ADF Code Corner

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 cteate 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.

twitter.com/adfcodecorner

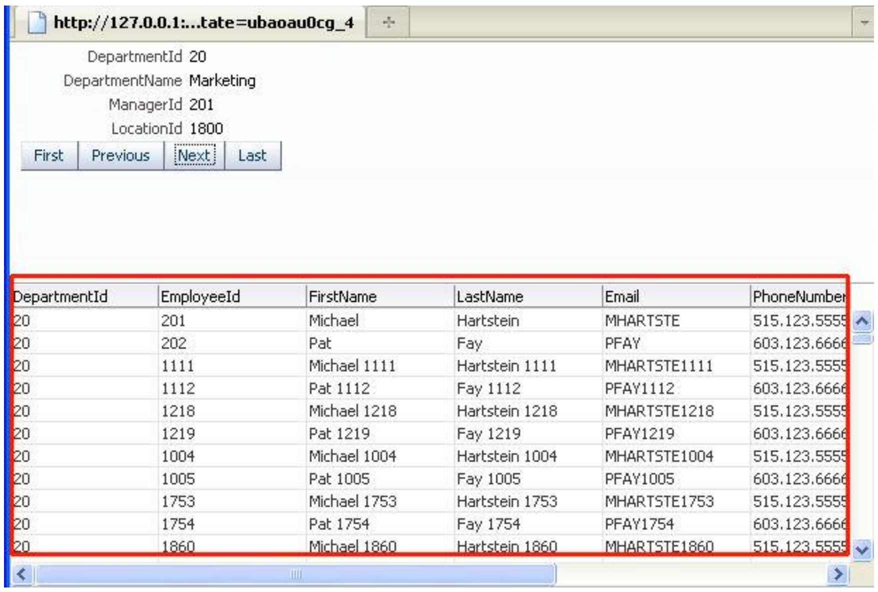
Author: Frank Nimphius, Oracle Corporation

twitter.com/fnimphiu
16-FEB-2010

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 conection
- Customizing the bounded task flow names in the ADF library

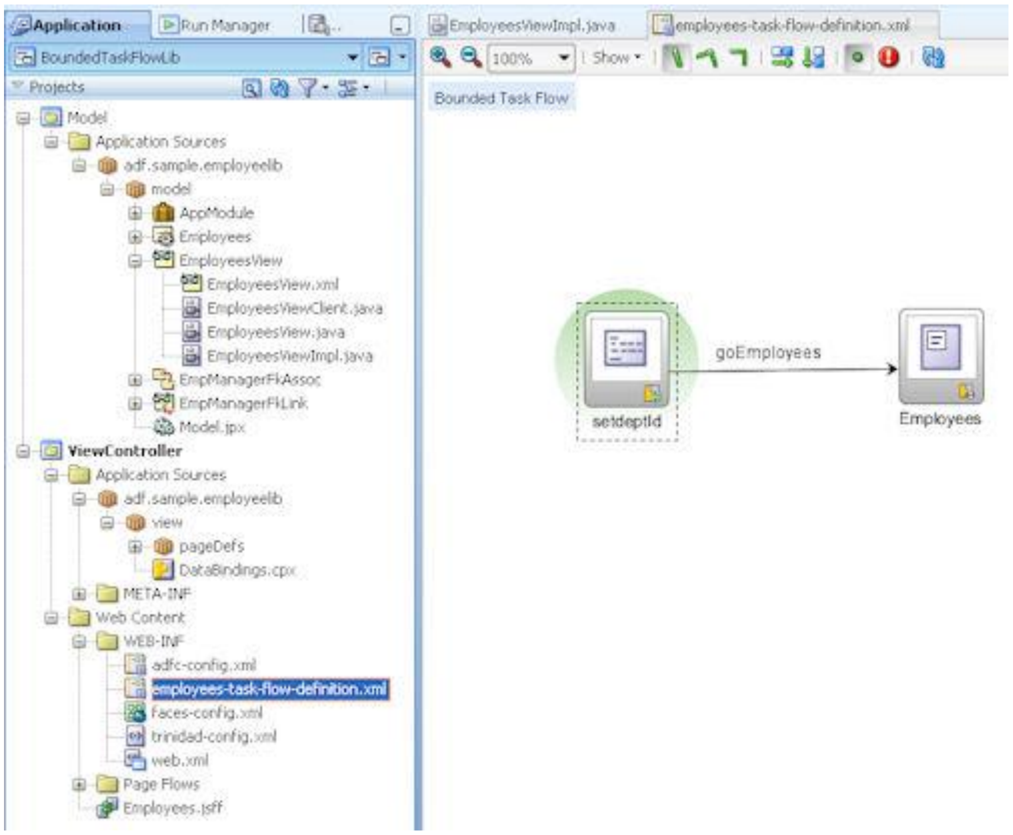


DepartmentId 20
DepartmentName Marketing
ManagerId 201
LocationId 1800

DepartmentId	EmployeeId	FirstName	LastName	Email	PhoneNumber
20	201	Michael	Hartstein	MHARTSTE	515.123.5555
20	202	Pat	Fay	PFAY	603.123.6666
20	1111	Michael 1111	Hartstein 1111	MHARTSTE1111	515.123.5555
20	1112	Pat 1112	Fay 1112	PFAY1112	603.123.6666
20	1218	Michael 1218	Hartstein 1218	MHARTSTE1218	515.123.5555
20	1219	Pat 1219	Fay 1219	PFAY1219	603.123.6666
20	1004	Michael 1004	Hartstein 1004	MHARTSTE1004	515.123.5555
20	1005	Pat 1005	Fay 1005	PFAY1005	603.123.6666
20	1753	Michael 1753	Hartstein 1753	MHARTSTE1753	515.123.5555
20	1754	Pat 1754	Fay 1754	PFAY1754	603.123.6666
20	1860	Michael 1860	Hartstein 1860	MHARTSTE1860	515.123.5555

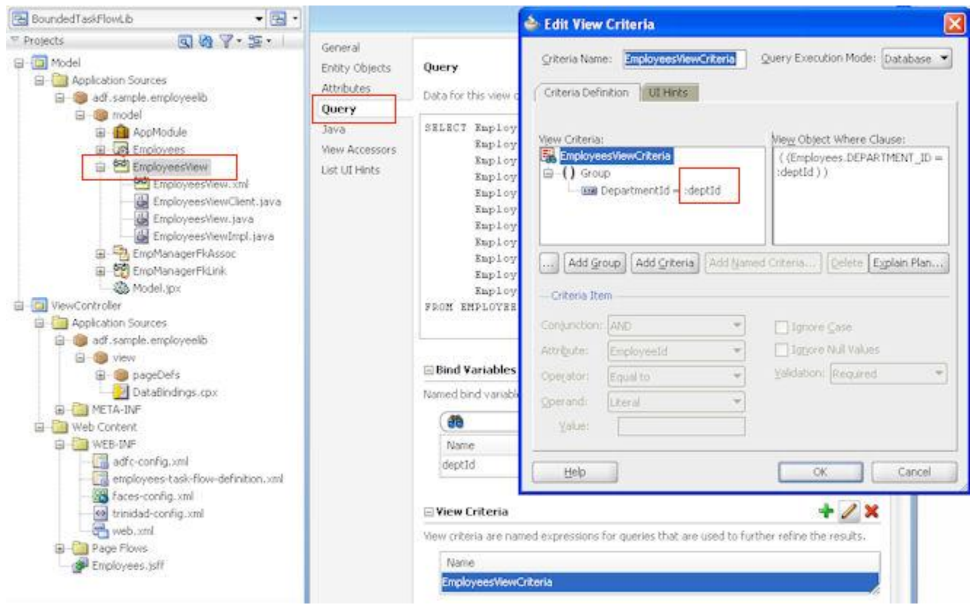
The Bounded Task Flow to reuse

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 Objec and select the View Criteria option in the Query panel. Define a bind variable input the department ID from external.



Criteria Name: EmployeesViewCriteria
Query Execution Mode: Database

View Criteria:
- EmployeesViewCriteria
  - Group
    - DepartmentId = :deptId

View Object Where Clause: ((Employees.DEPARTMENT_ID = :deptId))

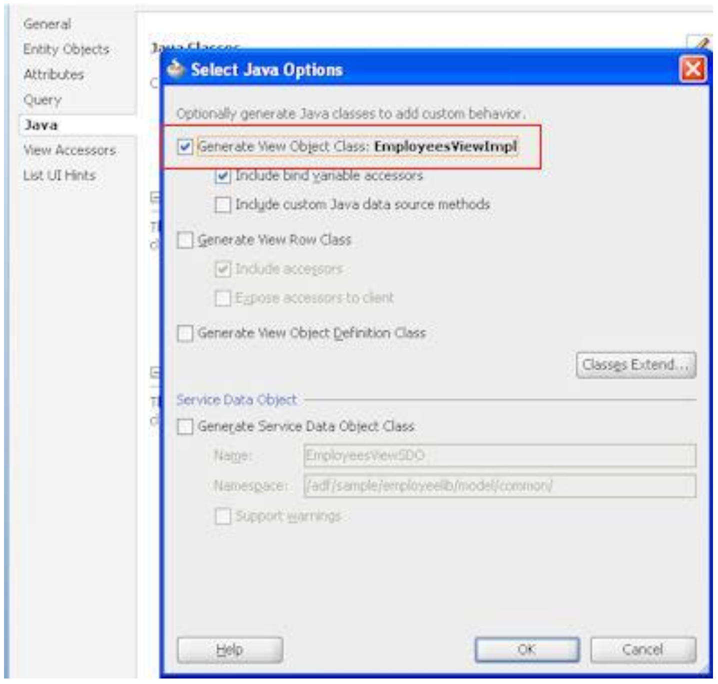
Criteria Item
Conjunction: AND
Attribute: EmployeeId
Operator: Equal to
Operand: Literal
Value:
Ignore Case
Ignore Null Values
Validation: Required

Bind Variables: deptId

View criteria are named expressions for queries that are used to further refine the results.

Name: EmployeesViewCriteria

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.



Optionally generate Java classes to add custom behavior.

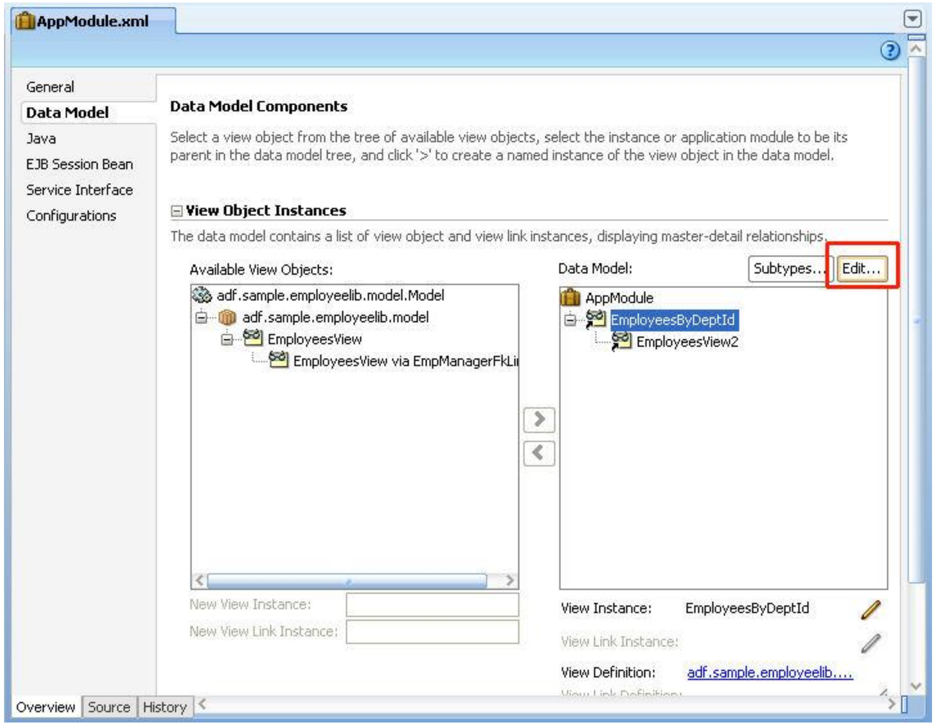
- [✓] Generate View Object Class: **EmployeesViewImpl**
  - [✓] Include bind variable accessors
  - [ ] Include custom Java data source methods
- [ ] Generate View Row Class
  - [✓] Include accessors
  - [ ] Expose accessors to client
- [ ] Generate View Object Definition Class

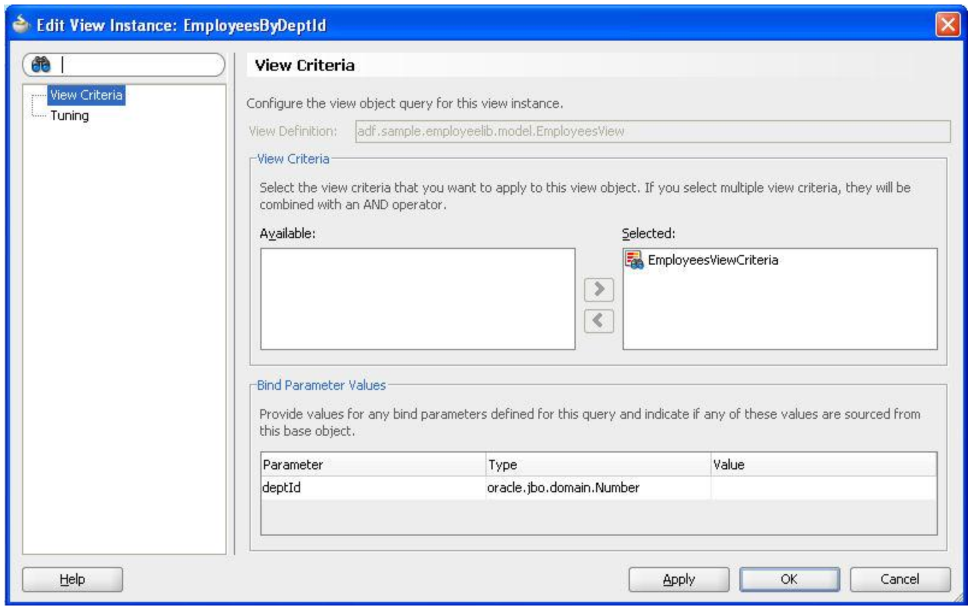
Service Data Object
- [ ] Generate Service Data Object Class
  - Name: EmployeesViewSDO
  - Namespace: /adf/sample/employeelib/model/common/
  - [ ] Support warnings

Select the client interface option of the Java panel and shuffle the setter method of the bind variabe 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 douple 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.


AppModule.xml
General
Data Model
Java
EJB Session Bean
Service Interface
Configurations
Data Model Components
Select a view object from the tree of available view objects, select the instance or application module to be its parent in the data model tree, and click '>' to create a named instance of the view object in the data model.
View Object Instances
The data model contains a list of view object and view link instances, displaying master-detail relationships.
Available View Objects:
adf.sample.employeelib.model.Model
adf.sample.employeelib.model
EmployeesView
EmployeesView via EmpManagerFkLi
Data Model:
Subtypes...
Edit...
AppModule
EmployeesByDeptId
EmployeesView2
New View Instance:
New View Link Instance:
View Instance: EmployeesByDeptId
View Link Instance:
View Definition: adf.sample.employeelib....
Overview
Source
History


Edit View Instance: EmployeesByDeptId
View Criteria
Tuning
View Criteria
Configure the view object query for this view instance.
View Definition: adf.sample.employeelib.model.EmployeesView
View Criteria
Select the view criteria that you want to apply to this view object. If you select multiple view criteria, they will be combined with an AND operator.
Available:
Selected:
EmployeesViewCriteria
Bind Parameter Values
Provide values for any bind parameters defined for this query and indicate if any of these values are sourced from this base object.
Help
Apply
OK
Cancel

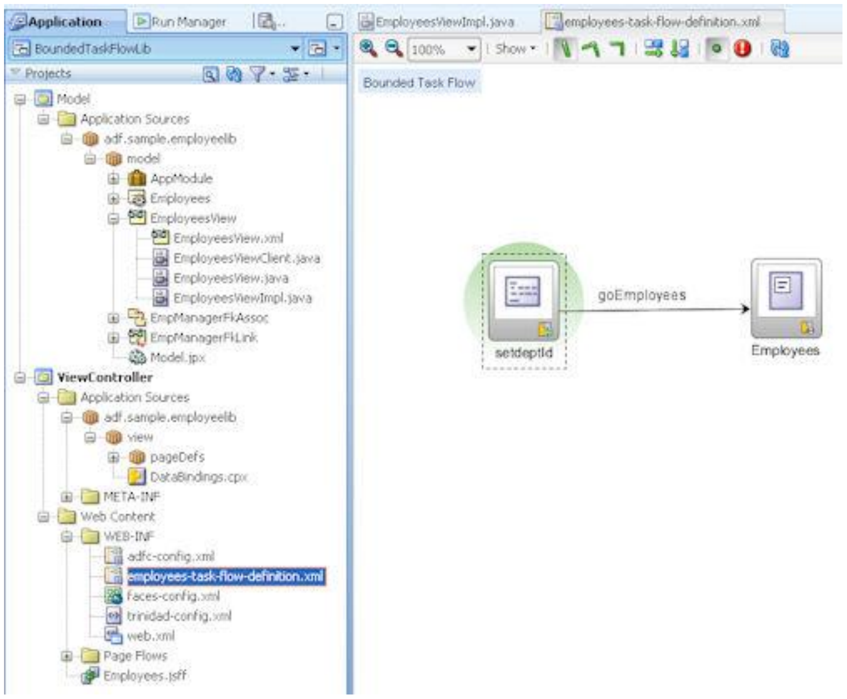
Parameter	Type	Value
deptId	oracle.jbo.domain.Number	

This instance of the View Object will alsways 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 component to deploy in the 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}.

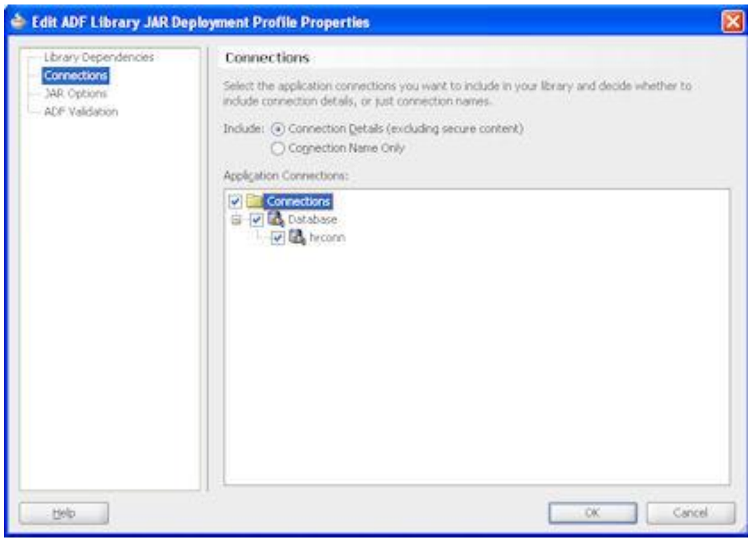
When you drag and drop the binding setter method, a dialog is shown for you to enter a value for the bind variable. Add the reference to the task flow input parameter as the value: #{pageFlowScope.deptId}. Because the method call activity is the default activity, refreshing the bounded task flow re-executes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

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.


Edit ADF Library JAR Deployment Profile Properties

Connections

Select the application connections you want to include in your library and decide whether to include connection details, or just connection names.

Include: [✓ Connection Details (excluding secure content)] Connection Name Only

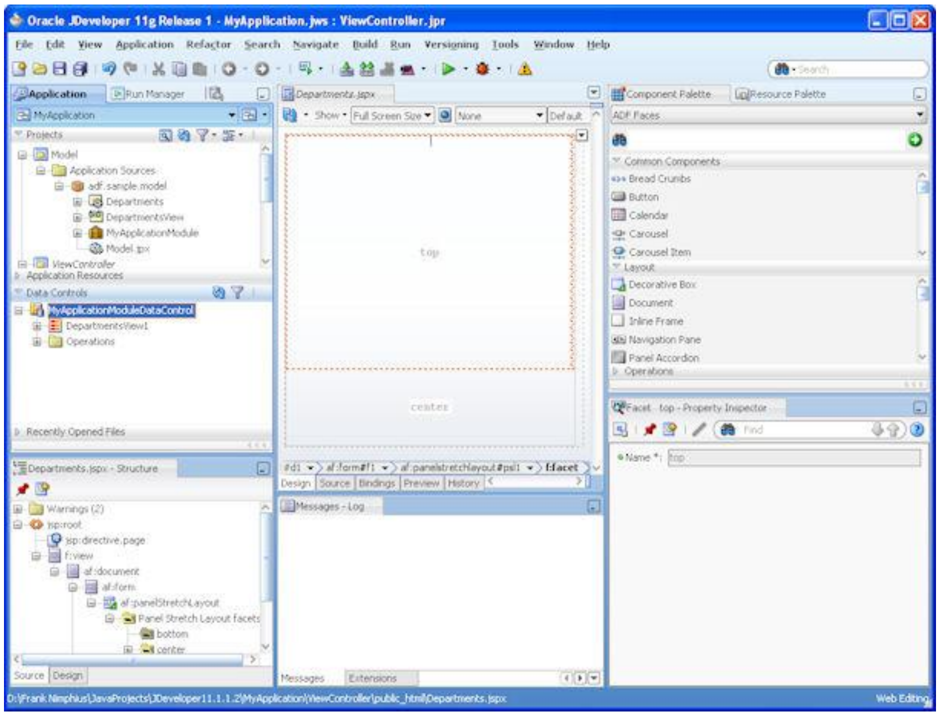
Application Connections:

- ☑ Connections
  - ☑ Database
    - ☑ hrconn

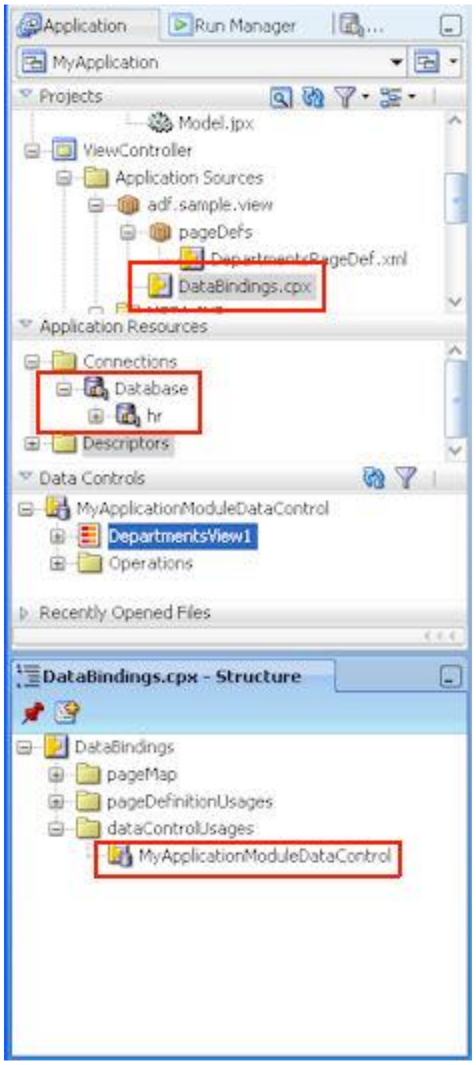
To deploy the ADF library, select the View layer project that contains the ADF bounded task flow and choose the deploy option from the right mouse context menu. Choose the ADF library entry to deploy the ADF library as a JAR file. By default the JAR file is created in the deploy folder of the View layer object.

Importing the Task Flow Library to an ADF Faces Web Project

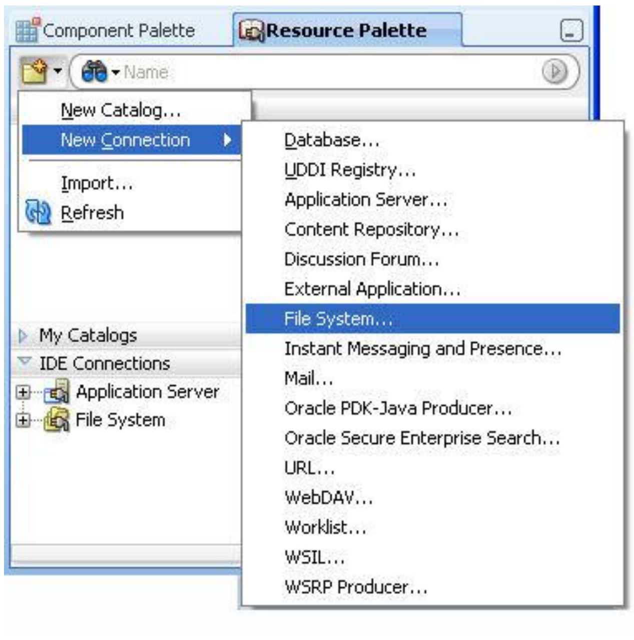
In this blog article, the bounded task flow in the ADF library is imported to a page that uses a panel splitter to layout the master area and the detail area. Because the task flow in the ADF ibrary is created as a page fragment, it will be added as an ADF region.



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.

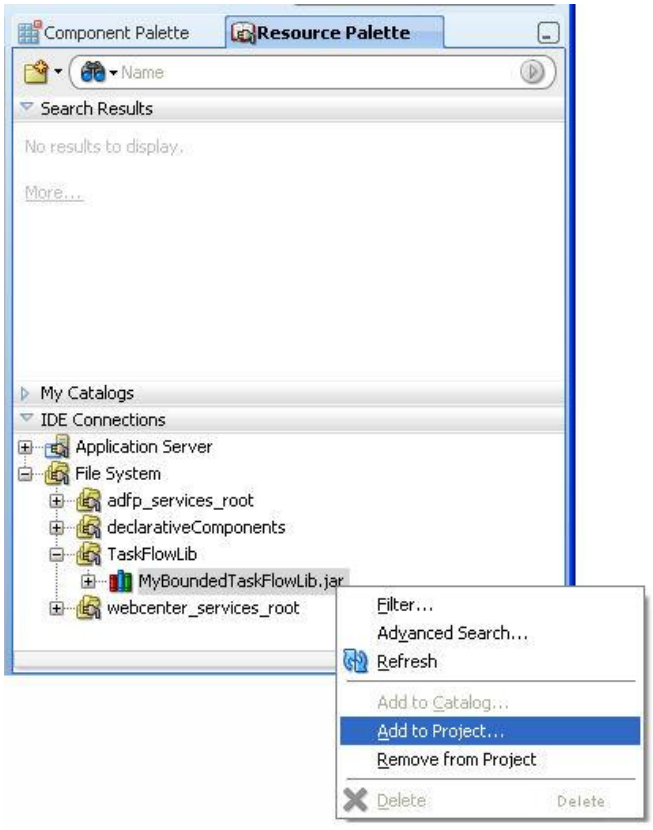


Before you can use the ADF library in your application development, you need to add it to the Oracle JDeveloper Resource Palette. Open the Resource Palette from the View menu or press ctrl+shift+O. Press the folder icon and choose New Connection | File System and point the dialog to the directory containing the ADF library JAR file. Don't point to the JAR directly as this wont 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.



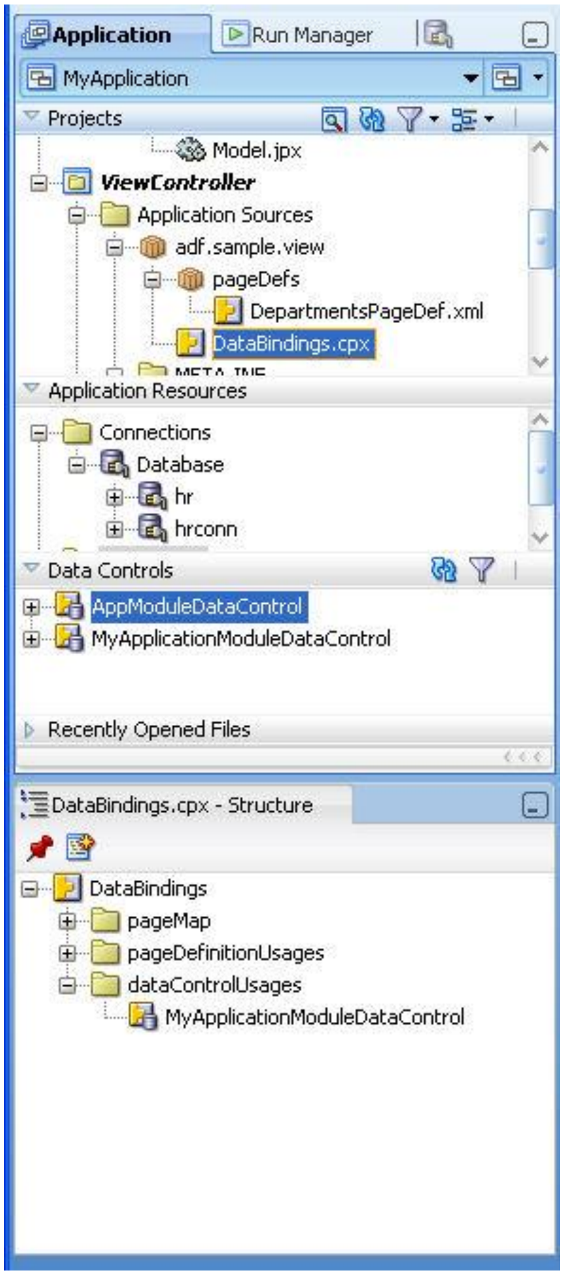
After OK'in the dialog, the ADF library JAR file is added to th Resource Palette. The name "TaskFlowLib" is the name I chose for the file system connection I created before. The ADF library, as you can see, is added by the name of the JAR file. This is not really user friendly, but I have a good solution for this at the end of this article.

.

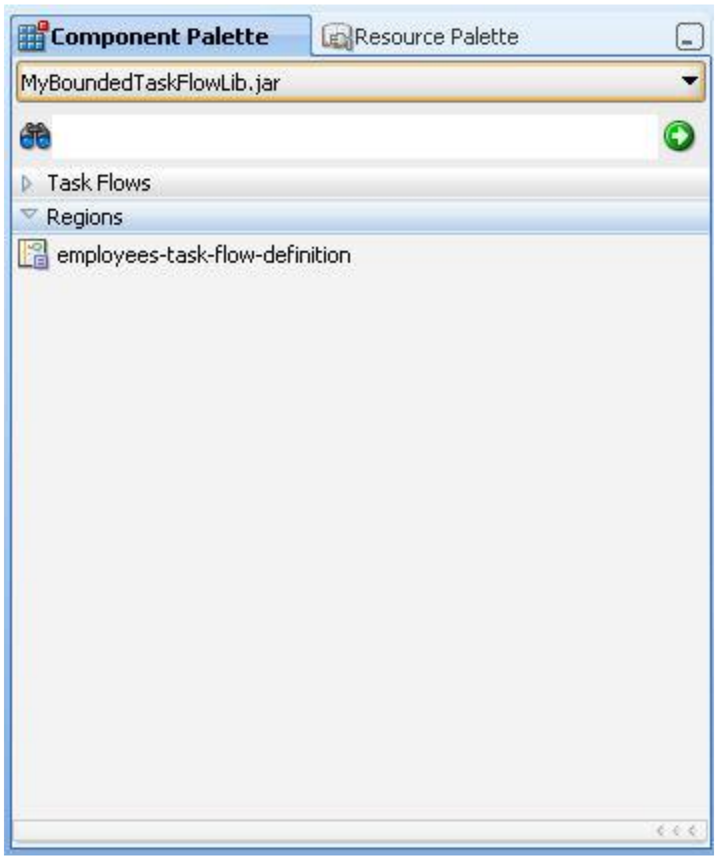


To add the project to a web application, select the View layer project of it and choose "Add to project" from the context menu. This creates an ADF Library entry to the project libraries and also sets up the Component palette

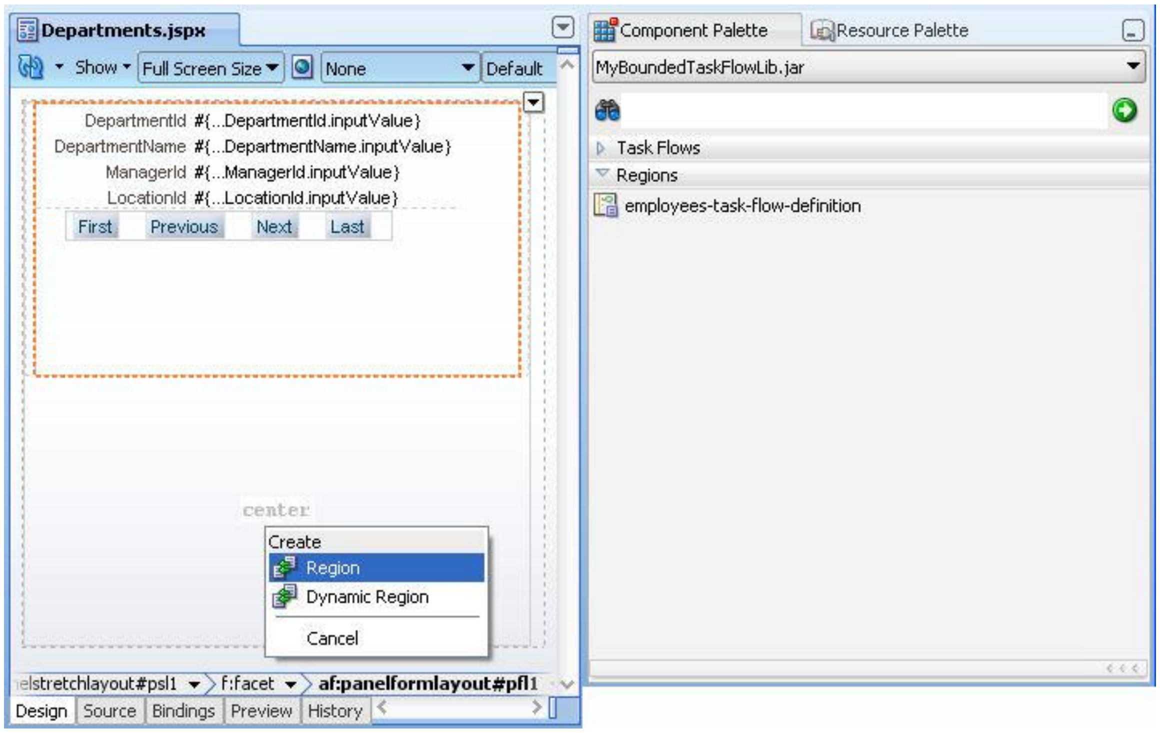
The ADF Library that I added in this article contains an ADF Business Components model, a database connection and the bounded task flow. As you can see in the image below, the new database connection "hrconn" is now listed under the "Connections" entry in the Application Resources accordion. The Data 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.



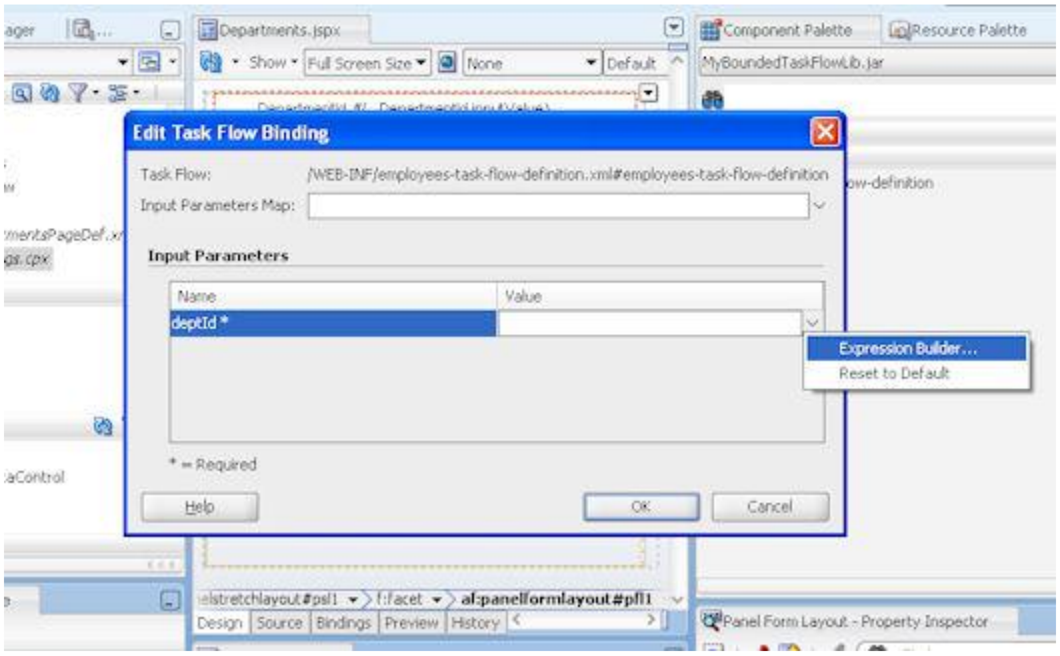
Looking at the Component Palette, a new entry "MyBoundedTaskFlowLib.jar" becomes available that contains the name of the bounded task flow region. Still, both namings are technical source names and may not speak to the developer reusing the resource. I will come back to this later and explain how to beautify this.



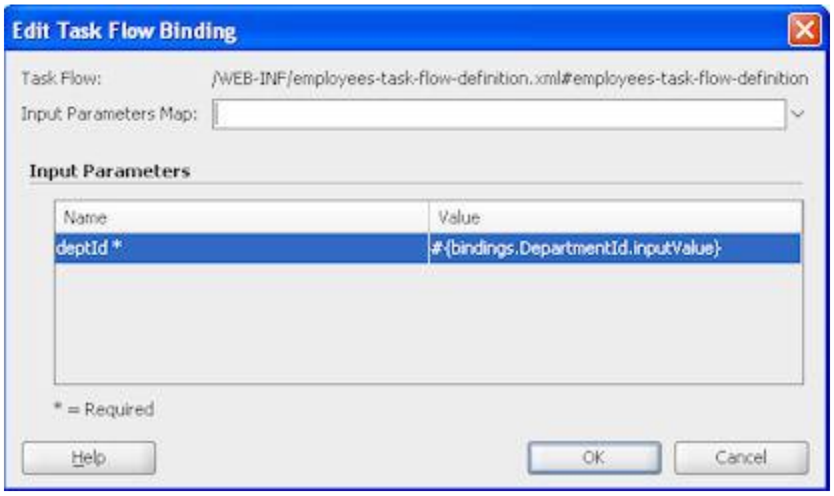
You can drag the Region entry to a JSF page where it is added as an ADF region.



The bunded 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.


Edit Task Flow Binding

Task Flow: /WEB-INF/employees-task-flow-definition.xml#employees-task-flow-definition

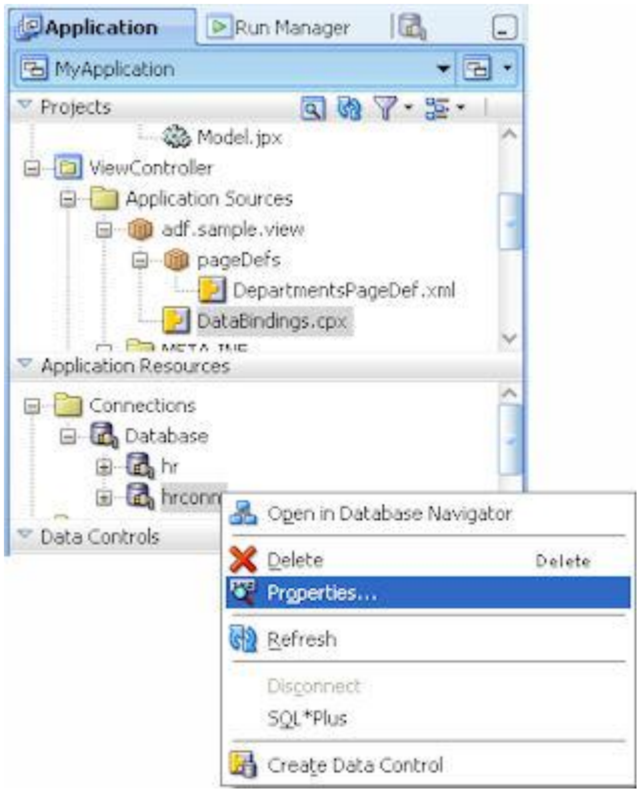
Input Parameters Map:

Input Parameters

Name	Value
deptId *	#{bindings.DepartmentId.inputValue}

* = Required

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.

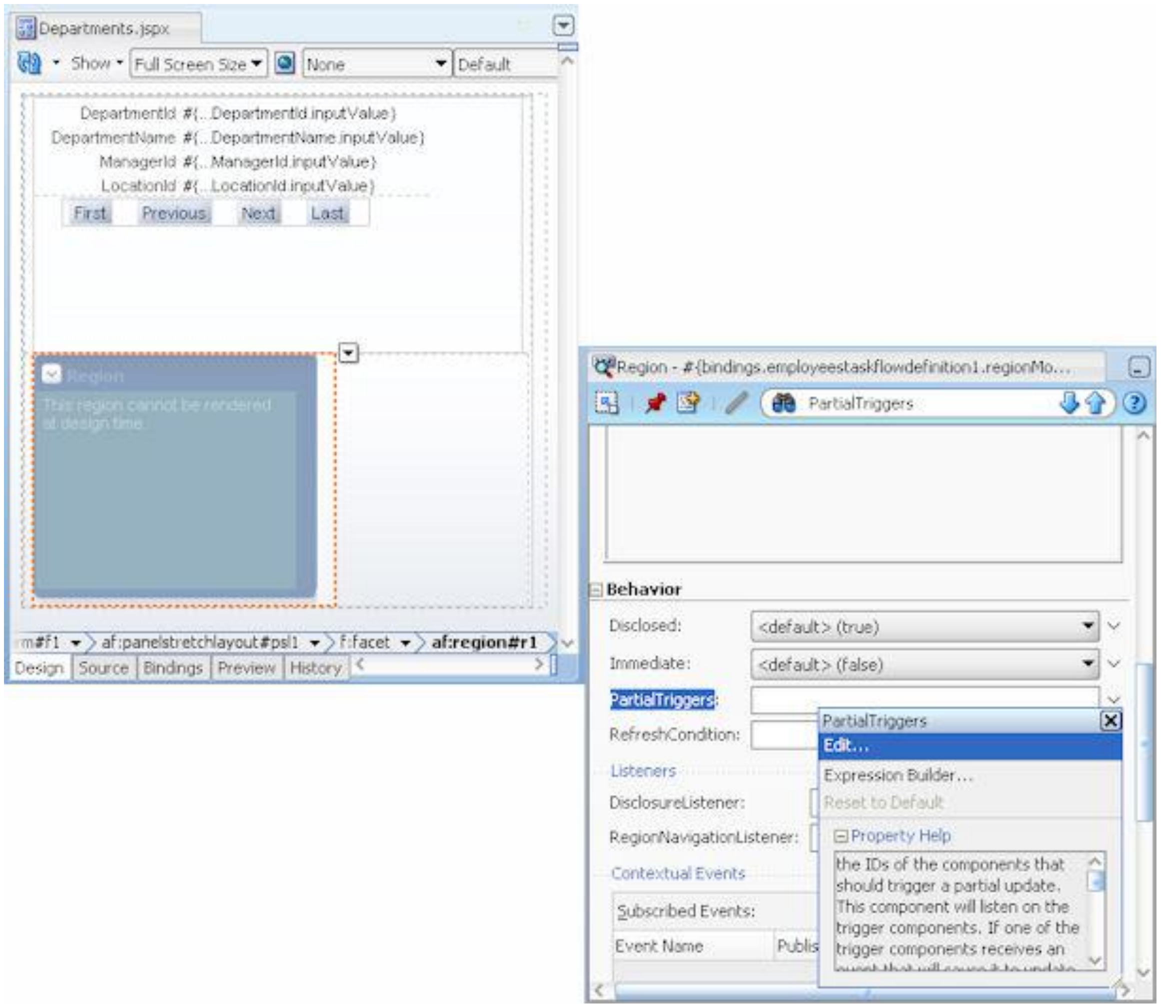


Note: the hrconn connection is not displayed in the database connection navigator in JDeveloper. This is because the connection is added as a resource and not as a reusable connection

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

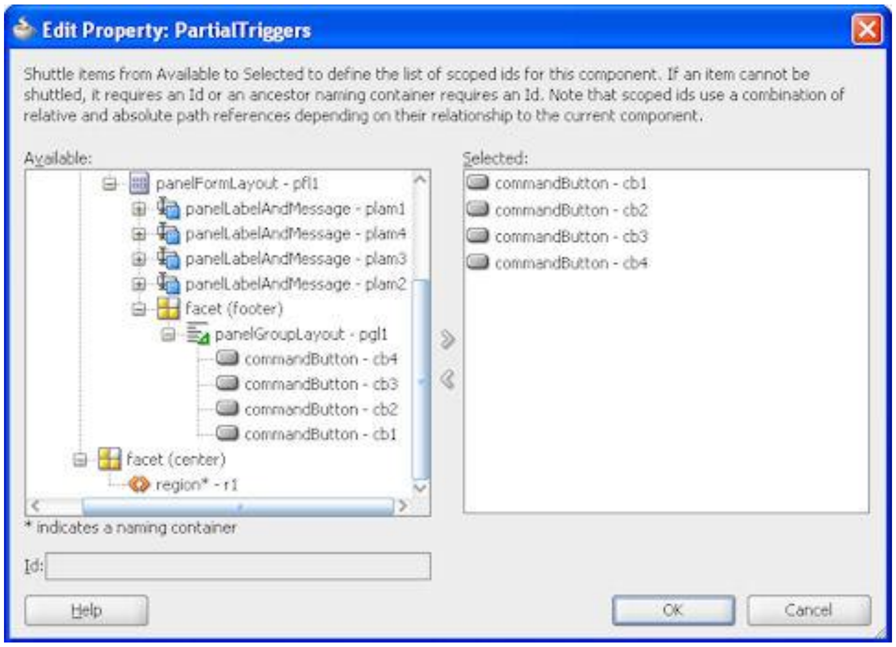
To setup partial refresh between the ADF region and the parnt form, select the ADF region and press the arrw icon on the PartialTriggers property. Choose "Edit" fromteh context menu and browse the pahe hierarchy for the navigation buttons. When creating a navigation button bar in ADF, the command buttons 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.



DepartmentId #{...DepartmentId.inputValue}
DepartmentName #{...DepartmentName.inputValue}
ManagerId #{...ManagerId.inputValue}
LocationId #{...LocationId.inputValue}

Disclosed: <default> (true)
Immediate: <default> (false)
PartialTriggers:
RefreshCondition:

the IDs of the components that should trigger a partial update. This component will listen on the trigger components. If one of the trigger components receives an

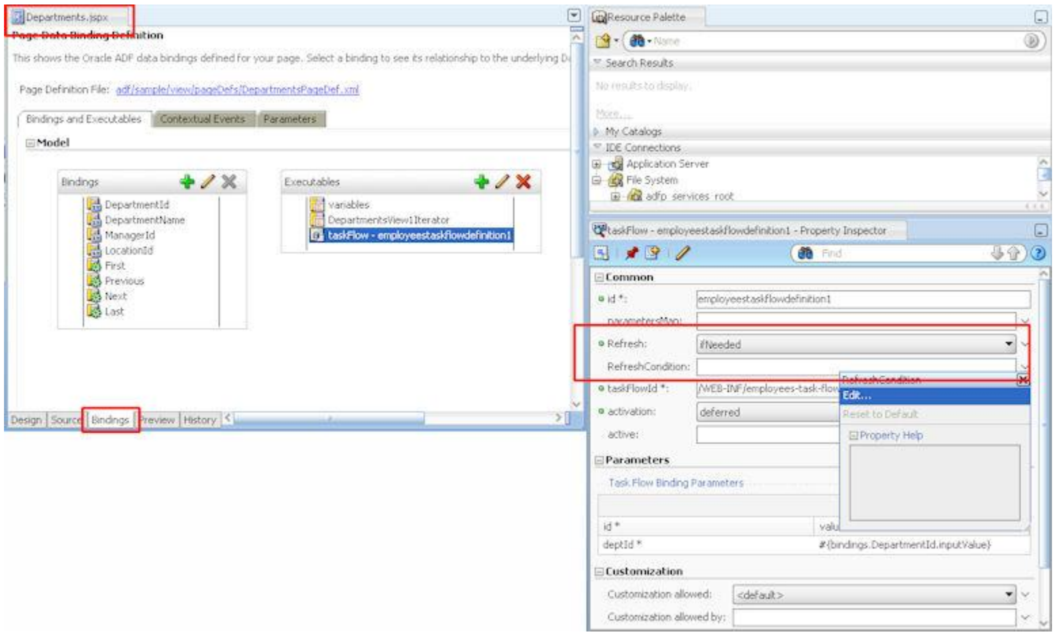


Shuttle items from Available to Selected to define the list of scoped ids for this component. If an item cannot be shuttled, it requires an Id or an ancestor naming container requires an Id. Note that scoped ids use a combination of relative and absolute path references depending on their relationship to the current component.

Selected:
commandButton - cb1
commandButton - cb2
commandButton - cb3
commandButton - cb4

* indicates a naming container

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 regio refreshes, which also means that the new Departments Id value is read and the Employees View Object executed.



Page Data Binding Definition

This shows the Oracle ADF data bindings defined for your page. Select a binding to see its relationship to the underlying D

Page Definition File: adf/sample/view/pageDefs/DepartmentsPageDef.xml

Bindings: DepartmentId, DepartmentName, ManagerId, LocationId, First, Previous, Next, Last

Executables: variables, DepartmentsView1Iterator, taskFlow - employeestaskflowdefinition1

taskFlow - employeestaskflowdefinition1 - Property Inspector

Common

id *:	employeestaskflowdefinition1
parametersMap:	
Refresh:	ifNeeded
RefreshCondition:	
taskFlowId *:	/WEB-INF/employees-task-flow
activation:	deferred
active:	

Parameters

Task Flow Binding Parameters

id *	valu
deptId *	#{bindings.DepartmentId.inputValue}

Customization

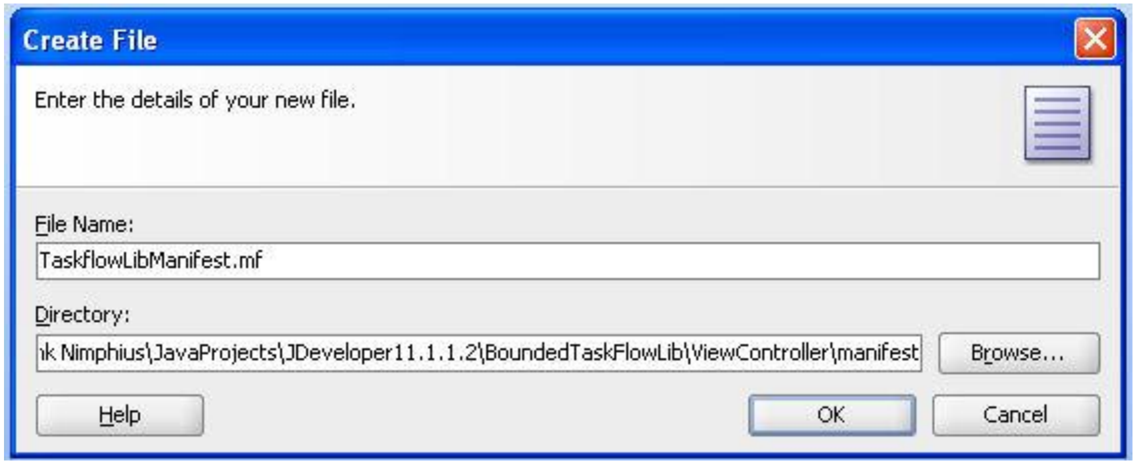
Customization allowed:	<default>
Customization allowed by:	

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 libray

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 beed to be the case and user frindly names can be defined in a custom manifest and properties file.

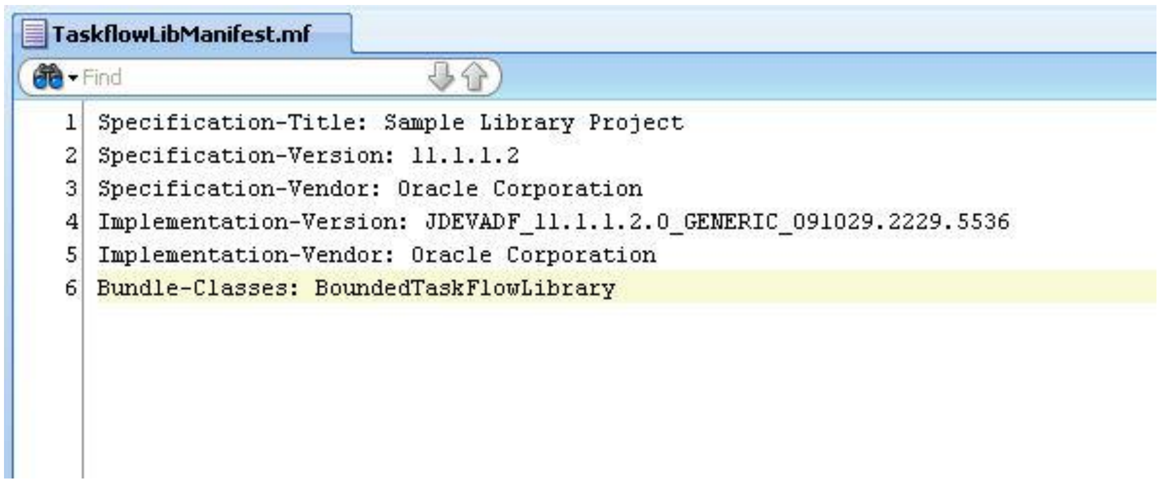
In the View Layer project of the task flow project that you created the ADF library from, create a new file choosing **New | General | File** from the JDeveloper menu. The name of the manifest is up to you, the extension though should be ".mf".



File Name: TaskflowLibManifest.mf

Directory: nk Nimphius\JavaProjects\JDeveloper11.1.1.2\BoundedTaskFlowLib\ViewController\manifest

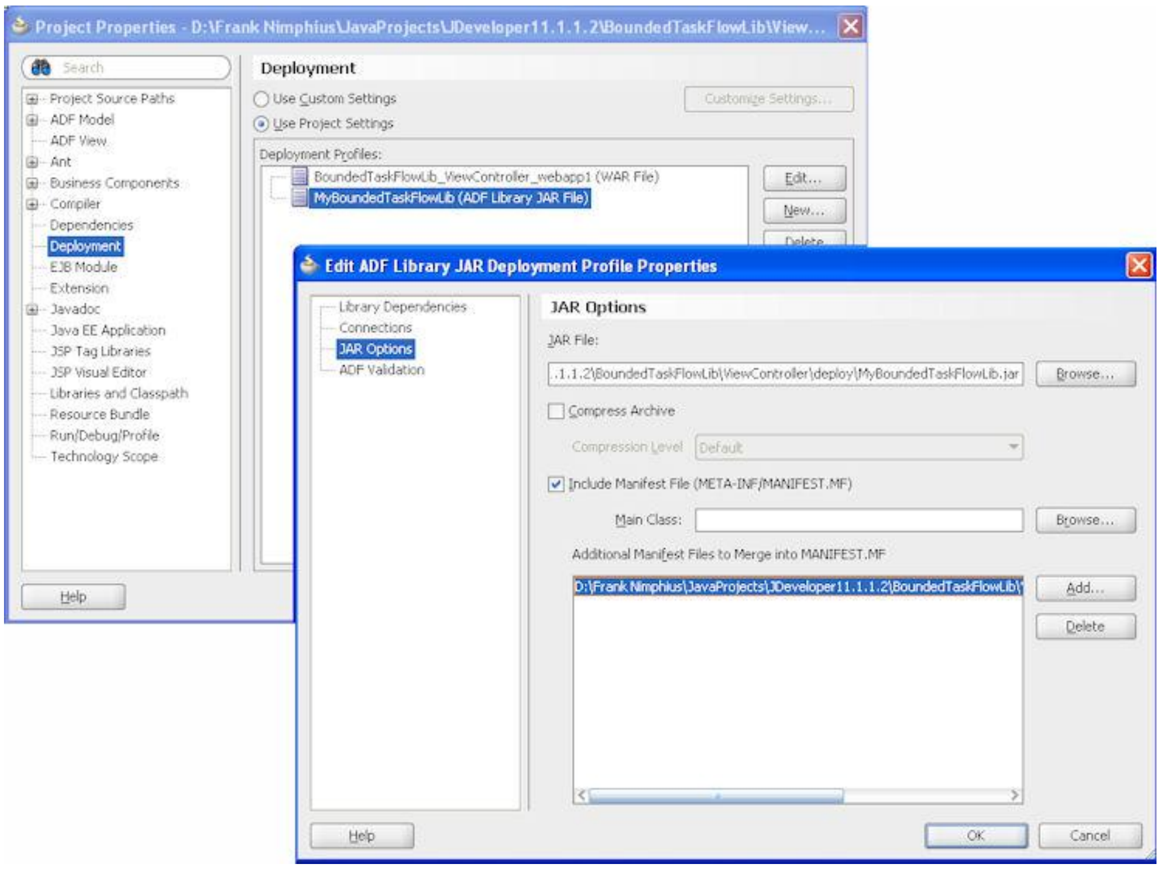
The content of the manifest has a Bundle-Classes attribute, which references a properties file. The properties file maps a task flow definition to a user friendly name, a description text and a tooltip. Optionally you can hide a task flow from showing in the ADF library, which makes sense for task flows 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.



```
Specification-Title: Sample Library Project
Specification-Version: 11.1.1.2
Specification-Vendor: Oracle Corporation
Implementation-Version: JDEVADF_11.1.1.2.0_GENERIC_091029.2229.5536
Implementation-Vendor: Oracle Corporation
Bundle-Classes: BoundedTaskFlowLibrary
```

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.



BoundedTaskFlowLib_ViewController_webapp1 (WAR File)
MyBoundedTaskFlowLib (ADF Library JAR File)

JAR File: ..1.1.2\BoundedTaskFlowLib\ViewController\deploy\MyBoundedTaskFlowLib.jar

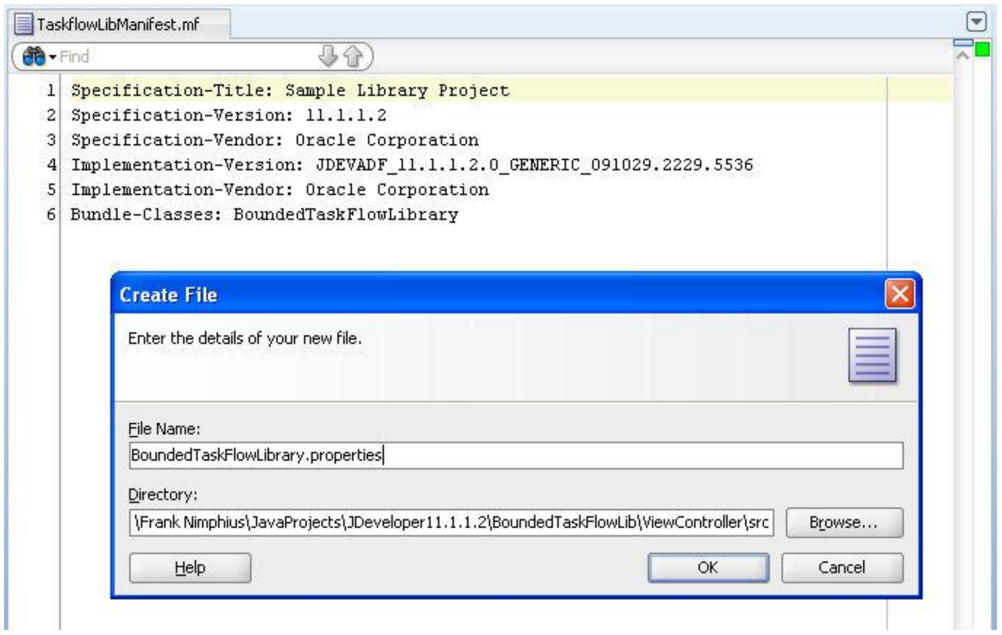
Use Project Settings [✓]

Compress Archive [ ]

Include Manifest File (META-INF/MANIFEST.MF) [✓]

Additional Manifest Files to Merge into MANIFEST.MF: D:\Frank Nimphius\JavaProjects\JDeveloper11.1.1.2\BoundedTaskFlowLib\

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.



TaskflowLibManifest.mf

```
Specification-Title: Sample Library Project
Specification-Version: 11.1.1.2
Specification-Vendor: Oracle Corporation
Implementation-Version: JDEVADF_11.1.1.2.0_GENERIC_091029.2229.5536
Implementation-Vendor: Oracle Corporation
Bundle-Classes: BoundedTaskFlowLibrary
```

Create File

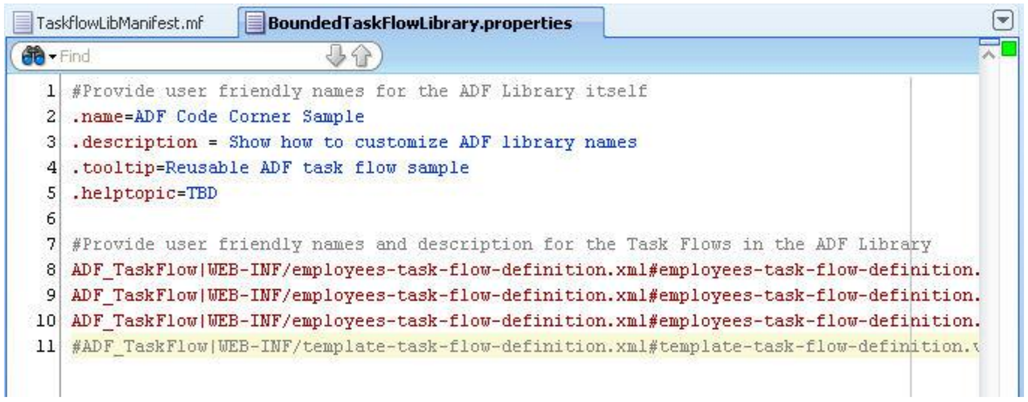
Enter the details of your new file.

File Name: BoundedTaskFlowLibrary.properties

Directory: \Frank Nimphius\JavaProjects\JDeveloper11.1.1.2\BoundedTaskFlowLib\ViewController\src

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 absoute 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.



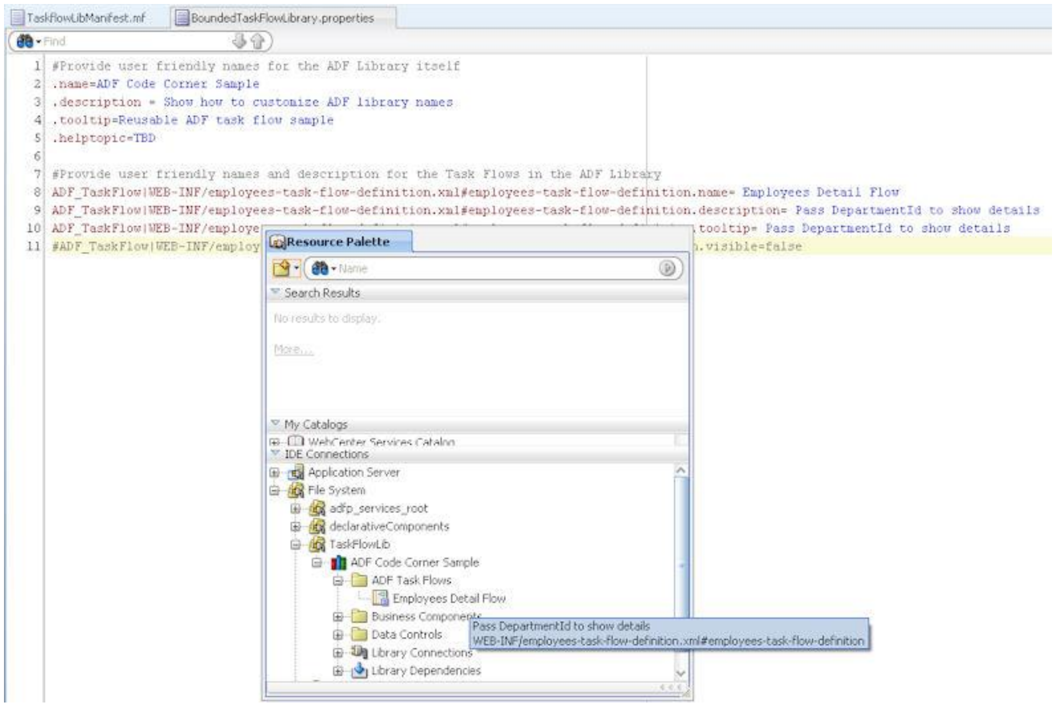
TaskflowLibManifest.mf | BoundedTaskFlowLibrary.properties

```
#Provide user friendly names for the ADF Library itself
.name=ADF Code Corner Sample
.description = Show how to customize ADF library names
.tooltip=Reusable ADF task flow sample
.helptopic=TBD

#Provide user friendly names and description for the Task Flows in the ADF Library
ADF_TaskFlow|WEB-INF/employees-task-flow-definition.xml#employees-task-flow-definition.
ADF_TaskFlow|WEB-INF/employees-task-flow-definition.xml#employees-task-flow-definition.
ADF_TaskFlow|WEB-INF/employees-task-flow-definition.xml#employees-task-flow-definition.
#ADF_TaskFlow|WEB-INF/template-task-flow-definition.xml#template-task-flow-definition.
```

The image below shows a properties file that defines a user friendly name, a description and tooltip for the conained 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 diaplayed for the imported ADF library. Notice the tooltip description.



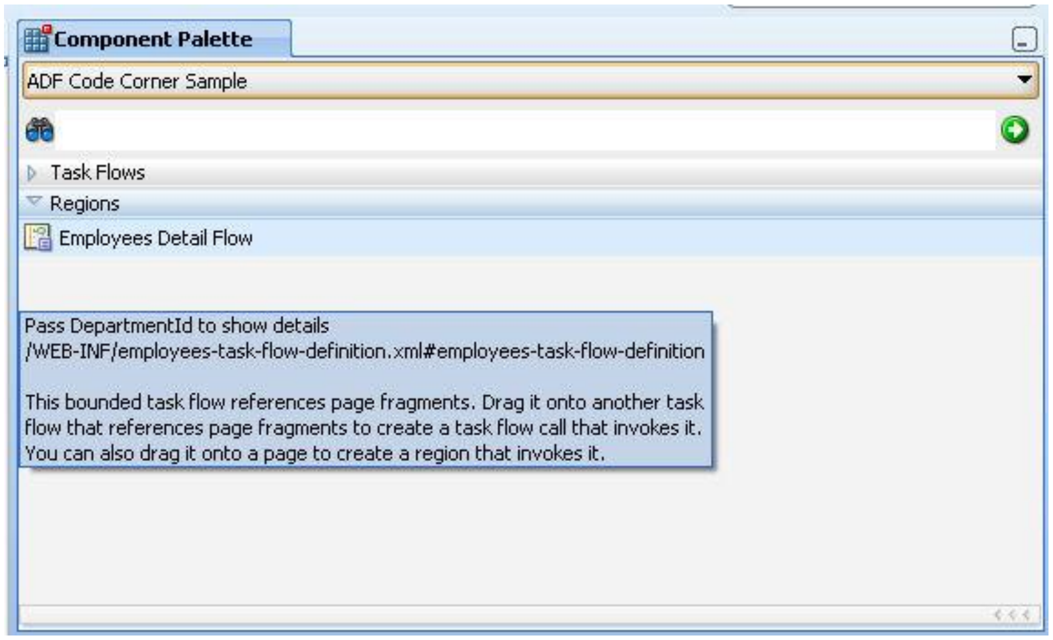
```
#Provide user friendly names for the ADF Library itself
.name=ADF Code Corner Sample
.description = Show how to customize ADF library names
.tooltip=Reusable ADF task flow sample
.helptopic=TBD

#Provide user friendly names and description for the Task Flows in the ADF Library
ADF_TaskFlow|WEB-INF/employees-task-flow-definition.xml#employees-task-flow-definition.name= Employees Detail Flow
ADF_TaskFlow|WEB-INF/employees-task-flow-definition.xml#employees-task-flow-definition.description= Pass DepartmentId to show details
ADF_TaskFlow|WEB-INF/employe                                              .tooltip= Pass DepartmentId to show details
#ADF_TaskFlow|WEB-INF/employ                                              n.visible=false
```

Pass DepartmentId to show details
WEB-INF/employees-task-flow-definition.xml#employees-task-flow-definition

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 iteslf also is shown by its name in the properties file.



Pass DepartmentId to show details
/WEB-INF/employees-task-flow-definition.xml#employees-task-flow-definition

This bounded task flow references page fragments. Drag it onto another task flow that references page fragments to create a task flow call that invokes it. You can also drag it onto a page to create a region that invokes it.

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:

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

RELATED DOCOMENTATION

⊠	Oracle® Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework11g Release 1 (chapter 14 – 19) http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/toc.htm
⊠	ADF Library documentation - http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/reusing_components.htm#BABCHHHJ
⊠	