be edited in the project that imports the ADF library.

The JAR option is a setting I use later in the advanced topic section, in which I explain how to customize the task flow name in the ADF library.
To deploy the ADF library, select the View layer project that contains the ADF bounded task flow and choose the deploy option from the right mouse context menu. Choose the ADF library entry to deploy the ADF library as a JAR file. By default the JAR file is created in the deploy folder of the View layer object.

**Importing the Task Flow Library to an ADF Faces Web Project**

In this blog article, the bounded task flow in the ADF library is imported to a page that uses a panel splitter to layout the master area and the detail area. Because the task flow in the ADF library is created as a page fragment, it will be added as an ADF region.
How-to deploy bounded task flows in an ADF library

The image below shows the configuration of the parent application before the ADF library is added. The Application Navigator view shows a single database connection and a single Data Control that is shown in the Data Controls palette and defined in the DataBindings.cpx file.

Before you can use the ADF library in your application development, you need to add it to the Oracle JDeveloper Resource Palette. Open the Resource Palette from the View menu or press ctrl+shift+O. Press the folder icon and choose New Connection | File System and point the dialog to the directory containing the ADF library JAR file. Don’t point to the JAR directly as this won’t work. In this example, the ADF library was deployed to the "Deploy" directory of the View Layer project within the bounded task flow application. In a real life scenario you may choose a server location to hold all your reusable ADF libraries.
How-to deploy bounded task flows in an ADF library

After OK’in the dialog, the ADF library JAR file is added to th Resource Palette. The name “TaskFlowLib” is the name I chose for the file system connection I created before. The ADF library, as you can see, is added by the name of the JAR file. This is not really user friendly, but I have a good solution for this at the end of this article.
To add the project to a web application, select the View layer project of it and choose "Add to project" from the context menu. This creates an ADF Library entry to the project libraries and also sets up the Component palette.

The ADF Library that I added in this article contains an ADF Business Components model, a database connection and the bounded task flow. As you can see in the image below, the new database connection "hrconn" is now listed under the "Connections" entry in the Application Resources accordion. The Data Controls panel shows the added Data Control, which is "AppModuleDataControl" in my example (the name could be chosen more unique, I agree). Note that the DataBindings.cpx file is not updated because the DataBindings.cpx file that is contained in the ADF Library file is automatically detected when adding the bounded task flow as a region.
Looking at the Component Palette, a new entry "MyBoundedTaskFlowLib.jar" becomes available that contains the name of the bounded task flow region. Still, both namings are technical source names and may not speak to the developer reusing the resource. I will come back to this later and explain how to beautify this.

You can drag the Region entry to a JSF page where it is added as an ADF region.
The bounded task flow is configured with a mandatory input parameter for the department Id, which I need to query the detail rows of the Employees View Object.

Using the Expression Builder, the deptId is linked to the DepartmentsId of the parent form. Note that the reference is to the DepartmentId.inputValue.
Next, I edit the database connection that was imported by the ADF library because it either contains no connect information - if only the name is deployed - or lacks the password - which is not deployed for security reasons. In the Application Resources panel, select the database connection and choose "Properties" from the context menu. This opens the database connection editor.

Note: the hrconn connection is not displayed in the database connection navigator in JDeveloper. This is because the connection is added as a resource and not as a reusable connection.

For the use case of a master / detail scenario between the parent page and the bounded task flow, two additional steps are required:
• Set up partial triggers between the ADF region and the navigation buttons

• Ensure the ADF region is refreshed when a new departments record is navigated to

To setup partial refresh between the ADF region and the parent form, select the ADF region and press the arrow icon on the PartialTriggers property. Choose "Edit" from the context menu and browse the page hierarchy for the navigation buttons. When creating a navigation button bar in ADF, the command buttons have the partialSubmit property set to "true", which means that pressing the button does not lead to a full page refresh but just a refresh of the form. So all I need to do is to include the ADF region to the partial refresh notification, which is what I do by configuring the PartialTriggers property.
To ensure the ADF region refreshes when the dependent parent record changes its row currency, press the "Bindings" tab on the parent JSPX page to access the ADF binding definition. Select the task flow binding in the Executables section and open the Property Inspector. Set the "Refresh" property to "ifNeeded" and the "RefreshCondition" to #{bindings.DepartmentId.inputValue} so that a change of the Department Id value, which happens when navigating the parent row set, leads to an ADF region refresh. The bounded task flow is re-executed when the region refreshes, which also means that the new Departments Id value is read and the Employees View Object executed.

You can now run the application and see the master/detail behavior working. At least you should be able to see it .... to err’ is human.
Advanced Tip: Customizing the ADF Task Flow name in the ADF library

I pointed this out quite frequently in this article that adding an ADF library to an application shows its content with technical namings that user friendly strings. At least for task flows this does not need to be the case and user friendly names can be defined in a custom manifest and properties file.

In the View Layer project of the task flow project that you created the ADF library from, create a new file choosing New | General | File from the JDeveloper menu. The name of the manifest is up to you, the extension though should be ".mf".

The content of the manifest has a Bundle-Classes attribute, which references a properties file. The properties file maps a task flow definition to a user friendly name, a description text and a tooltip. Optionally you can hide a task flow from showing in the ADF library, which makes sense for task flows that you want to call from another task flow but don't allow developers to use stand alone. In the image below, the properties file is referenced from the "Bundle-Classes" attribute and has the name "BoundedTaskFlowLibrary" (BoundedTaskFlowLibrary.properties), a name I came up with for this example.
The manifest file needs to be added to the ADF Library deployment configuration, which you do by double clicking the View Layer project containing the bounded task flow and choosing the "Deployment" entry. Select the ADF library profile and press the "Edit" button. In the JAR Options, press the "Add" button to include the manifest file you created. The image below details this steps. Don't forget!

Important: The manifest file needs to be added to the ADF library deployment profile.

Like creating the manifest file, create a properties file by selecting New | General | File from the JDeveloper menu. Make sure the file name is created with the extension ".properties" and has the same name as defined under "Bundle-Classes" in the manifest. The properties file must be stored in the View layer project's "src" directory. If the folder does not exit yet - which will be the case if your View Layer does not yet have managed beans created - create it. You can create the src directory by browsing to the ViewController project and appending "\src" when creating the properties file.
The properties file can configure multiple task flows, mapping them to meaningful and user friendly namings. To configure a user friendly name for a task flow, you use the "ADF_TaskFlow|" key word followed by the absolute task flow document name starting from the WEB-INF directory and the taskflow id. This way, all task flows in an ADF library can be uniquely addressed.

The attributes on top of the properties file define the user friendly naming of the ADF library itself. The name is shown in the component palette to identify the component panel. Its shown in an image later in this document.

The image below shows a properties file that defines a user friendly name, a description and tooltip for the contained task flow. You can also hide a task flow from showing in an ADF library by setting the "visible" attribute to false (this line is commented in the image below).

When you import an ADF library that is configured with a custom manifest file and properties file, the names from the properties file are shown in the Resource Palette, replacing the physical resource names. The image below shows how the settings from the properties file are displayed for the imported ADF library. Notice the tooltip description.
**Note:** If changes don't show immediately after updating an existing imported library, close and reopen the JDeveloper IDE to enforce the refresh.

Furthermore the Component palette shows the value of the ".name" value added on top of the properties file in the component list box. The task flow itself also is shown by its name in the properties file.
Download the Sample

Both application workspaces are available for download. To run the application, edit both database connections to point to a database of yours that has the HR schema installed. Also verify that the ADF library is found. Optionally you can re-import the library jar file from the Library application's ViewLayer project "deploy" folder. Download from ADF Code Corner:


RELATED DOCUMENTATION

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><a href="http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/toc.htm">http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/toc.htm</a></td>
</tr>
<tr>
<td></td>
<td>ADF Library documentation -</td>
</tr>
<tr>
<td></td>
<td><a href="http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/reusing_components.htm#BABCHHHH">http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/reusing_components.htm#BABCHHHH</a></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>