10 Workflow UI with ADF

Introduction

Note: The solution for this chapter can be found in `c:\bpm\solutions\10-workflow-ui.zip`

This lab focuses on providing you hands-on experience with creating highly user-friendly and rich user interface for your workflow tasks.

Oracle Fusion Middleware provides Application Development Framework, or ADF, for creating rich internet applications. Oracle BPM Suite fully leverages that framework for its workflow application. Naturally, task forms associated with a task are also ADF pages. These task forms are user defined ADF pages created as part of a task flow and registered with the workflow application so that when the user navigates to the task, the appropriate task form is displayed alongside the task.

In this lab you implement user interfaces for three task flows:

1. Forms for entering a quote
2. Forms for reviewing the quote
3. Forms for approving the deal

Although you can use the Create Form wizard from the human task definition to create task forms, you are limited to a default task-flow with a single page. Of course, you can then add more pages to the flow, but these additional pages will have to be created using the standard ADF page designer only. For this lab, you are creating moderately complex UI, hence you won't be using the automatic form generation feature.

To ease the page design, you are provided with page templates that already have the basic page layout completed. You will be using these templates to create your task form pages. Make sure that you follow the instruction for each task flow for copying in the appropriate templates.

10.2 Task forms for entering a quote

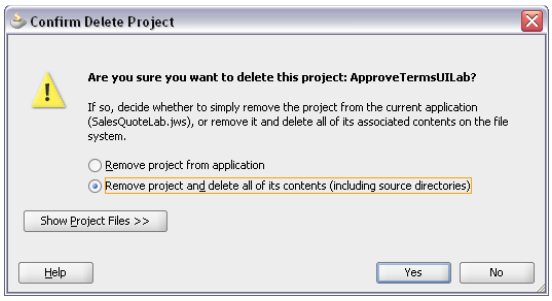
One of the first activities in the Sales Quote process is creating a quote and is initiated as a user task.

You now create the user interface for this user task for capturing the details of the quote.

10.4.1 Setup

You already have created default UI forms for all the human tasks in the process. In this lab you recreate those same forms from ground up. In order to start from a clean slate you need to delete all the UI projects from the JDeveloper application. Follow these steps for removing the default UI projects:

1. Exit JDeveloper and save your existing application by copying it to another folder or ZIPping it up. Make sure that you copy the full directory tree starting from the directory that contains the application workspace file, `SalesQuoteLab.jws`
2. Start JDeveloper, open the Application Navigator, right-click the **ApproveDealUILab** UI project, and select **Delete Project**.
3. In the **Confirm Delete Project** dialog window, select **Remove project and delete all of its contents**

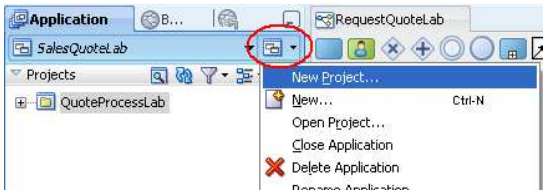


4. Repeat this for these two UI projects:
   - BusinessPracticesReviewUILab
   - EnterQuoteUILab
5. Do not delete these two UI projects:
   - ApproveTermsUILab

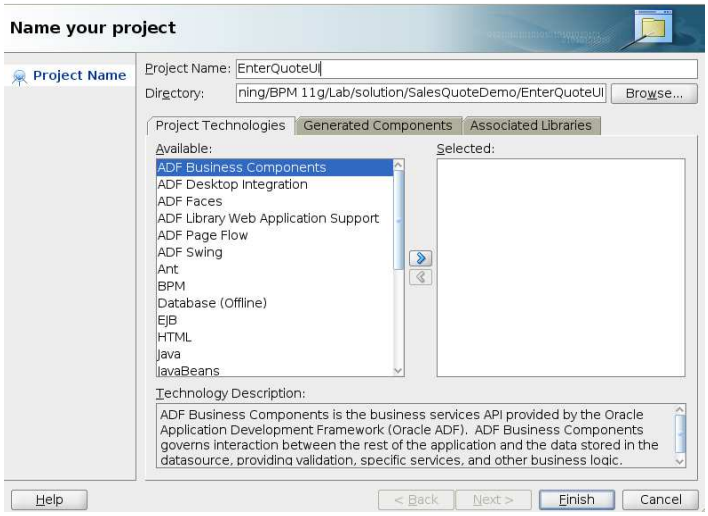
- FinalizeContractsUILab

10.4.2 Create a new UI project

1. Create a new project from the application menu



2. In the **New Gallery** window select **Generic Project** and click **OK**
3. The project dialog opens. Name your project **EnterQuoteUILab** and click **Finish**



4. Click **Save All** to save

10.4.3 Create Business Components

To create a sales quote you will need, in addition to other data, products that you are creating the quote for. In the task form the user should be able to select products to add to the quote. Typically this list would come from an item master or price list.

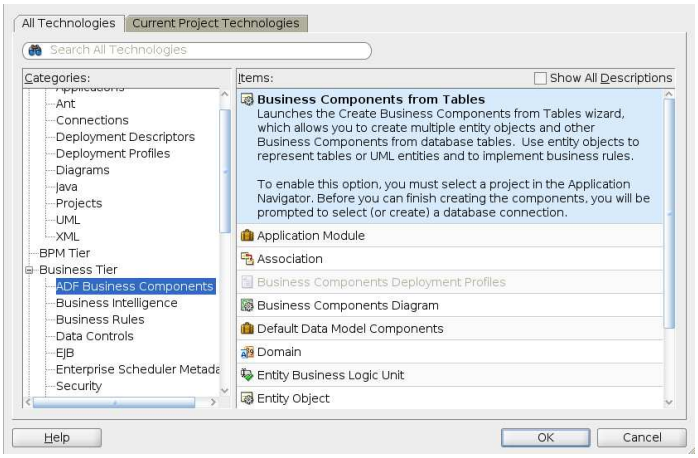
For this lab, you use ADF BC View Objects that provide this product data. You also create a second view object for providing the different terms and conditions that one can choose to add to the quote.

If you have not already created the QUOTE database schema, run the SQL scripts `c:\bpm\sql\create_user.sql` and `quote.sql`.
If you are using a pre-configured system for this training, these scripts have already been run. You can verify that the scripts have run by creating a database connection in JDeveloper for user quote and pw quote.

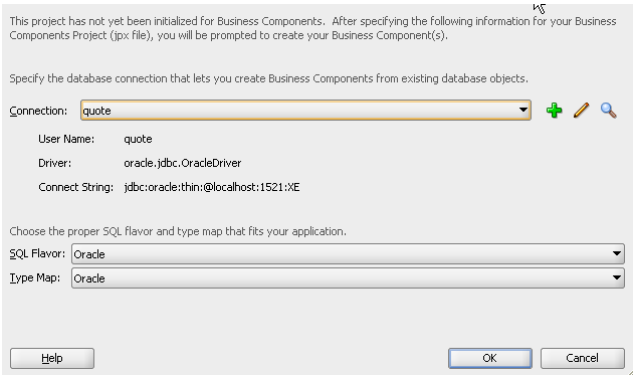
Follow these steps to create the business components for products and license terms.

1. Right-click on the **EnterQuoteUILab** project in the Application Navigator, and select **New…**

2. In the **New Gallery window**, select **ADF Business Components** in the Categories panel and **Business Components From Tables** in the **Items** panel and click **OK**.

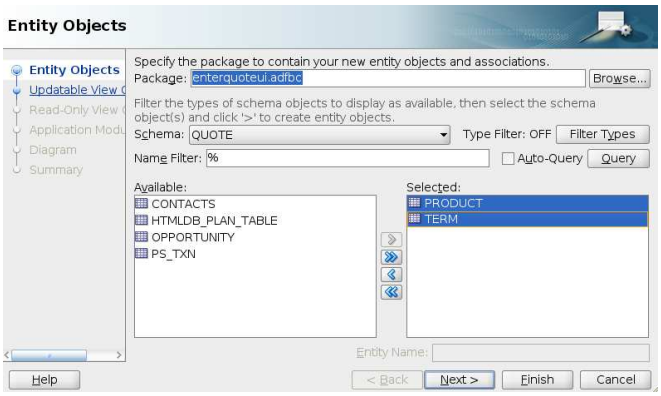


3. Create new database connection to the quote schema that you created. This connection is part of the project. You can also browse for and copy the connection if you already created one in the IDE.



4. Click **OK**

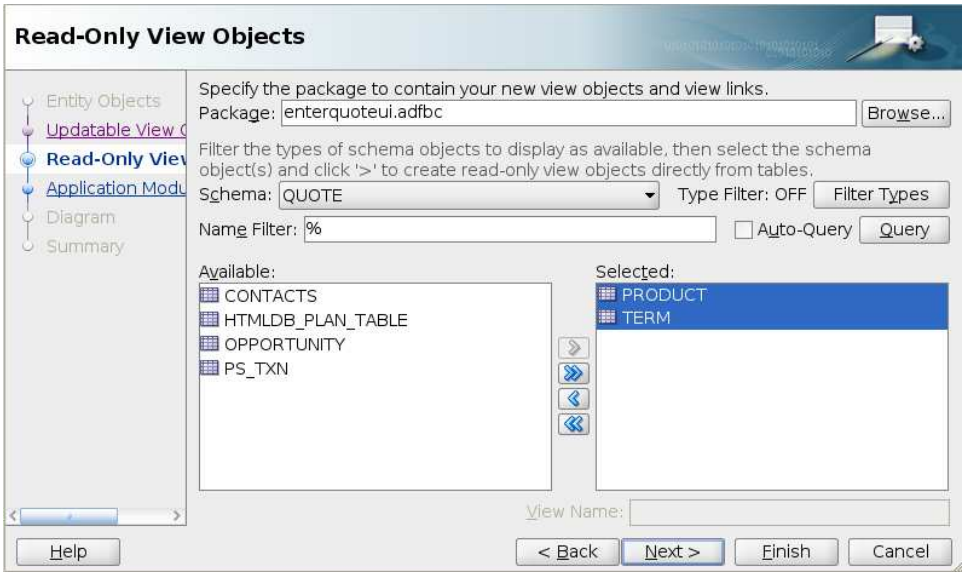
5. In the **Entity Objects** window, enter Package name as *enterquoteui.adfbc* and select **PRODUCT** and **TERM** tables. If you don't see the tables in the Available list, click on the **Query** button to query the database.



6. Click **Next**

7. Click **Next** in the **Updatable View Objects** window, without selecting any object

8. In the **Read-Only View Objects** window, click on **Query** to retrieve list of available objects and select **PRODUCT** and **TERMS** and click **Next**.



9. Click **Finish** in the **Application Module** window.

10. Click **Save All**.

You now have entity objects and corresponding view objects created for the underlying PRODUCT and TERMS table. You use these via ADF Data Controls in the task form later in this lab. Before you move on to creating the task forms, you need to update the application module configuration to use the JDBC data source for the QUOTE schema.

First, create the JDBC data source as follows.

11. Open http://localhost:7001/console to start the Web Logic Server (WLS) console and login using weblogic/welcome (replace the host and port and username/password to match your own configuration).

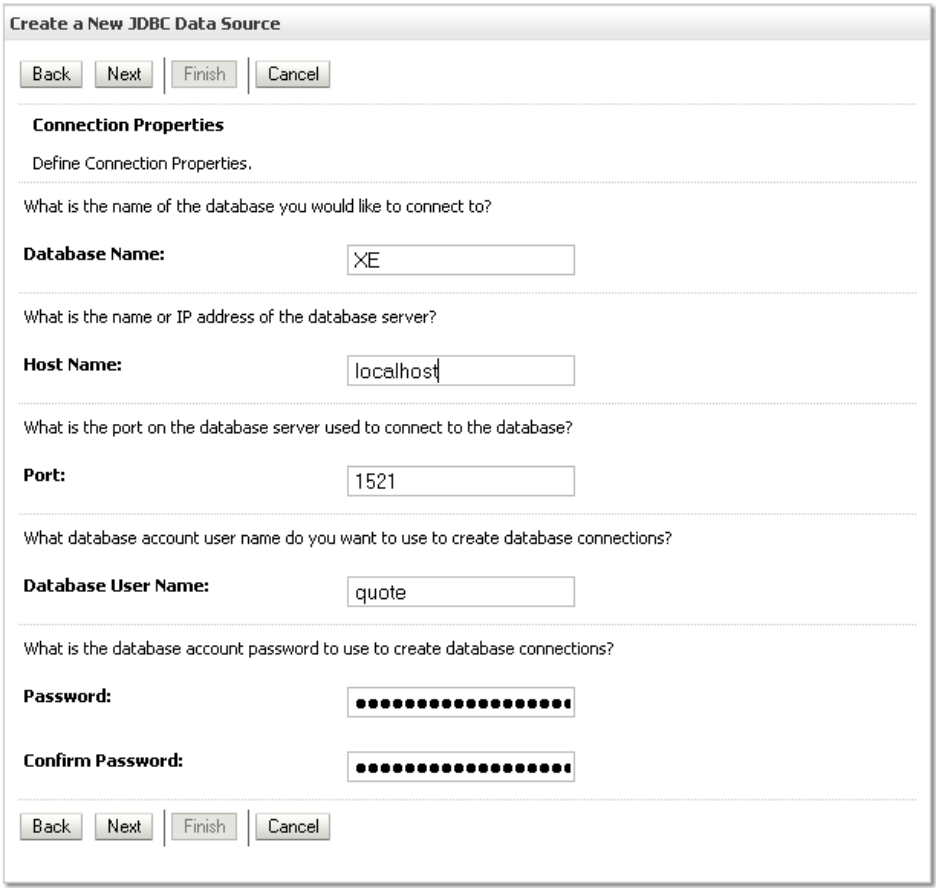
12. On the left navigation bar, click **Services > JDBC > Data Sources.**

13. At the top of the data source table, click **New**.

14. Enter the data source information
Name: *quoteDS*
JNDI Name: *jdbc/quoteDS*
Database Type: *Oracle*

15. Click **Next**, click **Next** twice more

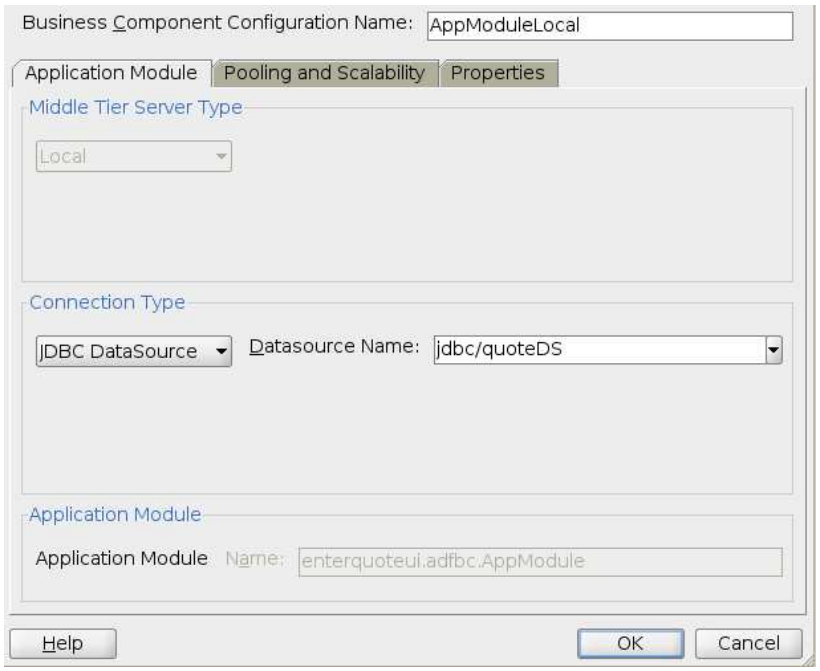
16. Enter the database information.
Database Name: XE (your database SID)
Host name: localhost (host where your database is running)
Port: 1521 (set according to your configuration)
Database user name: quote (created in previous section)
Database user password: quote



17. Click **Next**
18. Click **Test Configuration**. Confirm success message at top of page.
19. Click **Next**
20. Select the **Target server** where your BPM component is running: *soa_server1* or *AdminServer*
21. Click **Finish**

Now update the application module configuration.

22. Double-click on **AppModule** under **EnterQuoteUILab→Application Sources→enterquoteui.adfbc** to open the application module
23. Click on **Configurations**
24. Select **AppModuleLocal** and click on the Edit icon
25. Change in the **Connection Type** to **JDBC DataSource** and enter **Datasource Name** as *jdbc/quoteDS*.



26. Click **OK**.
27. Repeat the above step for **AppModuleShared**
28. Save All.

10.4.4 Create Task Flow

The sales quote requires following information to be captured from the user:

- Header-level information like customer address, contact information, type of customer, sales representative associated with this deal, industry type etc.
- List of products for which the quote is being prepared
- Information any discount being given. The discount could be different for each selected product
- Any terms and conditions that need to be added to the quote

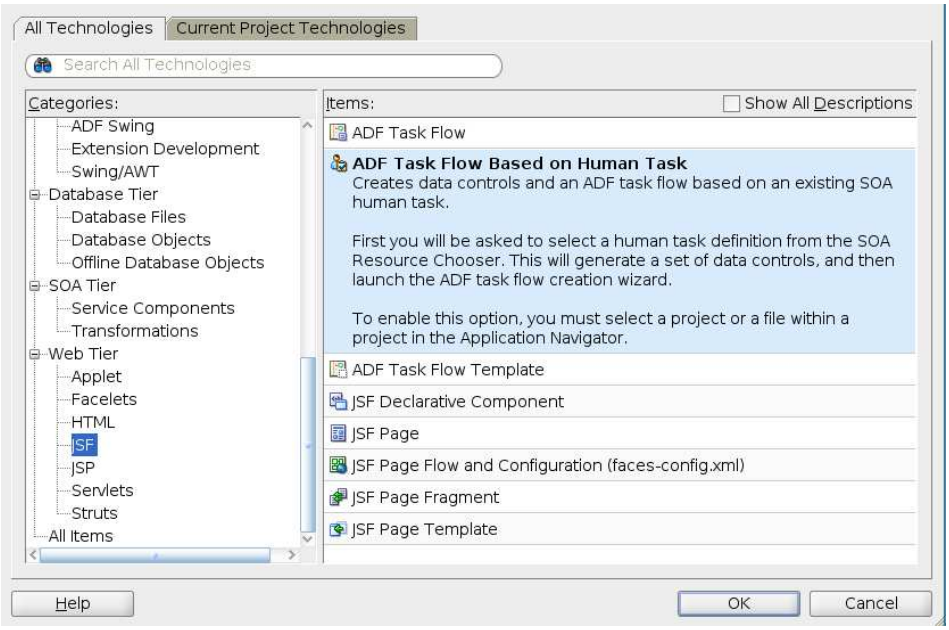
Rather than capturing all this information in a single page, you can design four separate pages, each capturing one type of information. Completion of one page will take the user to the next until the quote is completed and ready for submission for approval.

This can be done by creating a **Bounded Task-flow** with four **Views** arranged in a proper sequence.

To create a bounded task-flow that will be rendered as part of the Oracle BPM Worklist application, JDeveloper provides a convenient wizard for creating a new ADF task-flow based on a human task definition. The following steps will guide you in creating such a task-flow.

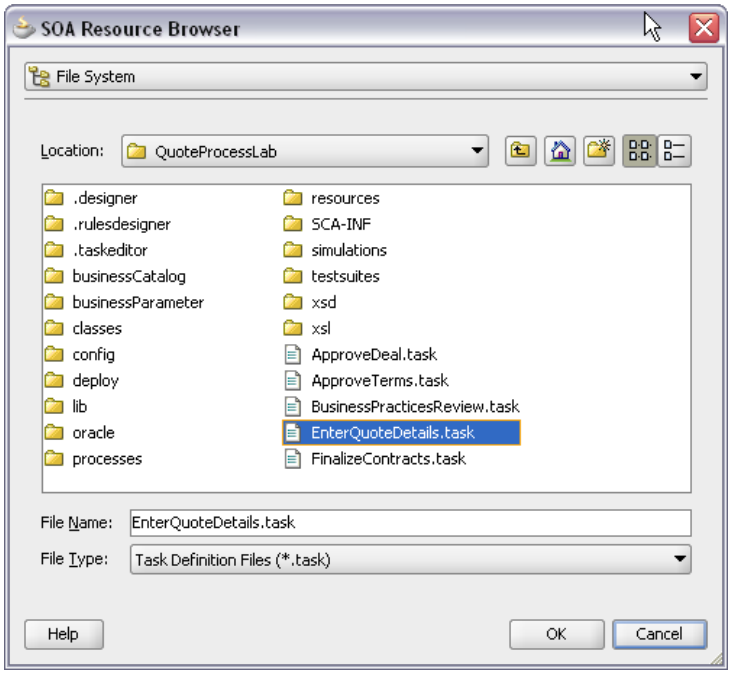
1. Right-click **EnterQuoteUILab** in the Application Navigator and select **New…**
2. In the **New Gallery** window, switch to the **All Technologies** tab and select **JSF** under the **Web Tier** node.

3. In the **Items** panel. Select **ADF Task Flow Based on Human Task** and click **OK**.

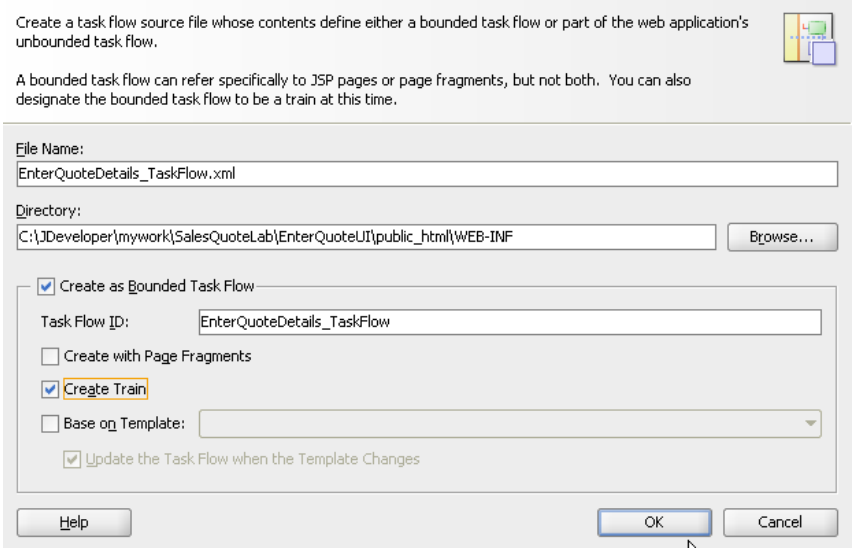


4. In the **SOA Resource Browser** window, go one directory up and double-click the **QuoteProcessLab** folder.

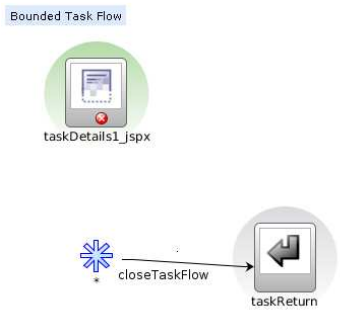
5. Select **EnterQuoteDetails.task** and click OK.



6. You should now see the **Create Task Flow** window. Select **Create Train** and click on **OK**. Selecting **Create Train** creates a special type of task flow that allows sequential flow pages which the ADF Controller manages. It also provides visual clues on the progress of the "train" represented by "train stops" which the user can click to progress through the flow.



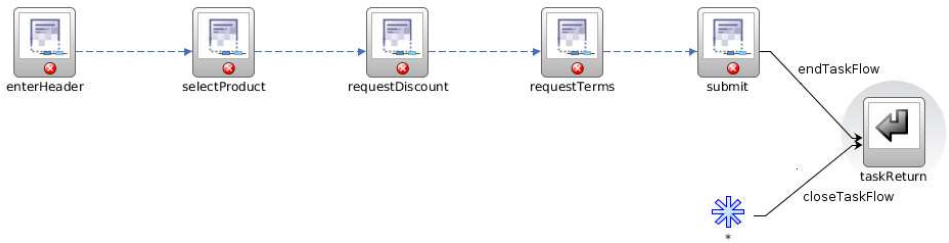
You should now see a basic flow as shown below:



7. Delete the **taskDetails1_jspx** view from the flow by right-clicking on the view icon and selecting **Delete** from the pop-up menu.
8. Drag and drop a **View** from the components palette. Name it **enterHeader.**
9. Similarly drop in a view for the remaining pages. Make sure you drop them in the right sequence and they will automatically get connected to each other in a flow. Name them **selectProduct**, **requestDiscount**, **requestTerms** and **submit**.
10. Save All.

You just created the pages for each train stop. The last page, **submit**, should return from this bounded task flow. To do that drag a **Control Flow Case** from the palette and drop it on the **submit** page. This will allow you to connect that page to **taskReturn**. The control flow can have an event associated with it. In this case, name the event **endTaskFlow**.

Your flow diagram should look like this:



11. Save All.

12. At this point you have the required task flow defined. The next step is to create pages for each of the views in the task flow.

10.4.5 Create a form for entering the quote header data

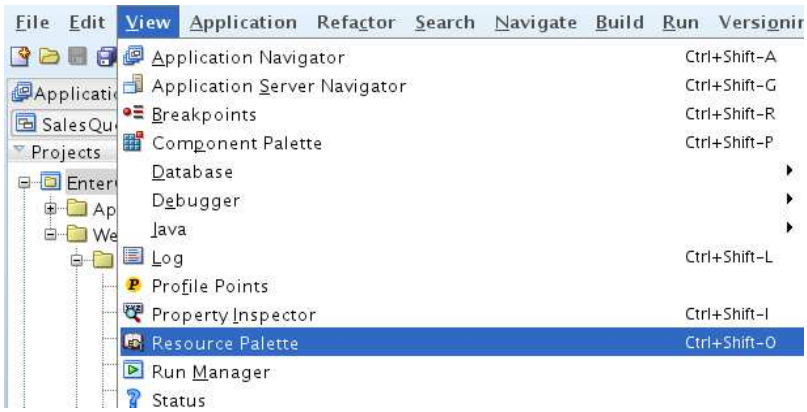
Views in the task flow are JSF pages that you design using the ADF page designer. JDeveloper provides an extensive list of UI components, layout containers and operations to create very complex pages.

Depending on the richness and complexity of the page design, the layout of the page can take some planning since there are a number of layout containers to choose from and each provides different ways to render UI components.

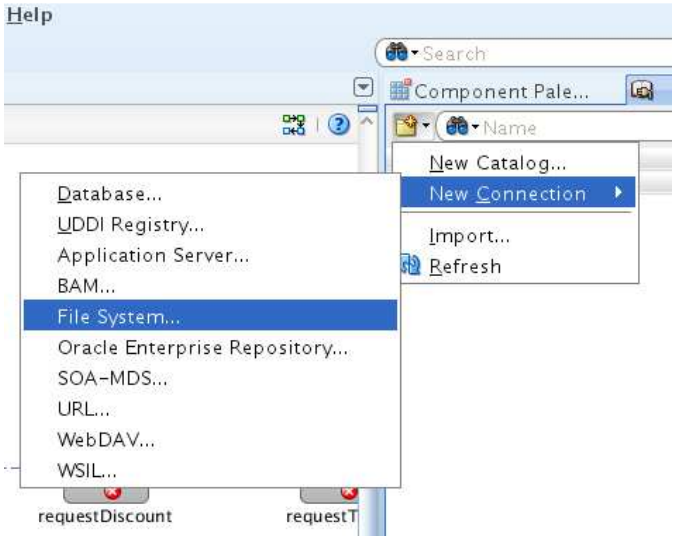
To reduce the time required to complete the lab, you are given JSF templates with basic page layout already completed. You will use these templates to build out all the required pages.

Before you begin creating the pages, you need to register the ADF Library that contains these templates into your project. Follow these steps to register the ADF Library

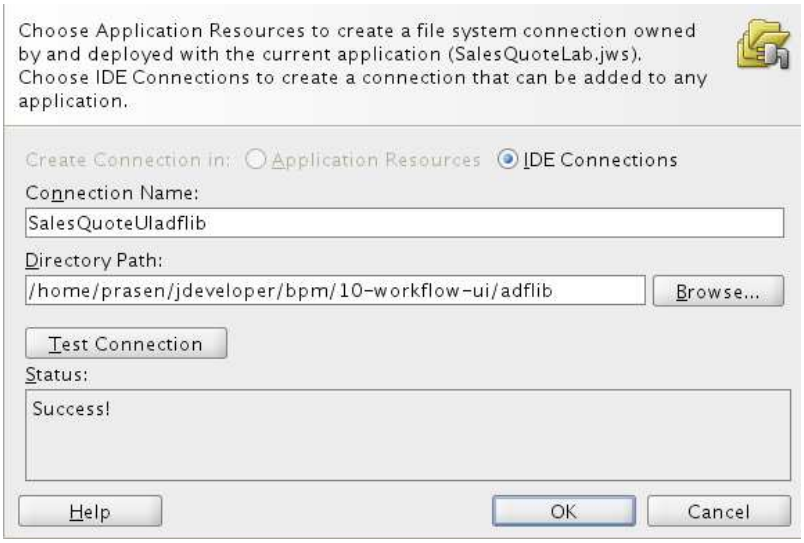
1. Open the **Resource Palette** from the **View** menu.



2. Create a new File System connection by clicking on the **New Folder** icon in the **Resource Palette** as shown.



3. Enter the connection name as *SalesQuoteUIadflib* and choose the directory **adflib**. You will find this directory at the same level as your lab docs folder

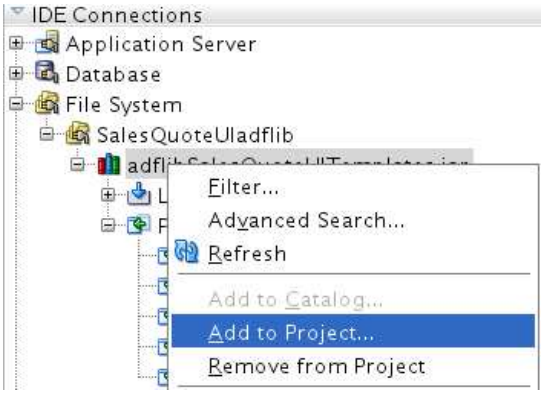


4. Click **OK**

5. Now, make sure that **EnterQuoteUILab** project is selected

6. Expand the **SalesQuoteUIadflib** folder

7. Right-click theADF library and select **Add to Project**.



8. Save All.

Now design the first page for entering the quote header information

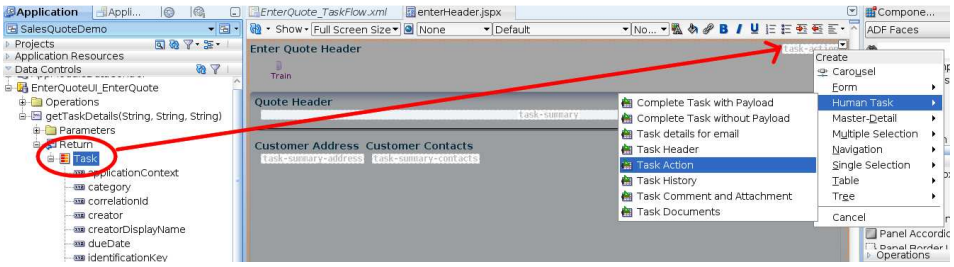
9. Open the task flow, if not already open, by double-clicking on **EnterQuoteDetails_Taskflow** under **Page Flows** in the Application Navigator.

10. Double-click the **enterHeader** view in the flow.

11. In the Create JSF Page window, select the page template **Oracle BPM 11g Training – Enter Header Template**, leave other fields as they are and click **OK**.

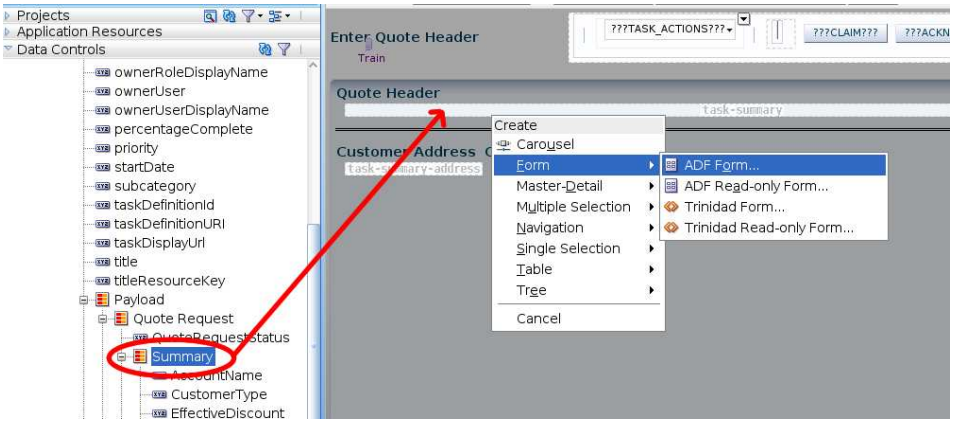
A new JSF file enterHeader.jspx is opened in the ADF designer. This template has four facets defined. Drag and drop **EnterQuoteUILab_EnterQuoteDetails** data controls from the **Data Controls** panel as explained in the following steps

12. Drag and drop the **Task** data control on the **task-action** facet. When you drop the control, a pop-up menu appears. Select **Human Task→Task Action**. Click **OK** in the two pop-ups that follow.



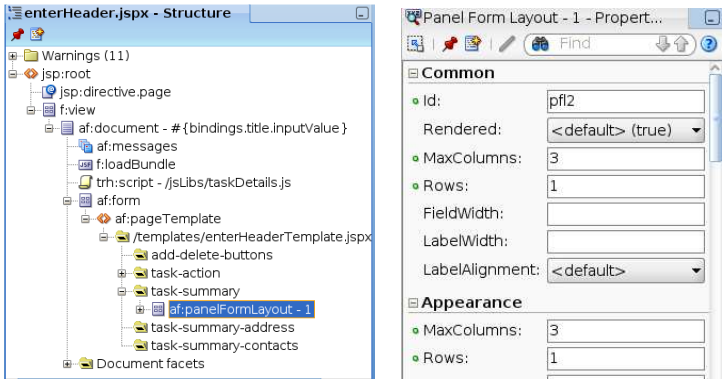
This adds the task action UI components with appropriate action and action listener setup.

13. Next, drop the **Summary** data control into the **task-summary** facet and select **Form→ADF Form**.



14. In the **Edit Form Fields** window, delete the following fields and click OK
    - NewCustomer
    - TotalNetRevenue
    - EffectiveDiscount

15. The default form is laid out in a single column. You can have the fields layout in 3 columns. To do that, select on the form to set the structure panel context and then select **af:panelFormLayout** under **task-summary** in the structure panel. In the **Property Inspector** (on the right side), set the **MaxColumns** to 3 and **Rows** to 1.



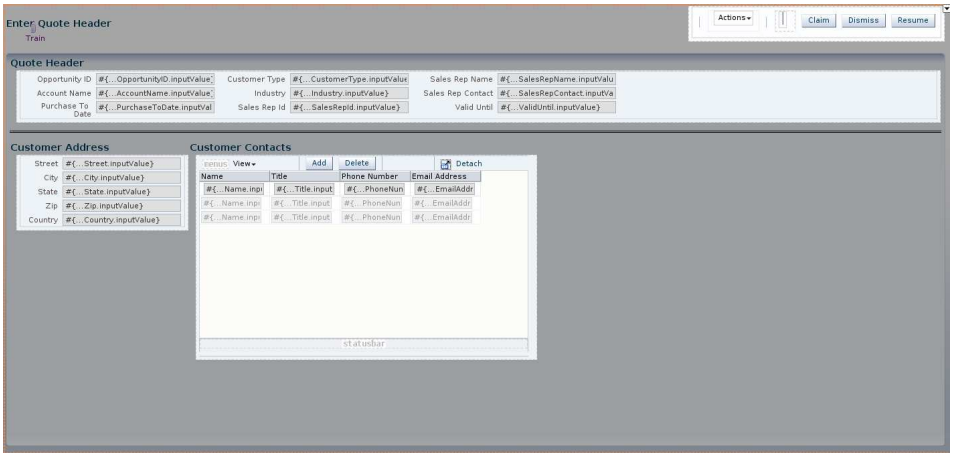
16. Drag and drop the **Address** data control from under the **Summary** data control to the **task-summary-address** facet.
17. Choose **Form→ADF Form** and then click **OK** in the **Edit Form Fields** window.

Now add the **Contacts** data control to the page. This time you want to create a table so that the user can add multiple contacts.

18. Before you drop the control onto the page, drop a **Panel Collection** from the **Layout** section of the **Component Palette** to the **task-summary-contacts** facet. A **Panel Collection** layout container is useful for rendering collection types like tables. Drag and drop **Contacts** data control to the new panel just added and choose **Table→ADF Table**. Click **OK** on the **Edit Form Fields** window.

Your first form is mostly complete, and just needs a few more additions. Your contacts table needs a couple of buttons to add and delete contacts. The next few steps show you how to do that.

19. Add a **Toolbar** from the components palette to the **toolbar** facet.
20. Expand the Contacts data control, and do the same for the Operations folder within it. Drag and drop the **Create** and **Delete** operations to the toolbar. Choose the **ADF Toolbar Button** as the UI component to use for these operations.
21. Select the **CreateInsert** button you just created and change the **Text** property in the **Property Inspector** to **Add**.
22. Your form is now ready and should look like this:



10.4.6 Create a form for adding products to the quote

Now create a form for selecting products for the quote. This form uses the drag-and-drop feature of ADF to drag the required product and drop it into the selected products table. For the source product list use the ADF BC view ProductView that you created earlier.

There are a number of ways one can use drag-and-drop in ADF. The simplest is dragging content or the value of an attribute and dropping it on a target. This typically requires no coding. In this case, though, you need to drag a row containing product item details and drop it to a target of a different shape. For such a drag-n-drop operation a bit of Java coding is required. Since advanced ADF Java coding is not the focus of this lab, you are given a JAR file with the required code already written.

10.2.6.1 Register Managed Beans

When building complex UI you come across instances where you may need custom sources of data for your components or may need custom handling of events generated by component in your UI. In ADF this is provided by way of Java Beans registered with the task flow called Managed Beans. Once registered, components can invoke methods on these beans as well as get or set properties exposed by the beans. In your lab you use beans for providing both, custom data sources values as well as custom function for handling drag-and-drop functionality.

To ease the development of these, you are provided with a pre-built JAR file that contains all the required Java Beans. The following steps will guide you through registering these classes as managed beans

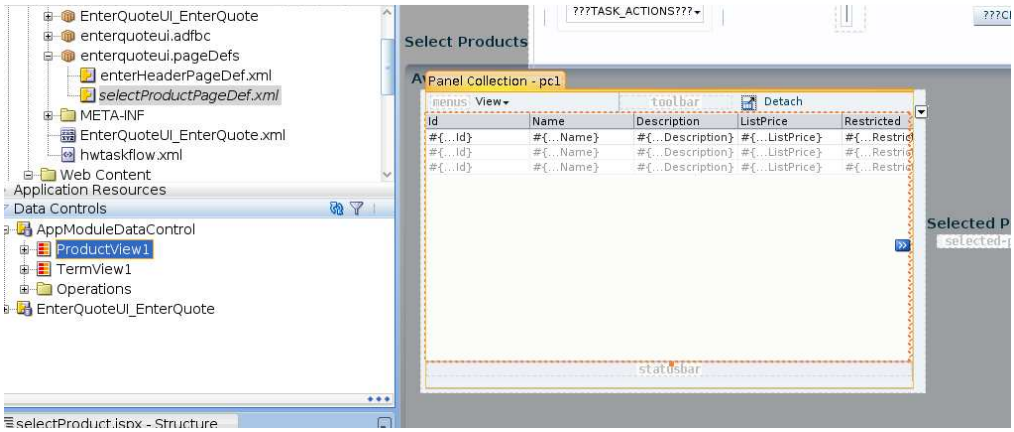
1. Open the **EnterQuoteUILab Project Properties** (right-click in the **EnterQuoteUILab** project).
2. In **Project Properties** window, click on **Libraries and Classpaths**.
3. Click on **Add JAR/Directory** and browse to c:\bpm\lib and select **bpm-training-salesquote.jar**
4. Click **Select**, click **OK**, and Save All.
5. Open the **EnterQuoteDetails_TaskFlow** task flow and switch from the **Diagram** view to the **Overview** view by clicking the tab at the bottom of the flow window.
6. Select the **Managed Beans** tab and add a new **Managed Bean** called **dropProduct**. Set the Class to *enterquoteui.backing.DropProduct* (just start typing and then select the class when it appears in the drop down). You can also browse the class hierarchy and select the class by clicking on the "v" that appears to the right of the Class field when you click on it. Set the **Scope** to **session**.
7. Similarly add the following and Save All.

Name	Class	Scope
discountHelper	enterquoteui.backing.DiscountHelper	session
termChoices	enterquoteui.backing.TermChoices	session

10.2.6.2 Build the form

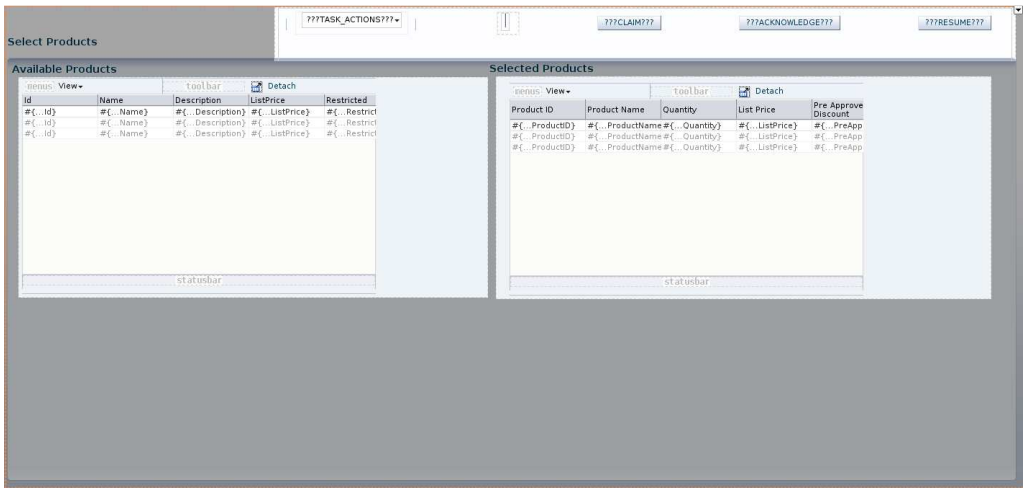
8. Open the **EnterQuoteDetails_TaskFlow** task flow diagram view (diagram tab)
9. Double-click on **selectProduct** view to create a new JSF page
10. Select **Oracle BPM 11gR1 Training – Select Product Template** as the **Page Template**
11. Click **OK**.
12. As before, create the task action buttons by dragging and dropping the **Task** data control from the **EnterQuoteUILab_EnterQuoteDetails** data control, to the task-action facet. Select **Human Task→Task Action** from the pop-up menu that appears. Click **OK** in the two pop-ups that follow.
13. Add a **Panel Collection** layout container to the **available-products** facet.
14. Drag and drop **ProductView1** data control from the **AppModuleDataControl** to the panel collection. This data control is for the ADF BC view you created at the beginning of this lab.

15. Choose **Table→ADF Read-only Table** as the UI component for this data control
16. In the **Edit Table Columns** dialog, delete the following fields
    - Category
    - ImageURL
17. Select **Row Selection** checkbox and click **OK**. You now have a table of products to select from.



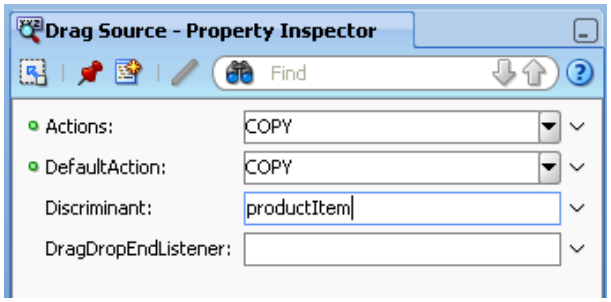
Now create the target table which will refer to the product list in the human task payload.

18. Add a **Panel Collection** to the **selected-products** facet
19. Drag and drop the **Product Item** data control from within the **Task** payload to the new panel
20. Choose **Table→ADF Read-only Table** as the UI component
21. In the **Edit Table Columns** delete the following columns:
    - RequestedDiscount
    - ControlledAvailability
22. You now have the full form complete and should look something like this:

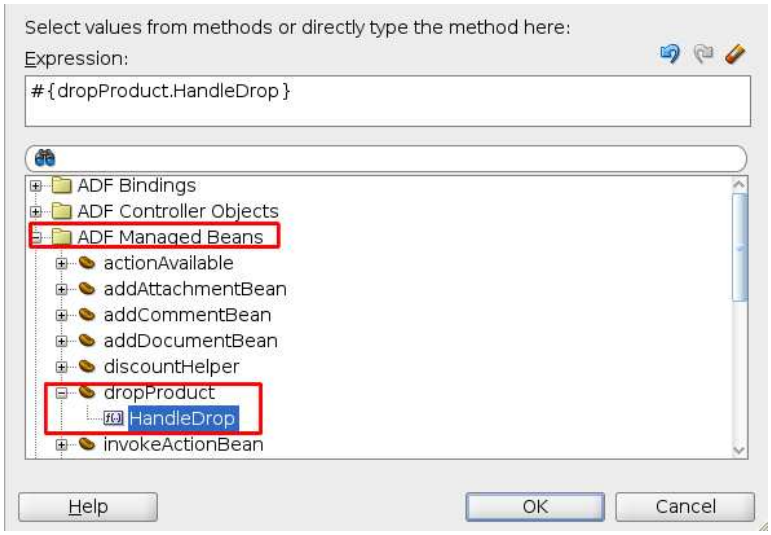


There are a few last steps you still need to complete to enable the drag and drop operation so that the user can select a product in the **Available Products** table and drop it into the **Selected Products** table.

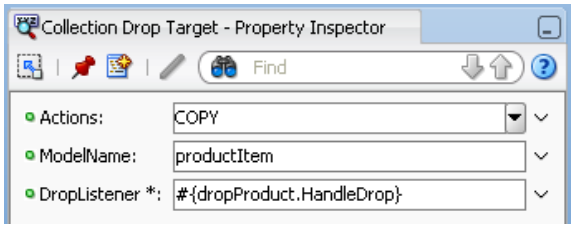
23. In the **Component Palette**, open the **Operations** section and drop **Drag Source** onto the **Available Products** table portion of the **Available Product** facet.
24. With the **Drag Source** selected in the **Structure Panel**, in the **Property Inspector**, set the **Actions** to **COPY**, the **DefaultAction** to **COPY** and the **Discriminant** to **productItem**.



25. Drop the **Collection Drop Target** to the **Selected Products** table. This will pop up a dialog for specifying a **DropListener**. Click on the **v** to the right of the field to open the expression editor and click on **Method Expression Builder**.
26. Clear any expression in the **Expression** box. Under the **ADF Managed Beans** folder, expand the **dropProduct** node and select **HandleDrop**. Click **OK** and click **OK** again to return to the designer.



27. With the **Collection Drop Target** selected in the **Structure Panel**, in the **Property Inspector**, set the **Actions** to **COPY**, and the **ModelName** to **productItem**.



28. Save All.

You now have a drag-and-drop enabled task form for selecting products for a sales quote.

10.4.7 Create a form for requesting discount

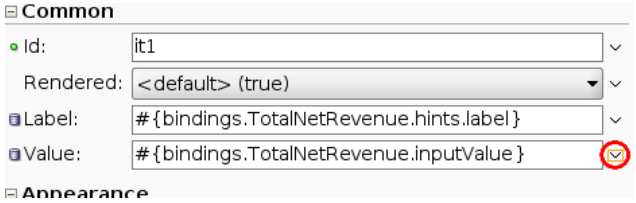
In this form you allow the user to request a discount for each product selected in the product selection form.

1. Open the task flow and double-click on **requestDiscount** view.
2. Select **Oracle BPM 11gR1 Training – Request Discount Template** for the **Page Template**
3. Click **OK** to create a new JSF page called **requestDiscount.jspx**
4. Create the task action button at the top of the page as you did for the previous pages
5. Add the **Summary** data-control as an **ADF Form** to the **summary** facet. In the **Edit Form Fields** window, delete all fields except:
   - TotalNetRevenue
   - EffectiveDiscount
6. Change the form to have the fields laid out in two columns by changing the **MaxColumns** and the **Rows** property of the form.

The effective discount and net revenue are calculated fields and should get updated as the discount is applied to each product. This requires a managed bean to do the computations. The JAR file you registered earlier already has Java bean that will do this computations. In the following steps you will use that bean as the source for these fields' values.

Note: with this change, the user can enter an integer percentage value instead of a decimal value. Once this form is in use, enter 25 instead of .25 for 25%.

7. Since these are calculated fields, make them read-only by selecting each field and changing the input field's **ReadOnly** property to **true**. This property can be found in the **Behavior** section in the property inspector.
8. Click on the **Net Revenue** field.
9. In the property inspector, click the property menu icon for the **Value** property to open the **Value** window.

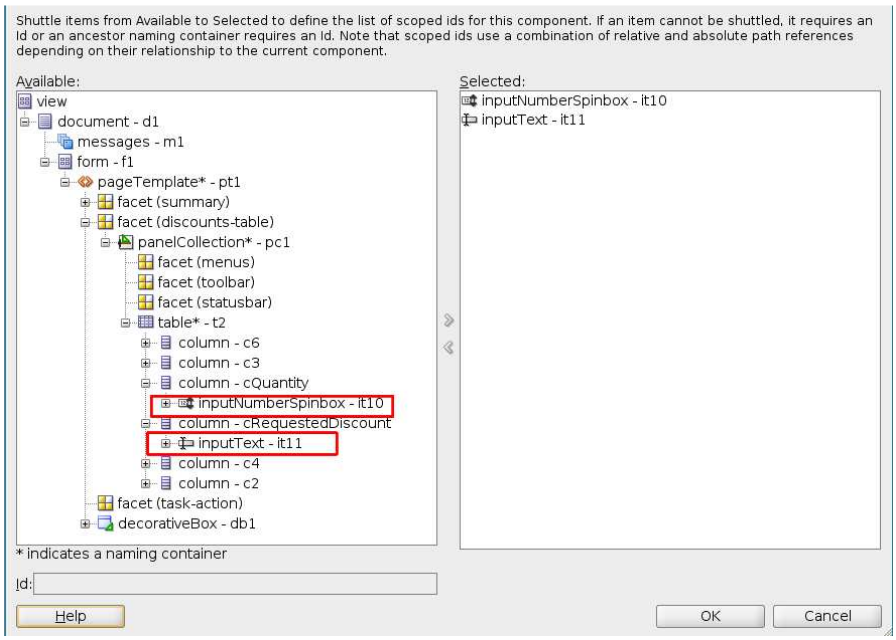


10. Click on Expression Builder to open the Expression Builder window
11. Clear the existing expression
12. Select **discountHelper→netRevenue** under ADF Managed Beans node.
13. Similarly set the value for Effective Discount by selecting **discountHelper→effectiveDiscount**.
14. Add a **Panel Collection** layout that will hold the product items table to the **discounts-table** facet.

15. Now add the selected products as an ADF table by dropping the **Product Item** data control to the new panel.
16. In the **Edit Table** window, delete the following columns:
    - RestrictedItem
    - ControlledAvailability
17. Save All.
18. Set all columns except **Quantity** and **Requested Discount** to read-only. It may be easier to use the Structure Panel to select the column's input text field.
19. Move the column **Requested Discount** to be after **Quantity**.
20. Change the component type of **Quantity** from a text input field to a number spin box so that the user can increase or decrease the value instead of typing it.
    - Right-click on **Quantity**, select **Convert To…** and then select **Input Number Spinbox**

Whenever the quantity changes or the discount changes, the fields in your Summary should also automatically change. You have already associated the appropriate properties in the discountHelper managed bean. But that value needs to be rendered every time it changes. This is achieved using ADF's partial page rendering feature. To use partial page rendering to update these values follow these steps:

21. Set the **Behavior** property **AutoSubmit** to true for both **Quantity** and **RequestedDiscount** input fields.
22. Select the **Quantity** column and change its **Id** property to **cQuantity**. Similarly change the **Id** of the **Requested Discount** column to **cRequestedDiscount**. ***Make sure you select the whole column and not the individual field.*** Use the Structure Panel if necessary.
23. Select **netRevenue** input field and edit the **PartialTriggers** property to pop up the **Edit Property: Partial Triggers** window and select the fields under the columns **cQuantity** and **cRequestedDiscount**



Note that the Ids of the fields may not be the same as in the above image.

24. Click **OK**

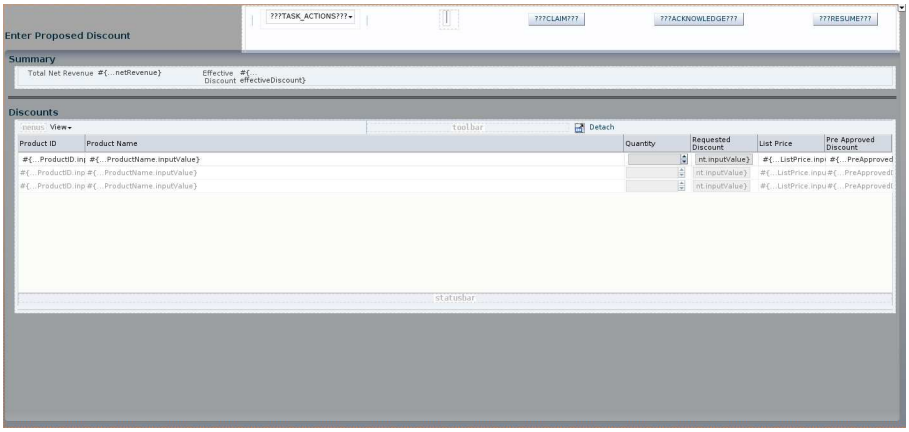
25. Repeat the same steps for the **Effective Discount** field.

Now do a bit of tidying up and you will have completed your third form. Notice that the table width doesn't fully occupy the page. To adjust that you need to:

26. Change **StyleClass** property of the panel collection that holds the table to use the style **AFStretchWidth**. You can type this style name in or select it from the **StyleClass** browser by clicking on the **v**.

27. Select the ADF Table and in the **Appearance** section of the property inspector set **ColumnStretching** property to the **Product Name** column. This change will stretch the product name column so that the table width occupies all available space.

    To make this change, first select the **Product Name** column (not the input field in the column) and note down the Id. Select this Id in the **ColumnStretching** property by first selecting the table.



28. Save All

10.4.8 Create a form for adding terms and conditions to the quote

In this form you provide the user pop-up based choices for adding pre-defined terms and conditions to the quote. As with the Product Items ADF BC data control you used earlier, you will now use the TermView ADF BC data control which will get the terms from the underlying database table.

1. Open the task flow and double-click the **requestTems** view and choose the page template as **Oracle BPM 11g Training – Request Terms Template**.

2. Add the task action buttons as you did for the earlier pages

3. Add a **Panel Collection** layout container to the business-terms facet and set the **StyleClass** property to **AFStretchWidth** as you did for the panel in the previous form.

4. Drop the **License Term** data control from the **Quote Request** to the panel collection container as an **ADF Table**.

5. Select the **Description** column and note down its Id from the **Common** properties

6. Select the table and change the **ColumnStretching** property to the Id of the Description column

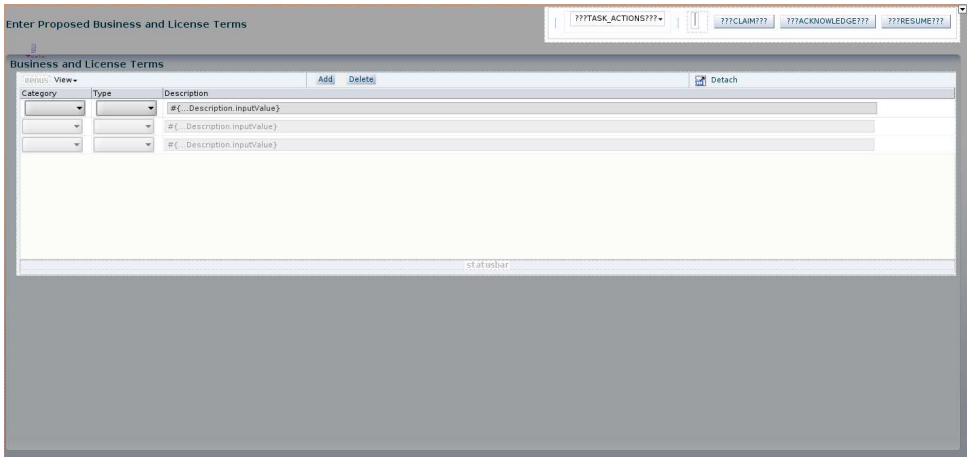
Instead of the text fields that were generated for the **Category** and **Type** fields, you want to use drop-down lists of existing term categories and types. The following steps show how to do that.

7. Drag and drop a **Select One Choice** component onto the **Category** column (make sure the full column is highlighted as you drop the component).
8. This pops up a window for selecting the source to be used for the value displayed in the drop-down list. Click on Bind… and under the **ADF Managed Beans** node select **termChoices→categoryChoices** and click **OK** and click **Next**
9. In the **Common Properties** window set the **Label** to *Category* and click **Finish**.
10. Repeat the above steps for the **Type** column, this time select **termChoices→termTypeChoices** and set the **Label** to *Type*.

You need to provide a way to add new terms or delete existing ones, so add a Toolbar to the toolbar facet.

11. Add a **Toolbar** from the components palette to the **toolbar** facet.
12. To this toolbar add **CreateInsert** and **Delete** operations as **ADF Buttons** from the **Operations** folder in the **LicenseTerm** data control
13. Change the **Text** property for **CreateInsert** button to **Add**.
14. Save all.

You should now see the completed form as show below:



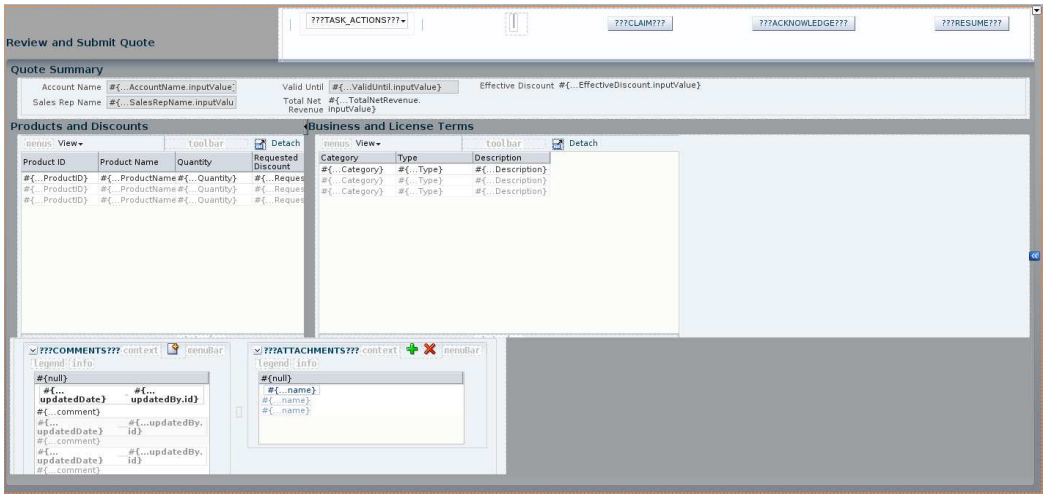
10.4.9 Create a submit form

The last form in this page flow is for reviewing the quote before submitting it for approval.

1. Open the task-flow and double-click the submit view to create a new JSF page. Select the **Oracle BPM 11g Training – Submit Template** as the page template.
2. Add the task action to the **task-action** facet
3. Add **Summary** data control from the Quote Request to the **quote-summary** facet as an **ADF Form**. Keep the following field from **the Edit Form Fields** window and delete the rest

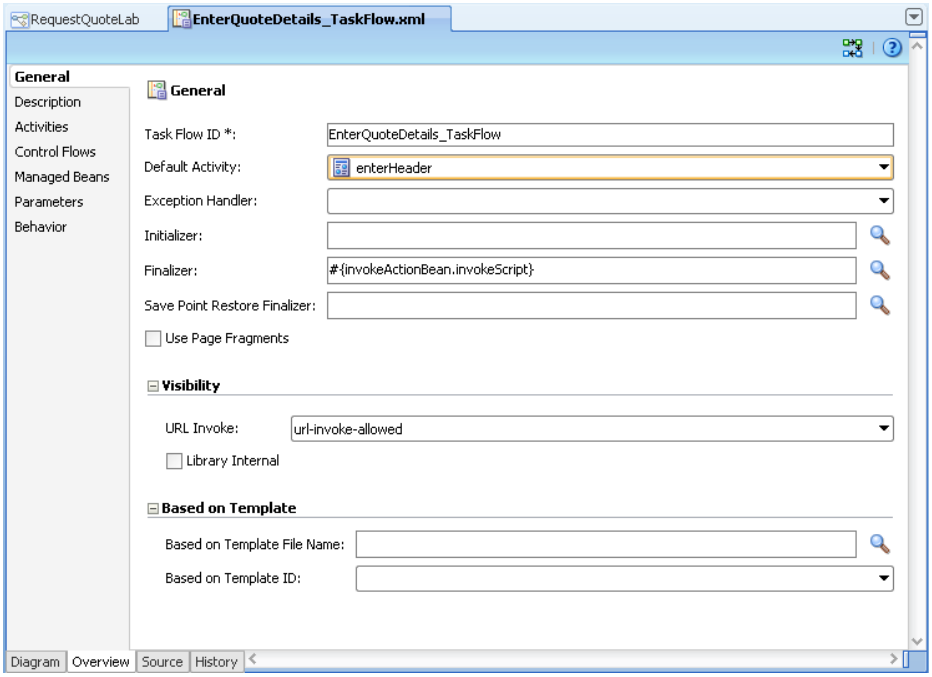
- AccountName
- SalesRepName
- TotalNetRevenue
- EffectiveDiscount
- ValidUntil

4. Set **Effective Discount** and the **TotalNetRevnue** fields to be read-only.
5. Update the form layout so that the fields are laid out in 3 columns
6. Add a **Panel Collection** to the **products-discount** facet
7. To this panel add the **Product Item** data control as a read-only **ADF Table**. Keep the following columns and delete the rest:
   - ProductId
   - Productname
   - Quantity
   - RequestedDiscount
8. Add another **Panel Collection** to the **license-terms** facet.
9. Add **License Term** data control to this panel as a read-only **ADF Table**.
10. Drag and drop the top level **Task** data control onto the **comments** facet. In the pop-up menu select **Human Task ->Task Comment And Attachment**.
11. Save All
12. Your page is now complete and should look something like this:



13. The last step is to set the default activity for this page flow. Open **EnterQuoteDetails_TaskFlow Overview** tab (tabs are at the bottom) and select the **General** tab on the left.

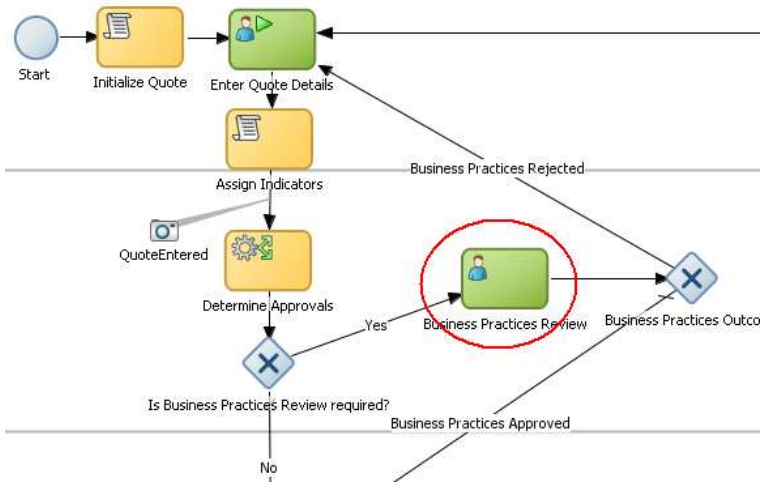
14. Select **enterHeader** for the **Default Activity**.



15. Save All.

10.3 Task form for reviewing the quote

The next activity in the Sales Quote process that requires human workflow is the Business Practices Review activity where the quote is reviewed and next task in the workflow is set. In this lab you create a form that shows the sales quote details and allows the reviewer to set the details of the next step in the business process, which is the sales quote approval activity.

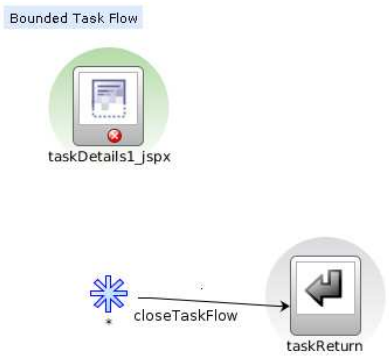


By default the work-list provides a Task History data control that allows you to set/edit the next approver in the human workflow. In this case you need to set the details of the next activity in the business process itself. Hence you need custom code that allows you to do that.

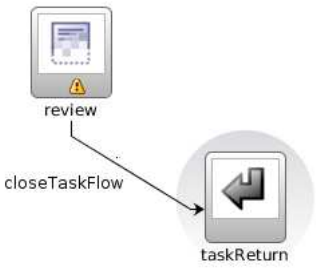
As for the EnterQuote user interface, you are provided with compiled Java code for this. Source code is also provided so that you can learn how to create such complex task flows.

10.4.10 Create Task Flow for the review task

1. Create a new **Generic Project** in the **SalesQuoteLab** application. Refer to beginning of this document for detailed steps
2. Name of the project will be **BPReviewUILab**
3. Add a new **ADF Task Flow Based on HumanTask**. Refer to steps starting from 2 under 10.4.4. Follow the steps, except for the following:
   - Select the **BusinessPracticeReview.task** for the task definition
   - Do not select **Create Train**
4. You should see a basic flow as before



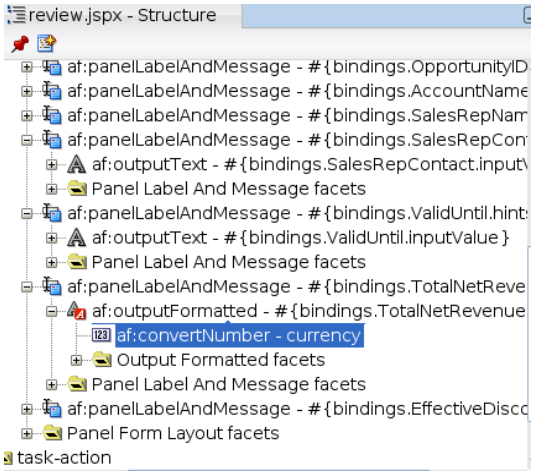
5. Delete the **taskDetails1_jspx** view from the flow by right-clicking on the view icon and selecting **Delete** from the pop-up menu.
6. Drag and drop a **View** from the components palette. Name it **review.**
7. Select the wildcard control flow indicated by the blue asterisk and delete it. With this change you are ensuring that the task flow will exit only through the **review** page.
8. Click on the **Control Flow Case** in the **Control Flow** section of the component palette and connect **review** to **taskReturn**. Name the flow **closeTaskFlow**.
9. Your flow should look like this:



10. Save All.
11. Add the templates library to this project the same way as you did for the earlier one.
    - Make sure you select the project **BPReviewUILab** in the **Application Navigator**

- In the **Resource Palette**, select the **adflibSalesQuoteUITemplates.jar**, right-click and select **Add to Project**.

12. Save All
13. Double-click on the **review** view to open the **Create JSP Page** window.
14. Select the **Oracle BPM 11gR1 Training - Business Practices Review Template**.
15. Drag and drop the **Task** data control from **BPReviewUILab_BusinessPracticesReview** to the **task-action** facet
16. Drag and drop the **Summary** data control from **BPReviewUILab_BusinessPracticesReview** to the **quote-summary** facet as a **Read-only ADF Form**. Keep the following fields and delete the rest.
    - Opportunity ID
    - AccountName
    - ValidUntil
    - TotalNetRevenue
    - EffectiveDiscount
    - SalesRepName
    - SalesRepContact
17. Change the **Rows & Columns** property of the form to layout the fields in three columns (Refer to step 13 under 10.4.5)
18. Select the **TotalNetRevenue** text field and right-click and select **Convert to…**, then select **Output Formatted**.
19. Make sure that the **TotalNetRevenue** field is selected and open the **Structure Panel**, if not already open. In the **Structure Panel**, select af:convertNumber and set its **Type** property in the property inspector to **currency**.



20. Add a **Panel Collection** to the **product-discount** facet. To this panel add **Product Item** data control as an **ADF read-only table**.
21. Add a **Panel Collection** to the **license-terms** facet and add the **License Term** data control as an **ADF read-only table** to this.

22. Save All.

23. Drag and drop the **Task** data control to the **task-history** facet. Select **Human Task→Task History** from the pop-up menu.

As mentioned at the start of this portion of the lab, this control needs to set details about the next activity in the business process and hence needs to bind to a custom Java code that uses the Worklist API to set the next activity. This is provided by a managed bean. A Java class for this available in the **bpm-training-salesquote.jar** that you registered in the earlier project. You need to do the same for this project.

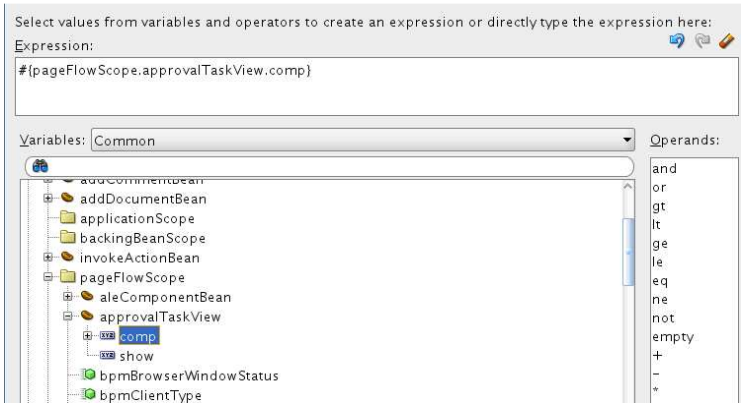
24. Select the **BPReviewUILab** project in the **Application Navigator**, right-click and select **Project Properties**.

25. Select **Libraries and Classpaths**, then **Add JAR/Directory**

26. Select the **bpm-training-salesquote.jar**, click **Select**, click **OK**, Save All.

27. Add the class **bpreviewui.backing.ApprovalTaskView** as a managed bean called **approvalTaskView**. Refer to 10.2.6.1 for steps required to register a managed bean. ***Make sure the Scope is set to pageFlow***.

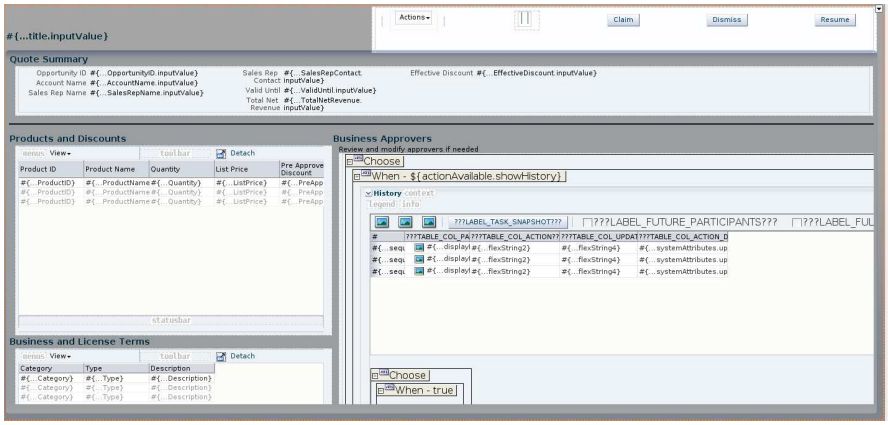
28. You need to use this bean in the **review.jspx**. Click on the task history in the page. In the **Structure Panel**, ensure that **wlc:taskHistory** is selected. In the **Property Inspector**, change the property **InitParam** to use the **approvalTaskView** managed-bean using the expression builder. You'll find it in the page flow scope.



29. Set the **Default Activity** as you did for the previous form

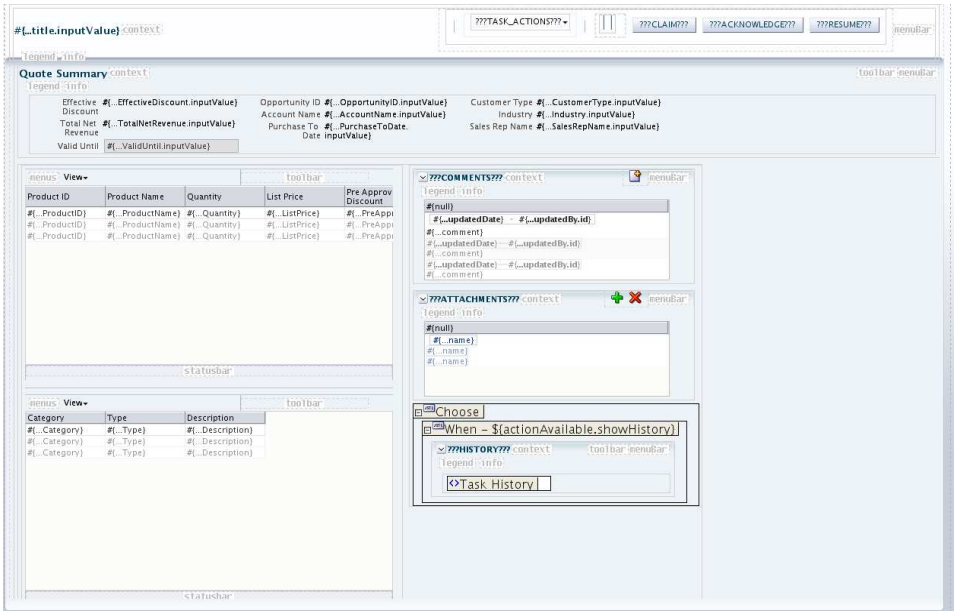
30. Save All.

This completes your review form.



10.4 Creating the UI for quote approval

This is a challenge exercise. You need to create a form that looks like this:



10.4.1 Hints to help you with the challenge exercise

The ADF TaskFlow is the same as the **BPReviewUILab**

1. Start with a **Decorative Box** layout
2. In the **top** facet of the Decorative Box, add a **Panel Header**, which will have a **toolbar facet** for the task action components
3. In the **center** facet of the Decorative Box, add a **Panel Group Layout** and set the layout to **scroll**.
4. Within this panel add a **Panel Header** for the quote summary, and two **Panel Group Layout** containers with layout set to **horizontal.**
5. In one of the containers add two **Panel Group Layout** containers for **Product Item** and **License Terms**. Both these containers should have layout set to **vertical**.
6. In the other container, add **Task Comments and Attachments** and **Task History**
7. Set the **Default Activity**

10.4.2 Using the pre-built ApproveDealUILab project

Alternatively, you can copy in a pre-built project for this form (from your version of the application that you saved in Setup step 1). Follow these steps to use a prebuilt project

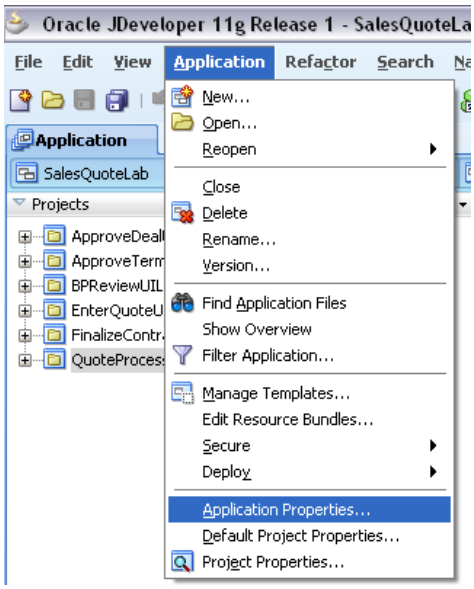
1. Copy the folder **ApproveDealUILab** to the **SalesQuoteLab** folder. It should be at the same level as the **BPReviewUILab** and the **EnterQuoteUILab** folders.
2. In JDeveloper, open the **ApproveDealUILab.jpr** using the **File→Open** menu.

10.5 Deploying the UI

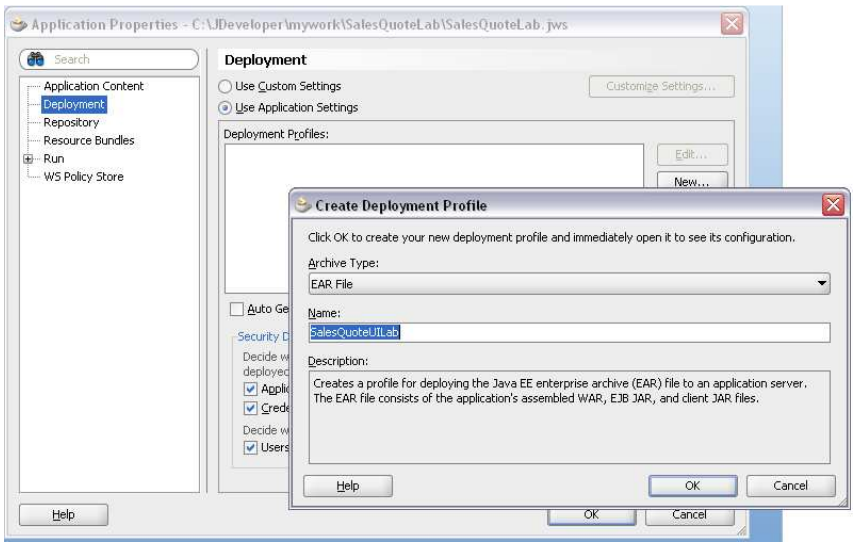
You can deploy the UI projects along with the composite as in earlier labs or you can create a single EAR and deploy it separately. This allows you to deploy the UI separately from the composite.

1. Since there is already an EAR deployed with the URL context set for these tasks, you must undeploy that one first. Undeploy it by right-clicking on the EAR and selecting undeploy. You can do this from an application server connection in JDeveloper, from the WebLogic server console, or from Enterprise Manager.

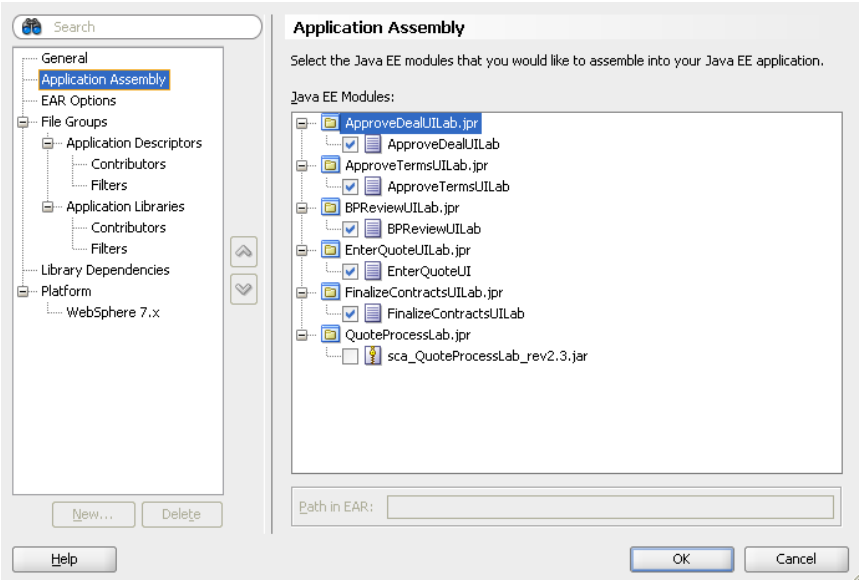
2. Create a new deployment profile for the application from the application properties menu.



3. Select **Deployment** and **New**. Name the EAR **SalesQuoteUILab**.



4. Select the UI projects in the **Application Assembly**



5. Deploy using the **Deploy** command on the **Application** menu.