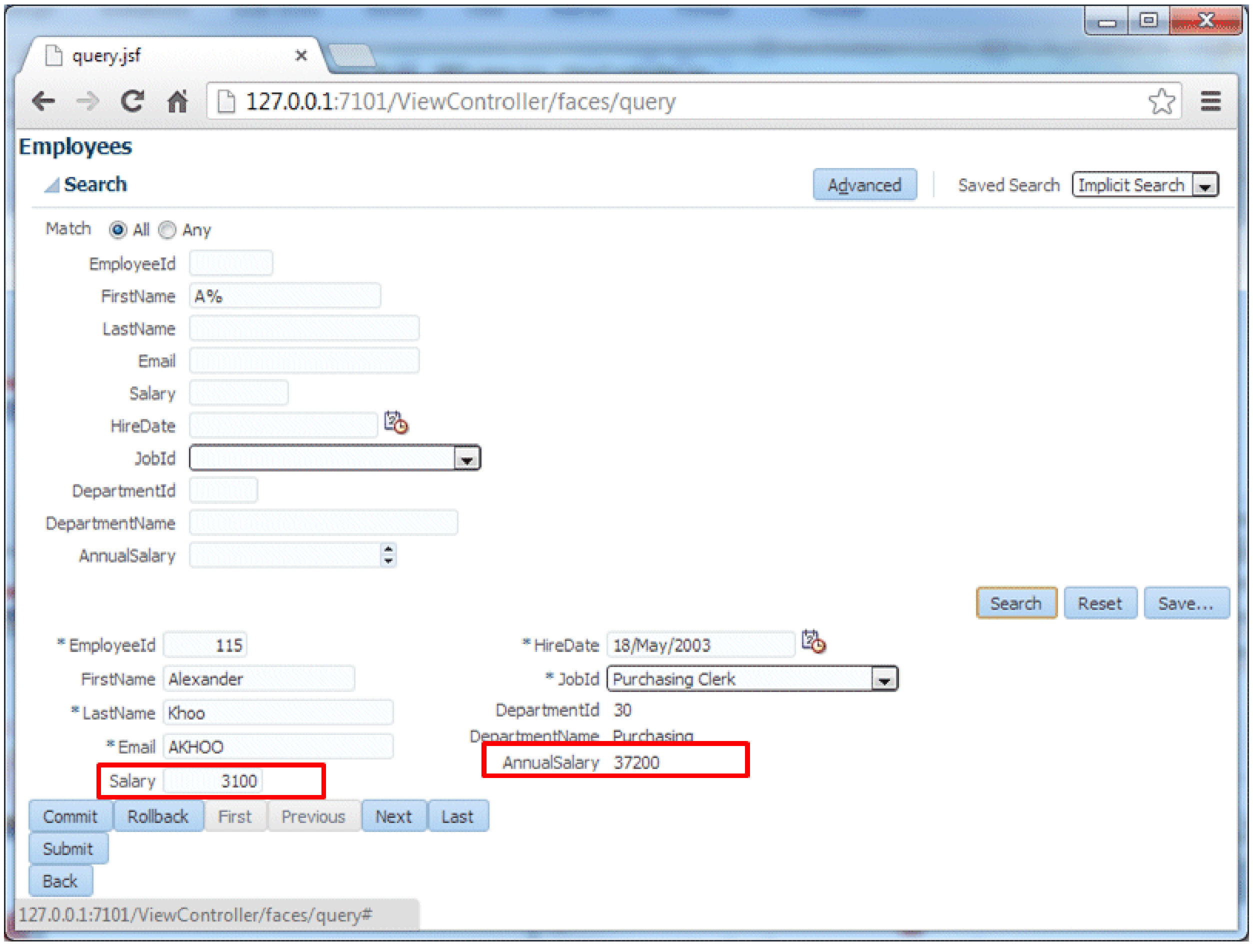
How-To Enhance User Interface in Oracle JDeveloper

Web applications usually have more than one page in them. In this part of the tutorial you add another page - a Search page - to your application and use the ADF Task Flow Diagram to define the navigation rules between the two pages. You then use features of the ADF Faces Framework to add extra functionality to the pages. Finally, you create a reusable page fragment and add it to the DeptEmpPage page.

This is the Search page for the application. Notice the list of values for the jobId and the relationship between the Salary and Annual Salary fields. In this section you'll see how to coordinate the salary values and to add an auto suggest behavior to the jobId.



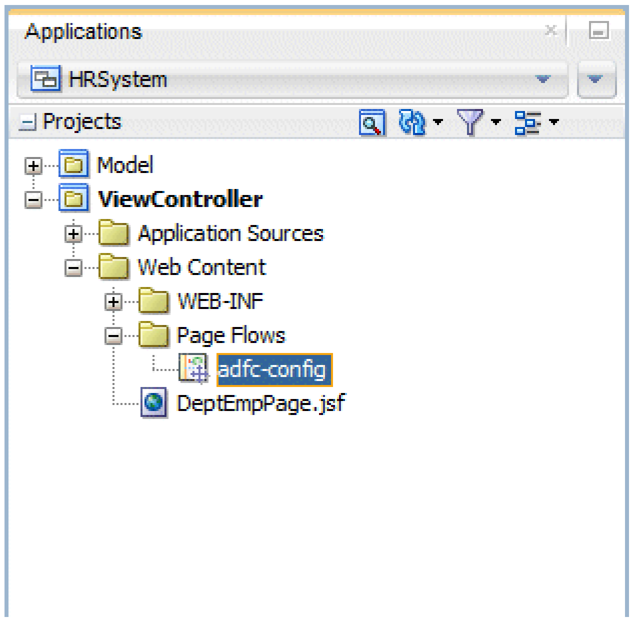
Match [✓ All] Any

EmployeeId	
FirstName	A%
LastName	
Email	
Salary	
HireDate	
JobId	
DepartmentId	
DepartmentName	
AnnualSalary	

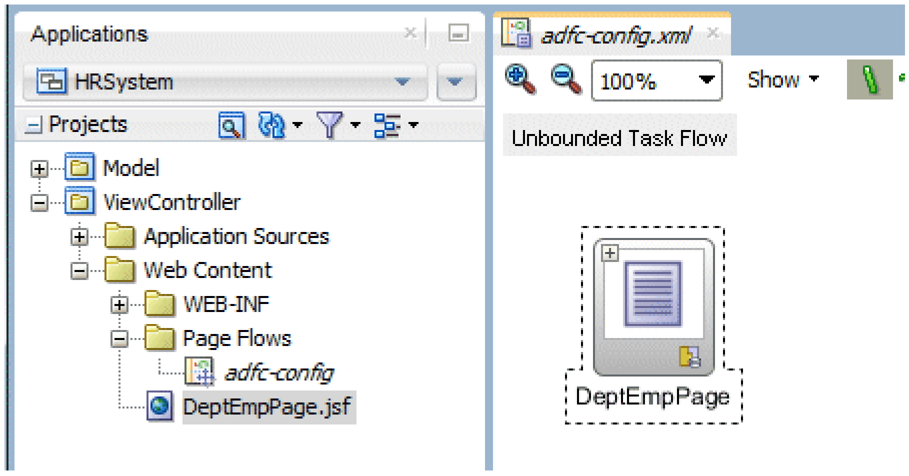
* EmployeeId	115	* HireDate	18/May/2003
FirstName	Alexander	* JobId	Purchasing Clerk
* LastName	Khoo	DepartmentId	30
* Email	AKHOO	DepartmentName	Purchasing
Salary	3100	AnnualSalary	37200

Step 1: Create a Page Flow

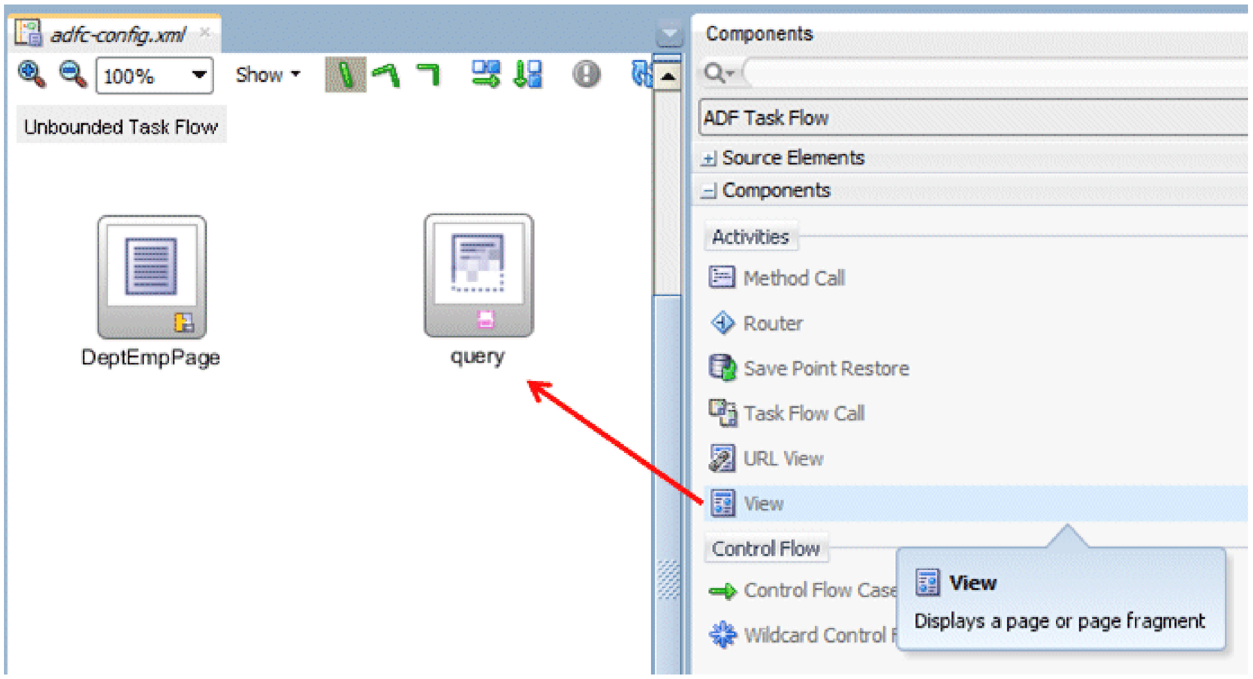
1. In the Applications window locate the **adfc-config** file under the **Page Flows** node in the **ViewController** project. Double-click it to open it in the editor. This is where you are going to define the application's navigation.



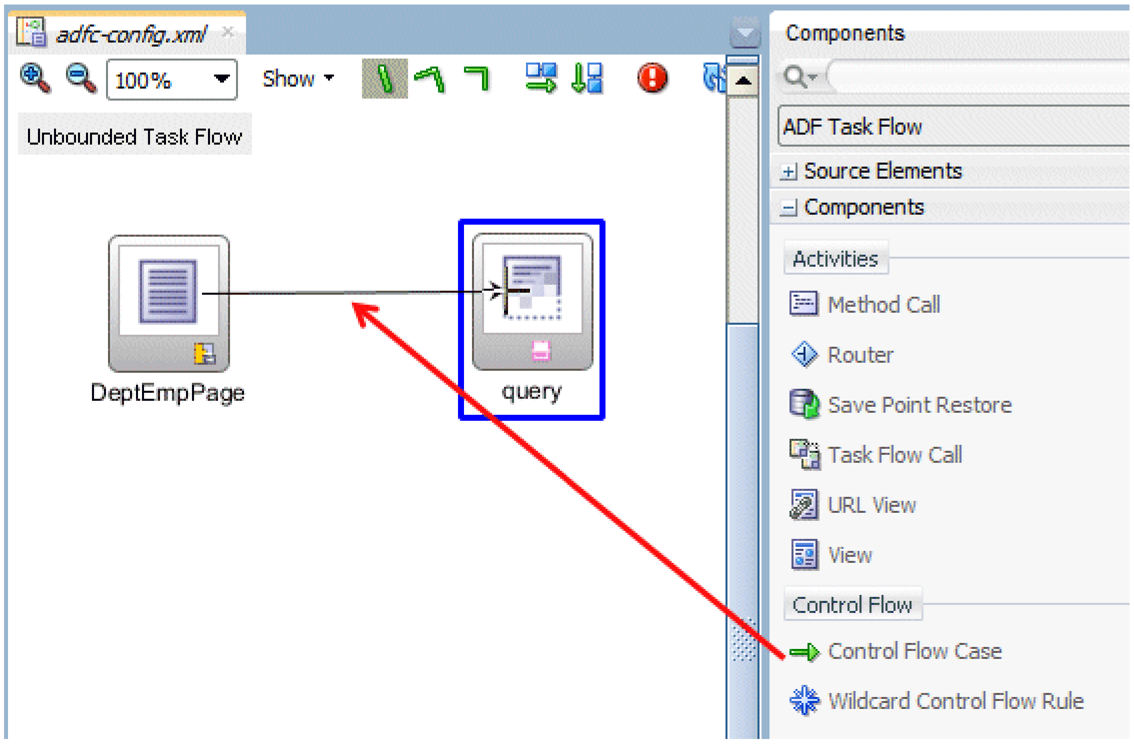
2. Drag the **DeptEmpPage.jsf** file from the Applications window into the empty **adfc-config** diagram.



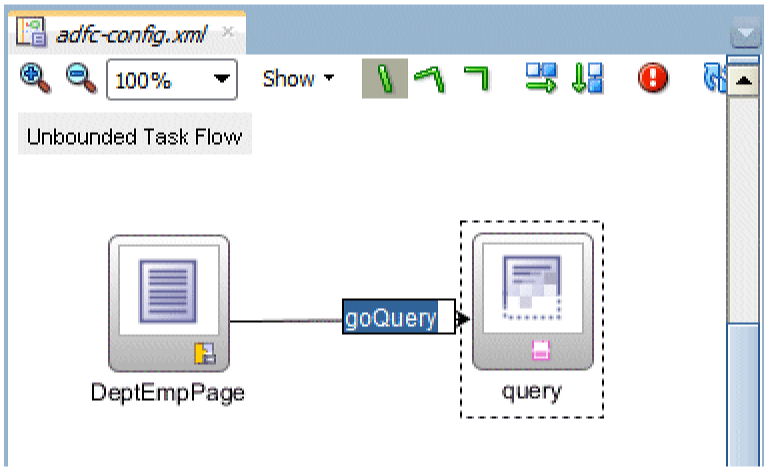
3. From the Components window drag and drop a **View** activity into the adfc-config diagram, and rename it **query**. This represents the new JSF page that you are about to create.



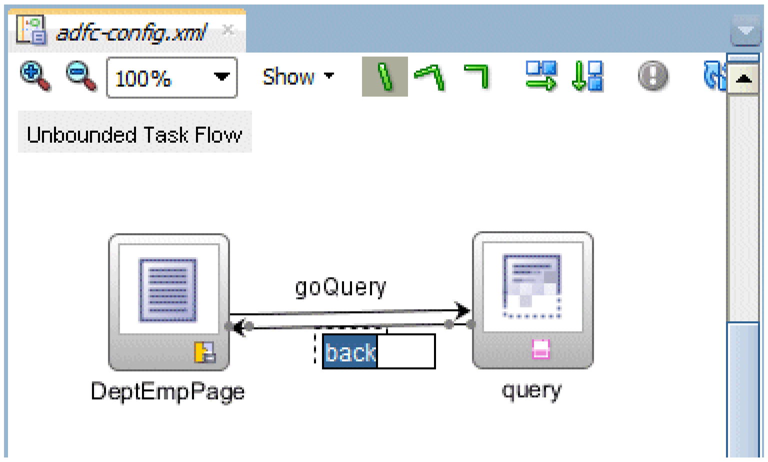
4. From the Components window click **Control Flow Case** and then click **DeptEmpPage**,and next click the **query** page.



Name this line **goQuery**.

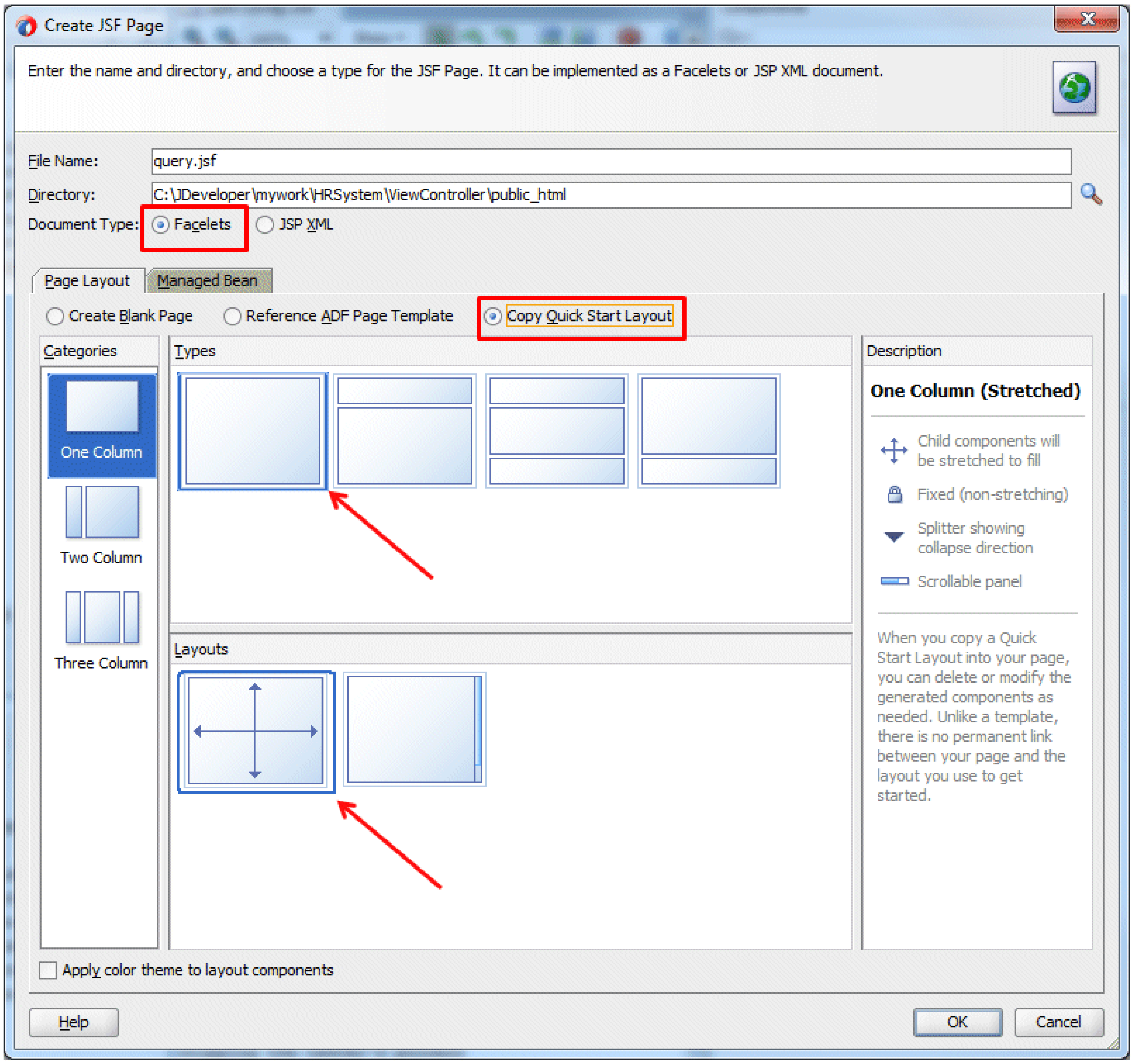


5. From the Components window choose another **Control Flow Case** and then create an opposite flow from the **query** page to the **DeptEmpPage**. Name this flow **back.**



6. Double-click the **query** activity in the diagram to create the new page. In the Create JSF Page dialog accept the default **Facelets** radio button, and click the **Quick Start Layout** radio button.

   Select the default **One Column** category, type and layout, then click **OK**.



Create JSF Page

Enter the name and directory, and choose a type for the JSF Page. It can be implemented as a Facelets or JSP XML document.

File Name: query.jsf

Directory: C:\JDeveloper\mywork\HRSystem\ViewController\public_html

Document Type: [✓ Facelets] JSP XML

[✓ Copy Quick Start Layout]

One Column (Stretched)

Child components will be stretched to fill

Fixed (non-stretching)

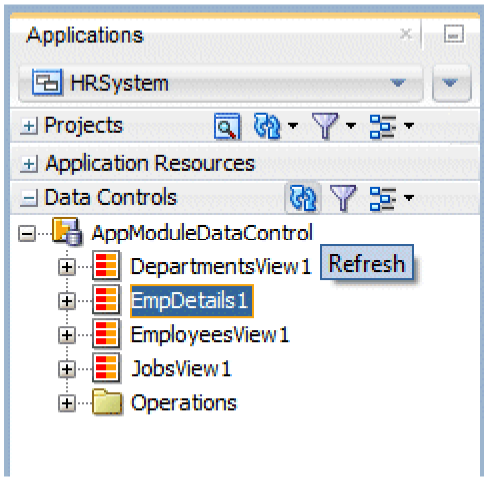
Splitter showing collapse direction

Scrollable panel

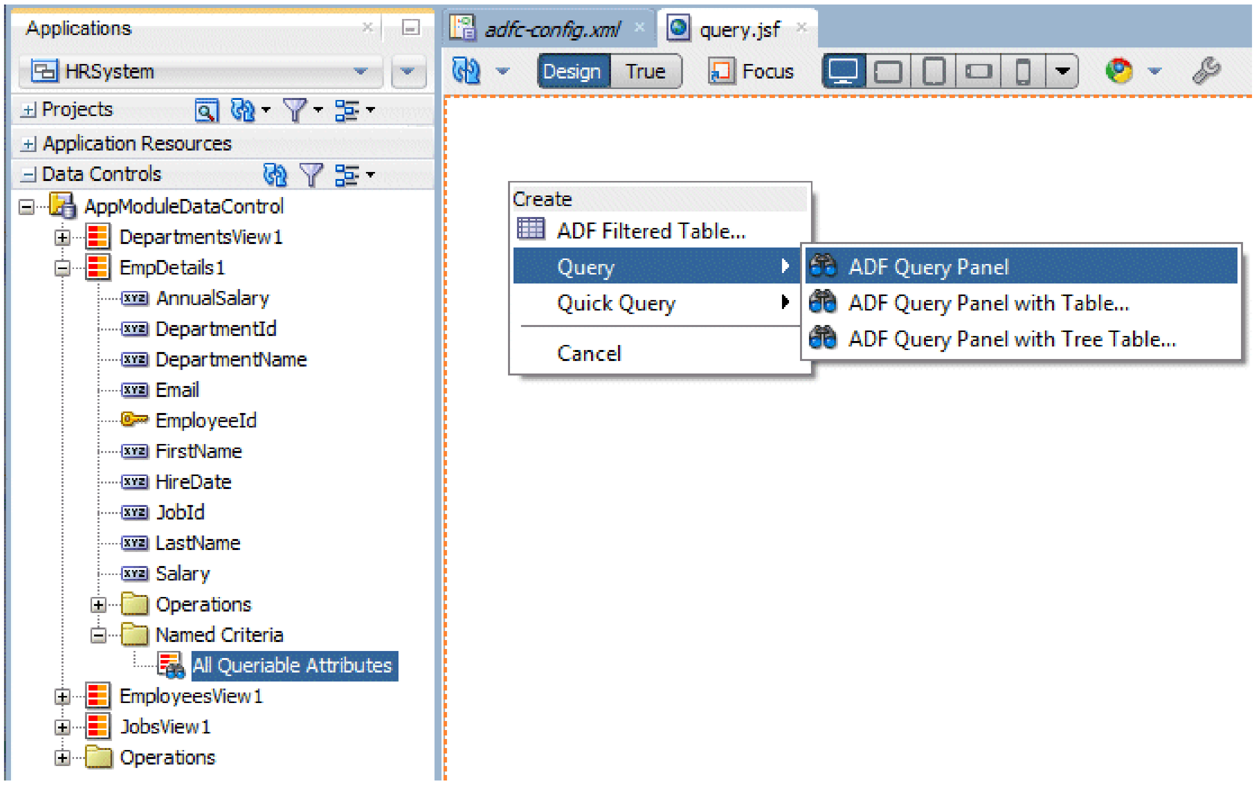
When you copy a Quick Start Layout into your page, you can delete or modify the generated components as needed. Unlike a template, there is no permanent link between your page and the layout you use to get started.

Apply color theme to layout components

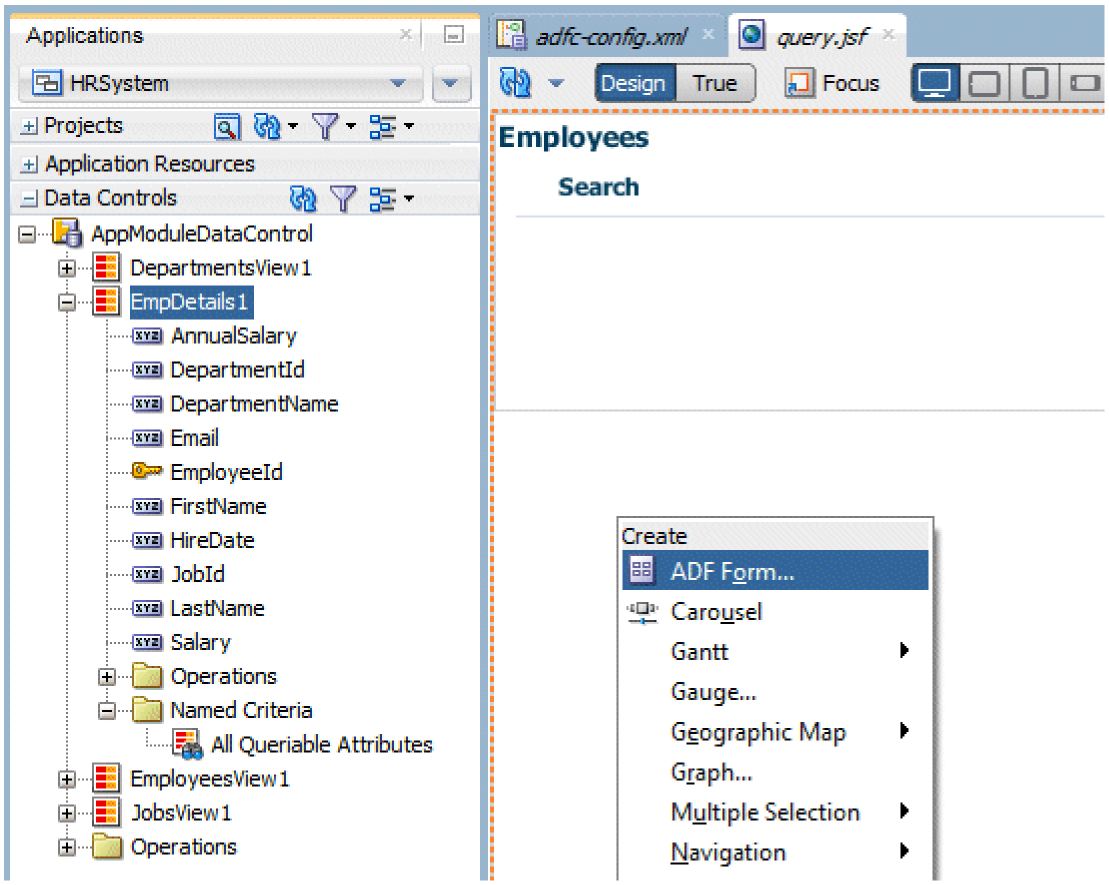
7. To add the employees search functionality to the page, open the Data Controls accordion, and locate **EmpDetails1**. (If you do not see it click the **Refresh** button).



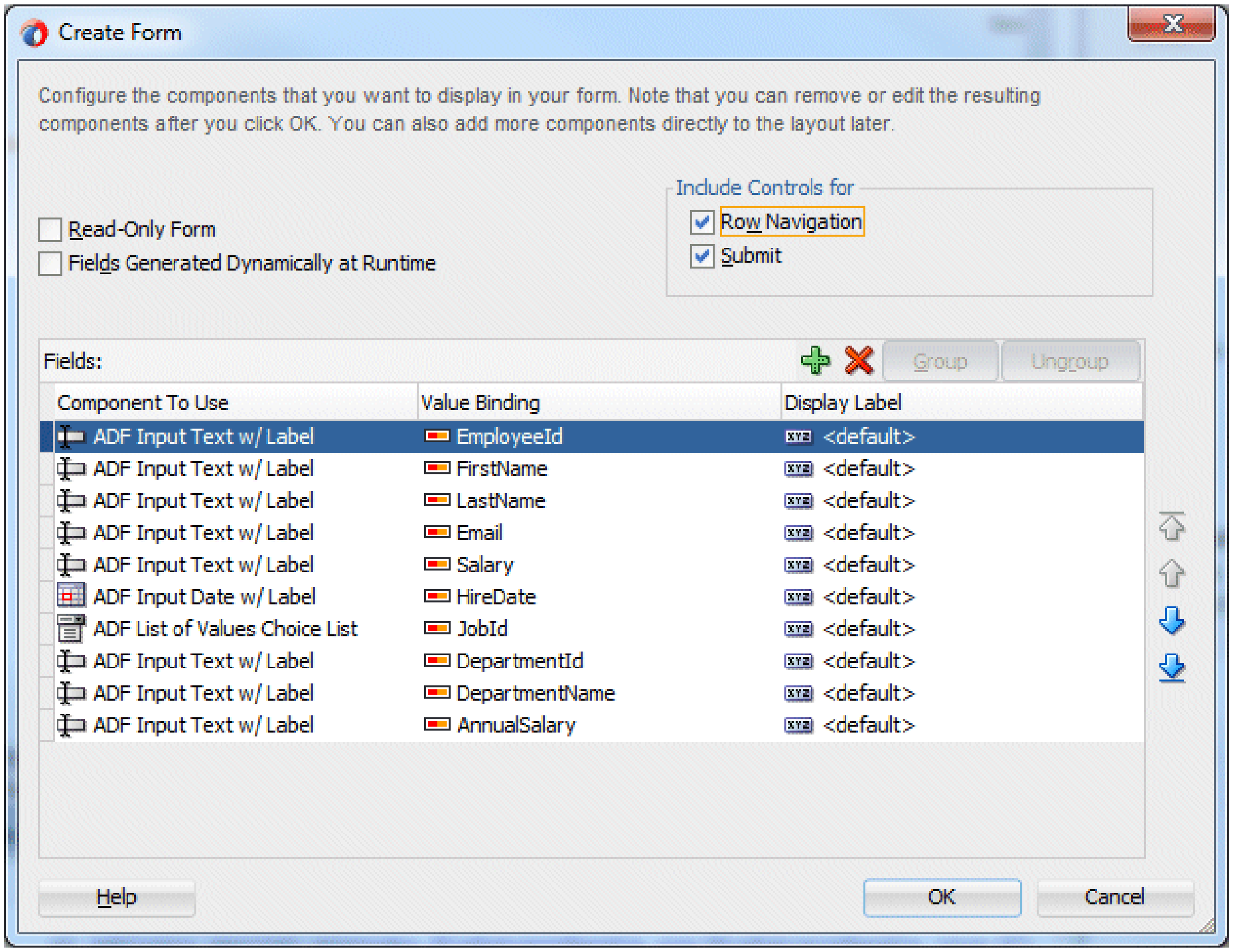
8. Expand the **EmpDetails1** data control and expand the **Named Criteria** node below it. Select **All Queriable Attributes** and drag it into the new **query.jsf** page. Create it as a **Query > ADF Query Panel**.



9. In the Data Controls accordion select the **EmpDetails1** data control and drag it into the center area of the page below the query component. Create it as an **ADF Form...**



In the Edit Forms Details, check both the **Row Navigation** and the **Submit Button** checkboxes. Click **OK**.



Create Form

Configure the components that you want to display in your form. Note that you can remove or edit the resulting components after you click OK. You can also add more components directly to the layout later.

☐ Read-Only Form
☐ Fields Generated Dynamically at Runtime

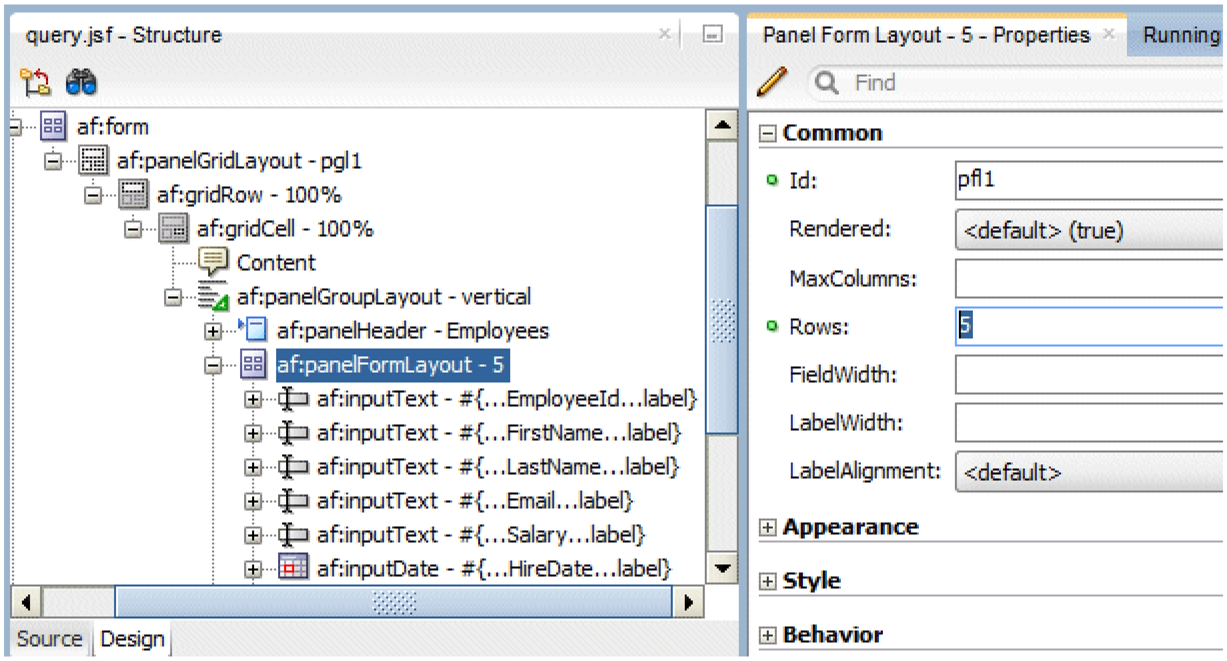
Include Controls for
☑ Row Navigation
☑ Submit

Fields:

Component To Use	Value Binding	Display Label
ADF Input Text w/ Label	EmployeeId	<default>
ADF Input Text w/ Label	FirstName	<default>
ADF Input Text w/ Label	LastName	<default>
ADF Input Text w/ Label	Email	<default>
ADF Input Text w/ Label	Salary	<default>
ADF Input Date w/ Label	HireDate	<default>
ADF List of Values Choice List	JobId	<default>
ADF Input Text w/ Label	DepartmentId	<default>
ADF Input Text w/ Label	DepartmentName	<default>
ADF Input Text w/ Label	AnnualSalary	<default>

10. With the **panelFormLayout** still selected, use the Properties window to set the **Rows** property to **5**.

    Also, set the **Short Desc** property to **salary graph**

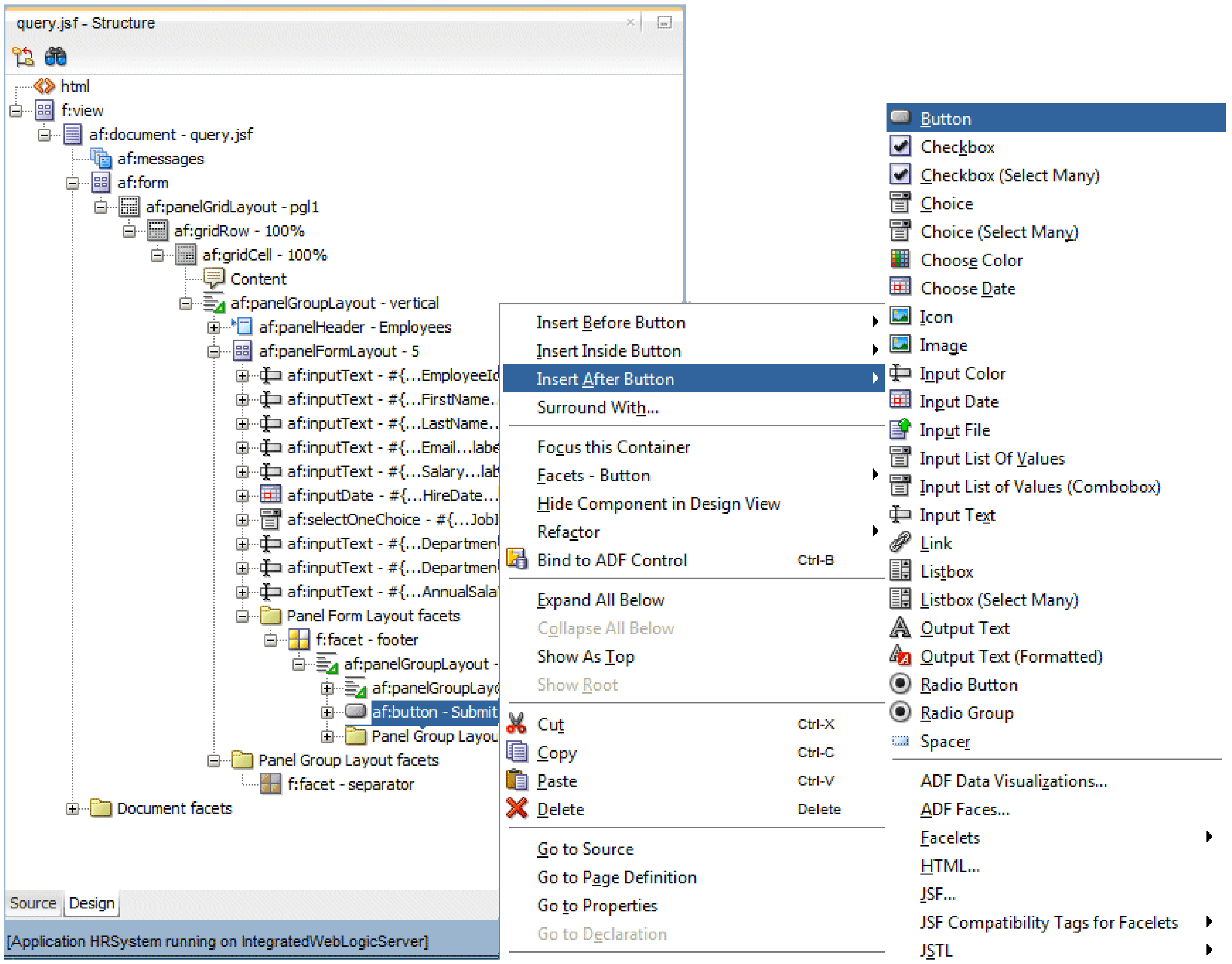

query.jsf - Structure

af:form
af:panelGridLayout - pgl1
af:gridRow - 100%
af:gridCell - 100%
Content
af:panelGroupLayout - vertical
af:panelHeader - Employees
af:panelFormLayout - 5
af:inputText - #{...EmployeeId...label}
af:inputText - #{...FirstName...label}
af:inputText - #{...LastName...label}
af:inputText - #{...Email...label}
af:inputText - #{...Salary...label}
af:inputDate - #{...HireDate...label}

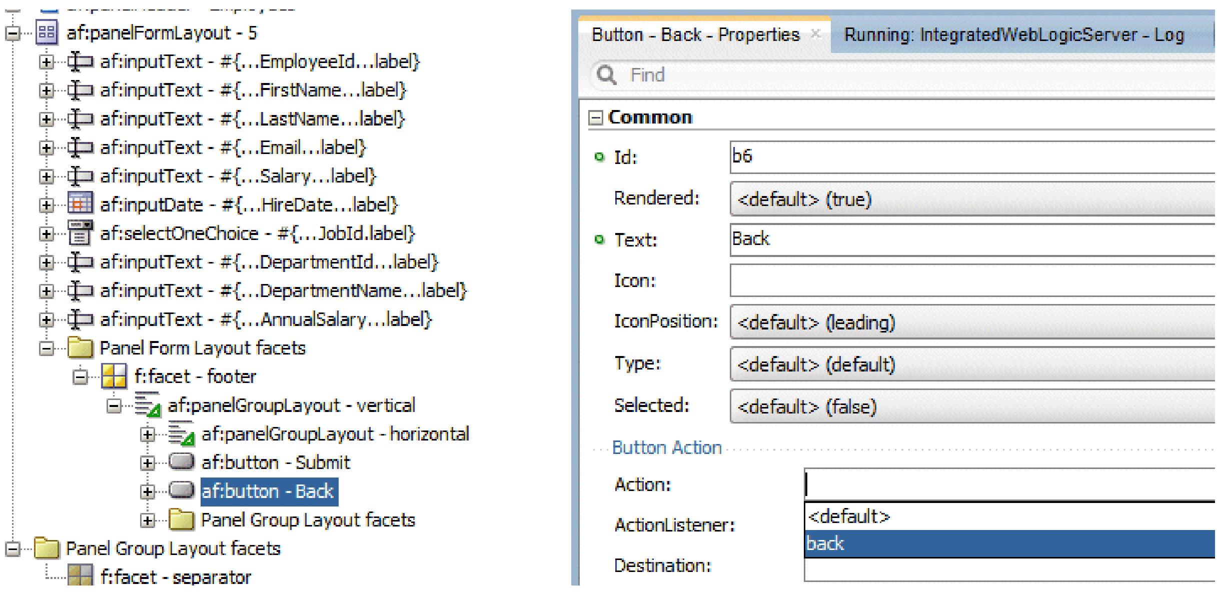
Panel Form Layout - 5 - Properties

Common
Id: pfl1
Rendered: <default> (true)
MaxColumns:
Rows: 5
FieldWidth:
LabelWidth:
LabelAlignment: <default>

11. In the Structure window locate the **Submit** button, right-click it and choose **Insert After Button > Button.**

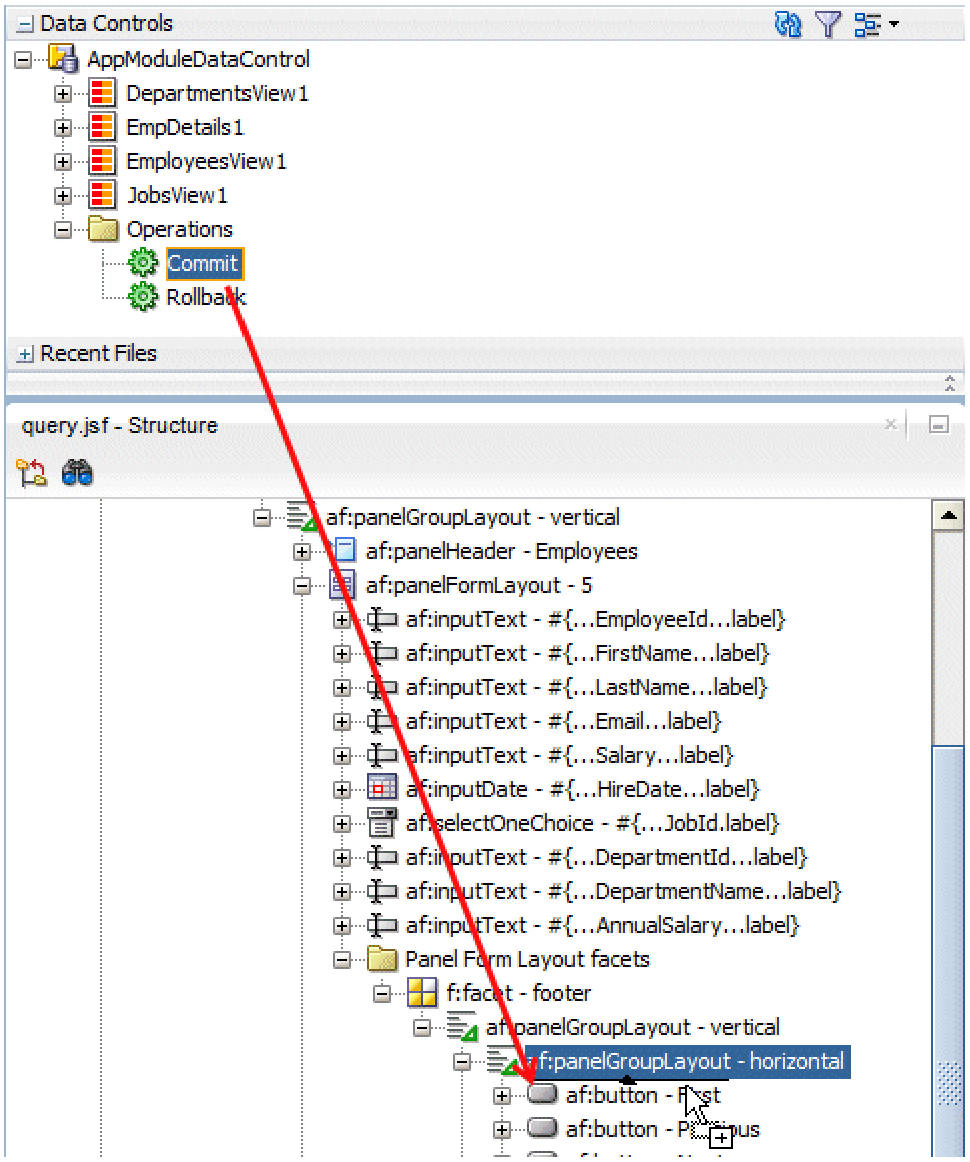


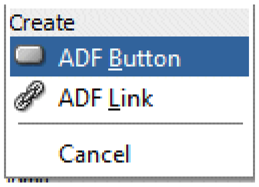
12. In the Properties window change the **Text** of the new button to **Back** and for the **Action** property select **back** from the drop down list. This causes the button to perform the navigation you defined in the page flow diagram.


af:panelFormLayout - 5
af:inputText - #{...EmployeeId...label}
af:inputText - #{...FirstName...label}
af:inputText - #{...LastName...label}
af:inputText - #{...Email...label}
af:inputText - #{...Salary...label}
af:inputDate - #{...HireDate...label}
af:selectOneChoice - #{...JobId.label}
af:inputText - #{...DepartmentId...label}
af:inputText - #{...DepartmentName...label}
af:inputText - #{...AnnualSalary...label}
Panel Form Layout facets
f:facet - footer
af:panelGroupLayout - vertical
af:panelGroupLayout - horizontal
af:button - Submit
af:button - Back
Panel Group Layout facets
Panel Group Layout facets
f:facet - separator
Button - Back - Properties
Running: IntegratedWebLogicServer - Log
Find
Common
Button Action
<default>
back

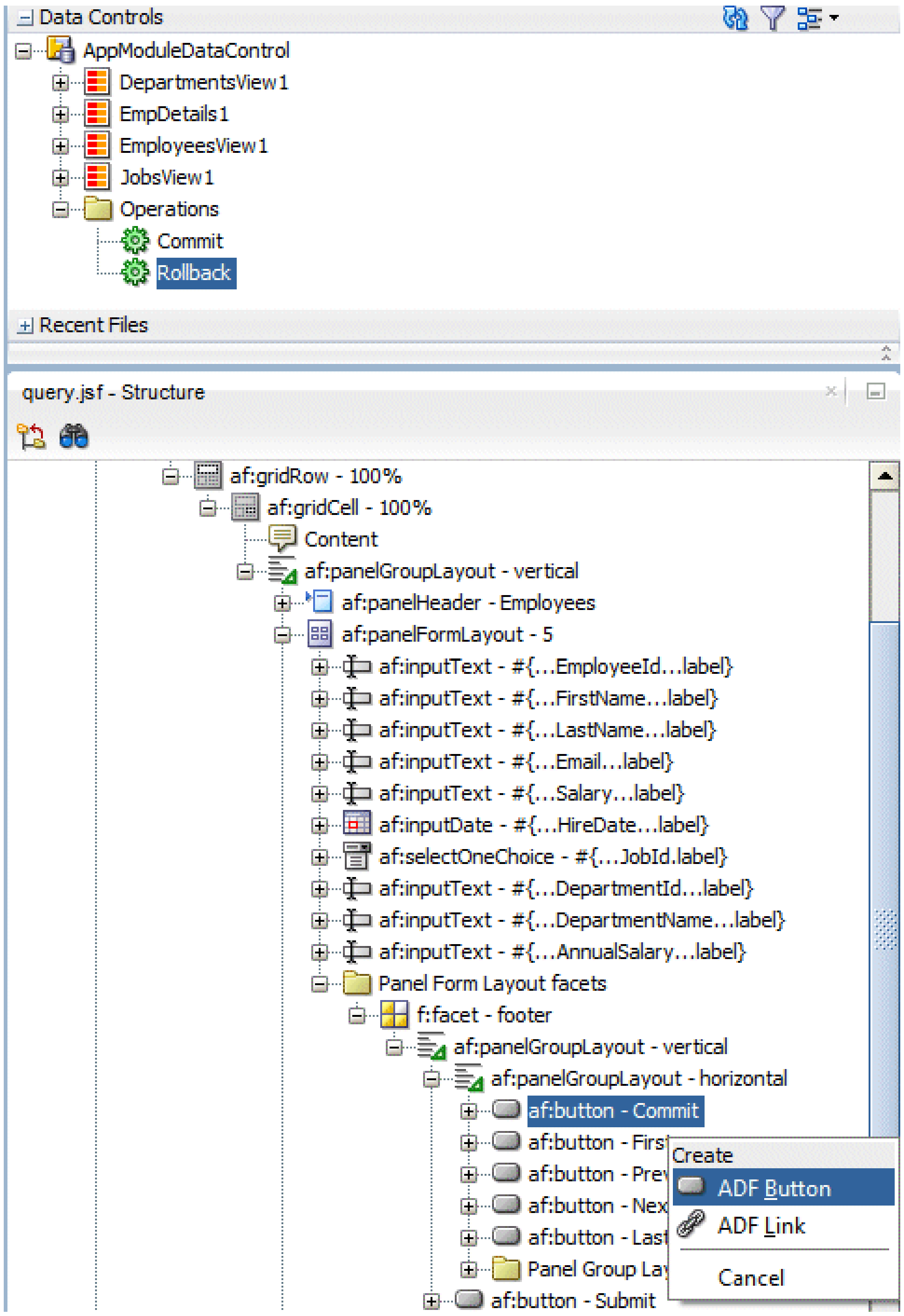
Id: b6
Rendered: <default> (true)
Text: Back
Icon:
IconPosition: <default> (leading)
Type: <default> (default)
Selected: <default> (false)
Action:
ActionListener:
Destination:

13. Next you add transaction operations to the page to allow you to commit and rollback changes. In the Data Controls Palette expand the application module level **Operations** node to locate the commit and rollback operations. Drag the **Commit** operation into the Structure window before the **First Button**. When prompted for a drop target choose **ADF Button.**


Data Controls
AppModuleDataControl
DepartmentsView1
EmpDetails1
EmployeesView1
JobsView1
Operations
Commit
Rollback
Recent Files
query.jsf - Structure
af:panelGroupLayout - vertical
af:panelHeader - Employees
af:panelFormLayout - 5
af:inputText - #{...EmployeeId...label}
af:inputText - #{...FirstName...label}
af:inputText - #{...LastName...label}
af:inputText - #{...Email...label}
af:inputText - #{...Salary...label}
af:inputDate - #{...HireDate...label}
af:selectOneChoice - #{...JobId.label}
af:inputText - #{...DepartmentId...label}
af:inputText - #{...DepartmentName...label}
af:inputText - #{...AnnualSalary...label}
Panel Form Layout facets
f:facet - footer
af:panelGroupLayout - vertical
af:panelGroupLayout - horizontal
af:button - First
af:button - Previous

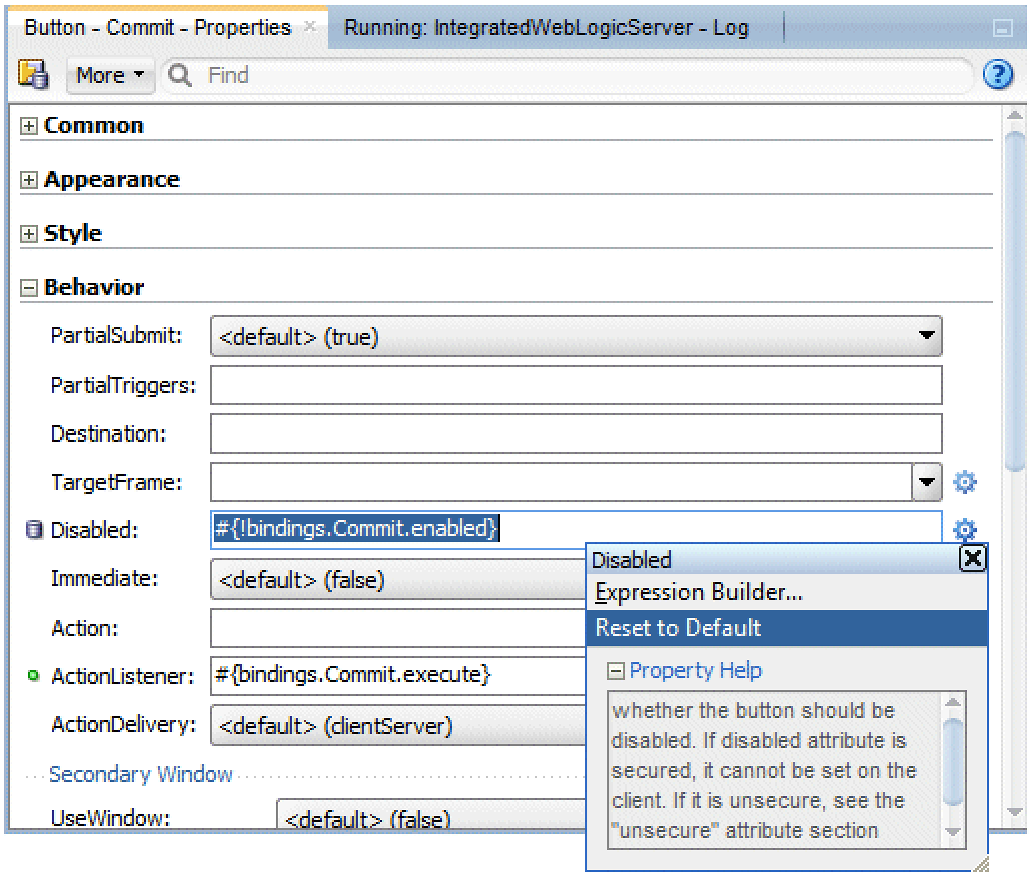


14. Repeat the same steps for the **Rollback** operation.


Data Controls
AppModuleDataControl
DepartmentsView1
EmpDetails1
EmployeesView1
JobsView1
Operations
Commit
Rollback
Recent Files
query.jsf - Structure
af:gridRow - 100%
af:gridCell - 100%
Content
af:panelGroupLayout - vertical
af:panelHeader - Employees
af:panelFormLayout - 5
af:inputText - #{...EmployeeId...label}
af:inputText - #{...FirstName...label}
af:inputText - #{...LastName...label}
af:inputText - #{...Email...label}
af:inputText - #{...Salary...label}
af:inputDate - #{...HireDate...label}
af:selectOneChoice - #{...JobId.label}
af:inputText - #{...DepartmentId...label}
af:inputText - #{...DepartmentName...label}
af:inputText - #{...AnnualSalary...label}
Panel Form Layout facets
f:facet - footer
af:panelGroupLayout - vertical
af:panelGroupLayout - horizontal
af:button - Commit
af:button - Firs
af:button - Prev
af:button - Nex
af:button - Last
Panel Group Lay
af:button - Submit
Create
ADF Button
ADF Link
Cancel

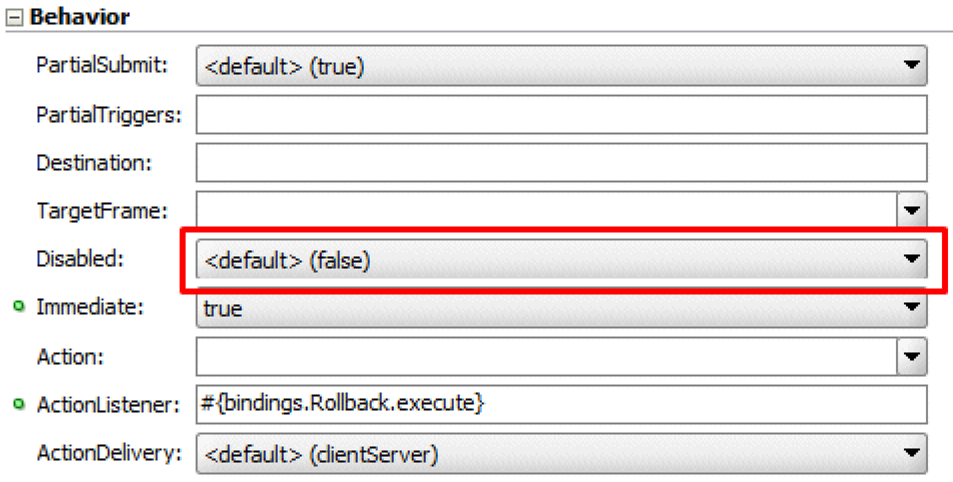
15. In the Properties window, for the **Rollback** and for the **Commit** buttons, reset the Disabled property to **default** to make both buttons always selectable.

Move the mouse over then end of the property field and you'll see a light blue gear appear. Click it to perform this operation.



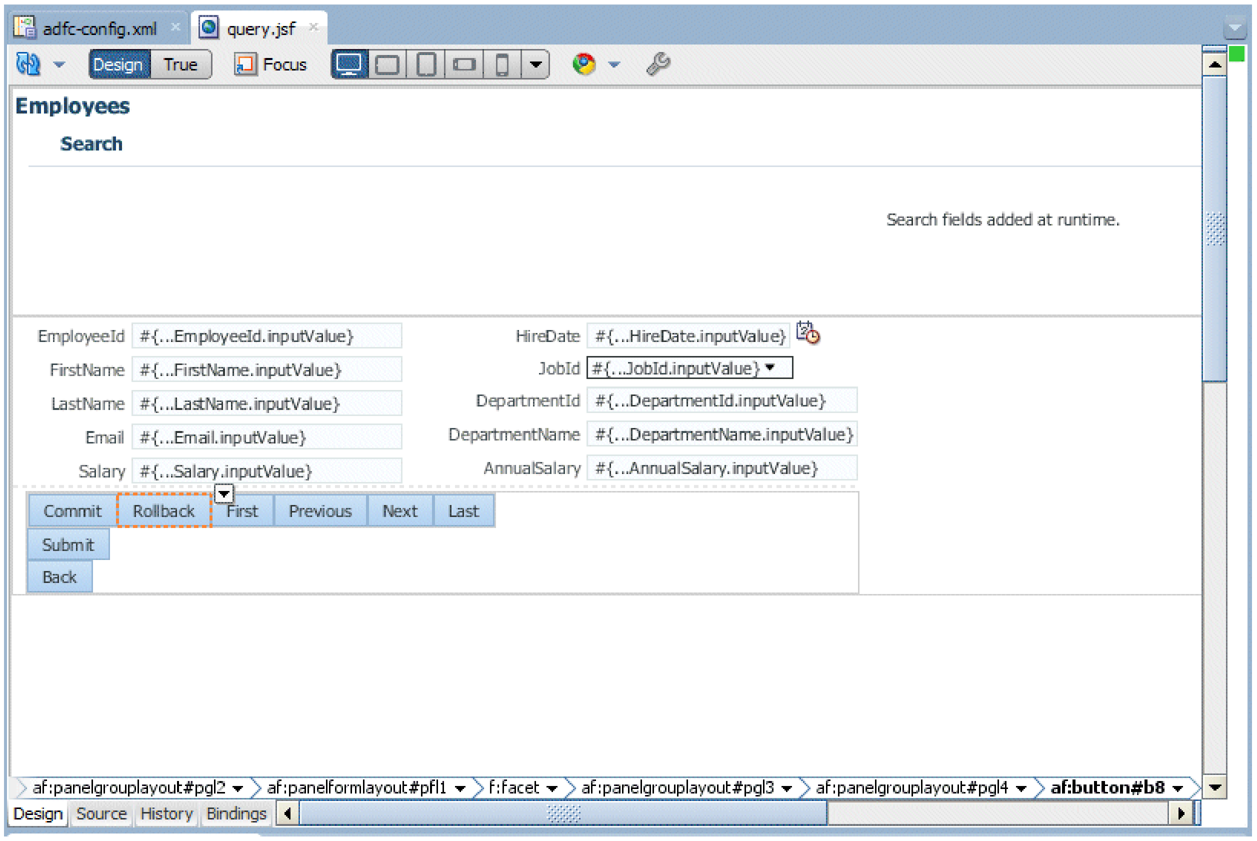
PartialSubmit:	<default> (true)
PartialTriggers:	
Destination:	
TargetFrame:	
Disabled:	#{!bindings.Commit.enabled}
Immediate:	<default> (false)
Action:	
ActionListener:	#{bindings.Commit.execute}
ActionDelivery:	<default> (clientServer)
UseWindow:	<default> (false)

whether the button should be disabled. If disabled attribute is secured, it cannot be set on the client. If it is unsecure, see the "unsecure" attribute section

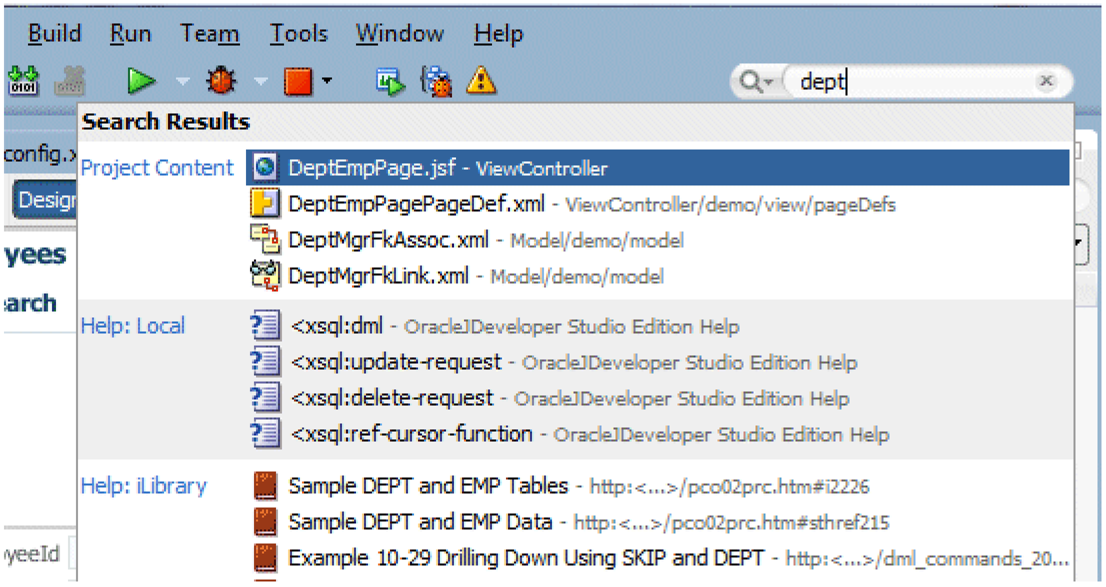


PartialSubmit:	<default> (true)
PartialTriggers:	
Destination:	
TargetFrame:	
Disabled:	<default> (false)
Immediate:	true
Action:	
ActionListener:	#{bindings.Rollback.execute}
ActionDelivery:	<default> (clientServer)

16. Double-click the **Query.jsf** tab to maximize the page in the Design Editor. Your page should look as follows:

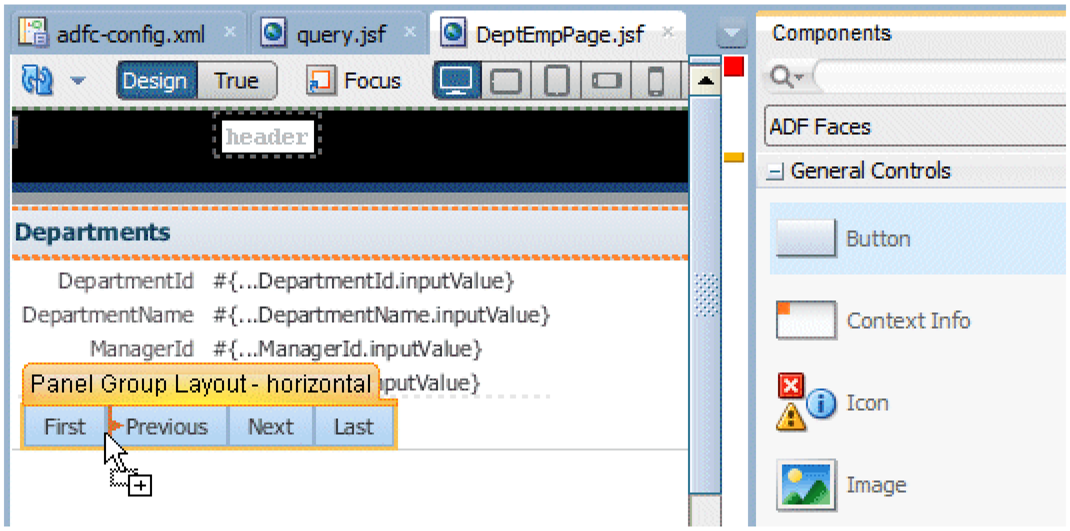

adfc-config.xml
query.jsf
Design
True
Focus
Employees
Search
Search fields added at runtime.
EmployeeId #{...EmployeeId.inputValue}
FirstName #{...FirstName.inputValue}
LastName #{...LastName.inputValue}
Email #{...Email.inputValue}
Salary #{...Salary.inputValue}
HireDate #{...HireDate.inputValue}
JobId #{...JobId.inputValue}
DepartmentId #{...DepartmentId.inputValue}
DepartmentName #{...DepartmentName.inputValue}
AnnualSalary #{...AnnualSalary.inputValue}
Commit
Rollback
First
Previous
Next
Last
Submit
Back
af:panelgrouplayout#pgl2
af:panelformlayout#pfl1
f:facet
af:panelgrouplayout#pgl3
af:panelgrouplayout#pgl4
af:button#b8
Design
Source
History
Bindings

17. Double-click the query.jsf tab again to return it to normal size.
18. Click the **DeptEmpPage.jsf** tab to switch back to the page. A quick way to navigate to this or any other file is by using the global find box at the top right of JDeveloper and typing the file name there.

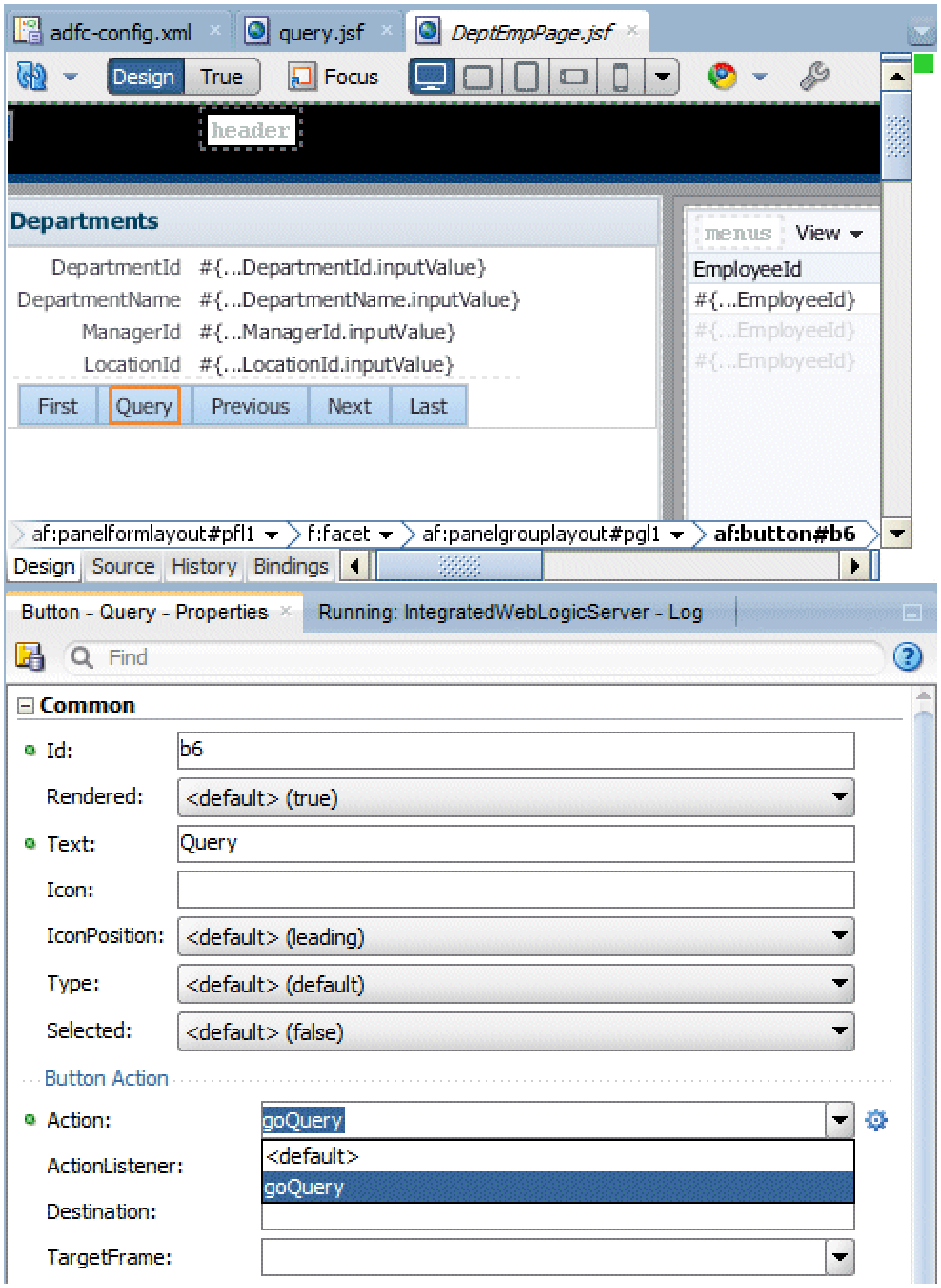

Build
Run
Team
Tools
Window
Help
dept
Search Results
Project Content
DeptEmpPage.jsf - ViewController
DeptEmpPagePageDef.xml - ViewController/demo/view/pageDefs
DeptMgrFkAssoc.xml - Model/demo/model
DeptMgrFkLink.xml - Model/demo/model
Help: Local
<xsql:dml - OracleJDeveloper Studio Edition Help
<xsql:update-request - OracleJDeveloper Studio Edition Help
<xsql:delete-request - OracleJDeveloper Studio Edition Help
<xsql:ref-cursor-function - OracleJDeveloper Studio Edition Help
Help: iLibrary
Sample DEPT and EMP Tables - http:<...>/pco02prc.htm#i2226
Sample DEPT and EMP Data - http:<...>/pco02prc.htm#sthref215
Example 10-29 Drilling Down Using SKIP and DEPT - http:<...>/dml_commands_20...

Then just click the file name to open it in the editor.

19. In the page design, expand the **Departments** accordion. From the Components window, choose a **Button** component and drag it into the **Departments** accordion between the **First** and **Previous** buttons. Alternatively you can right-click the First button and choose insert after > button to add the new button.


adfc-config.xml
query.jsf
DeptEmpPage.jsf
Design
True
Focus
header
Departments
DepartmentId #{...DepartmentId.inputValue}
DepartmentName #{...DepartmentName.inputValue}
ManagerId #{...ManagerId.inputValue}
Panel Group Layout - horizontal
First
Previous
Next
Last
Components
ADF Faces
General Controls
Button
Context Info
Icon
Image

20. Using the Properties window change the **Text** of the button to **Query** and for the **Action** property type **goQuery** or select it from the drop down list if available. This causes the button to perform the navigation you defined in the page flow diagram.



Departments

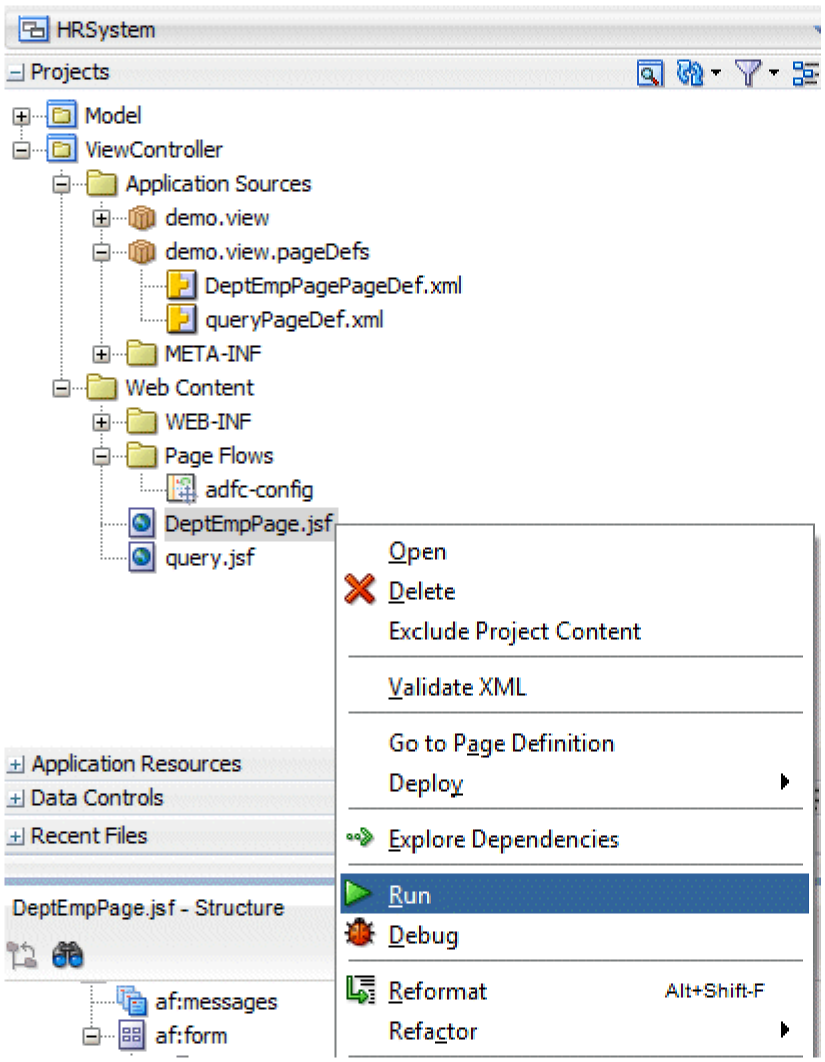
DepartmentId	#{...DepartmentId.inputValue}
DepartmentName	#{...DepartmentName.inputValue}
ManagerId	#{...ManagerId.inputValue}
LocationId	#{...LocationId.inputValue}

First | Query | Previous | Next | Last

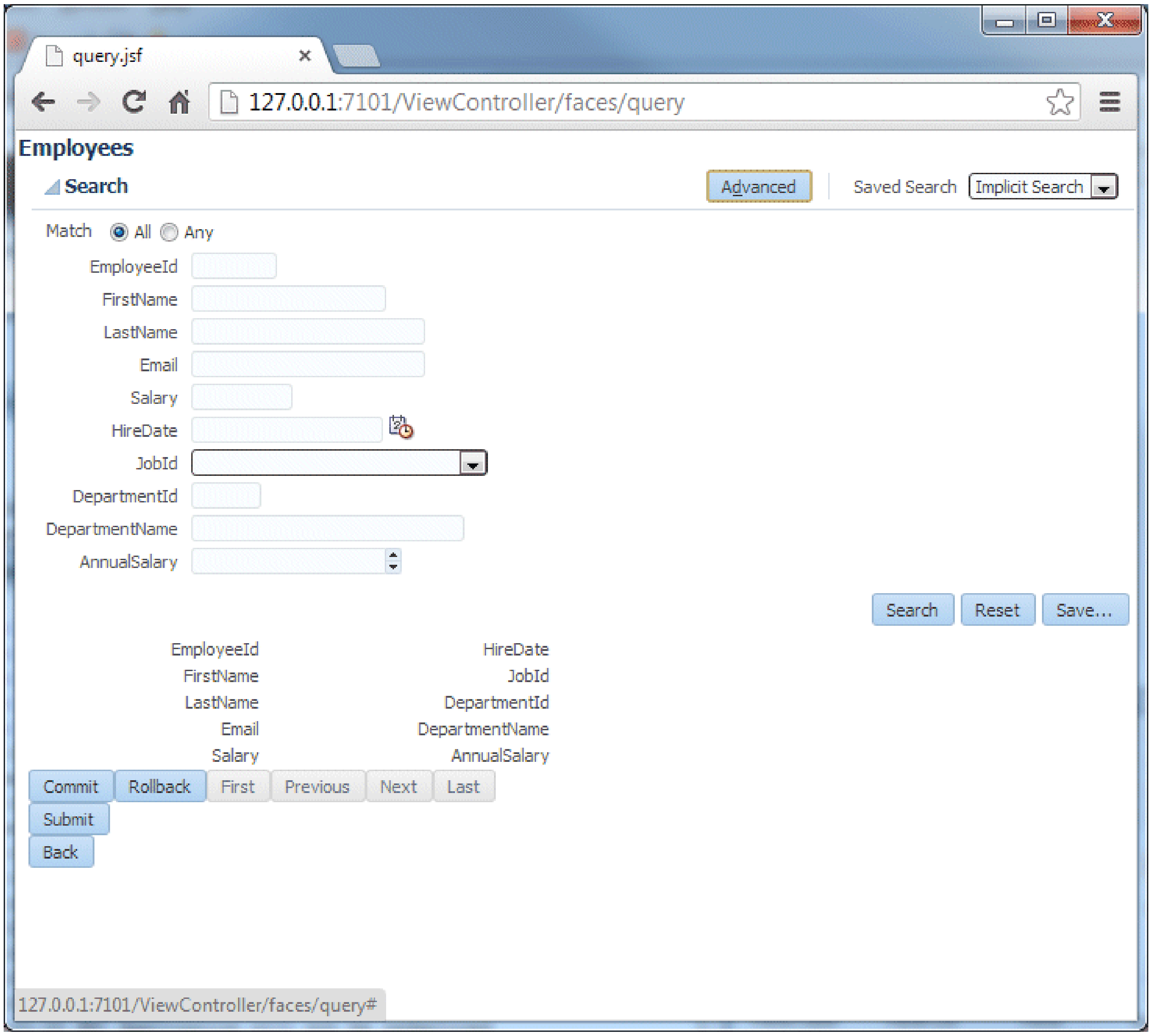
EmployeeId
#{...EmployeeId}
#{...EmployeeId}
#{...EmployeeId}

Id:	b6
Rendered:	<default> (true)
Text:	Query
Icon:	
IconPosition:	<default> (leading)
Type:	<default> (default)
Selected:	<default> (false)
Action:	goQuery <default> goQuery
ActionListener:	
Destination:	
TargetFrame:	

21. Click the **Save All** icon on the JDeveloper menu bar to save your work, and then right-click the **DeptEmpPage.jsf** page and choose **Run**.



22. When the page displays in your browser click the **Query** button to navigate to the new page. In the Search page click **Advanced** to display detailed search criteria.



query.jsf

127.0.0.1:7101/ViewController/faces/query

Employees

Search

Advanced | Saved Search: Implicit Search

Match [✓ All] Any

EmployeeId	
FirstName	
LastName	
Email	
Salary	
HireDate	
JobId	
DepartmentId	
DepartmentName	
AnnualSalary	

Search | Reset | Save...

EmployeeId		HireDate	
FirstName		JobId	
LastName		DepartmentId	
Email		DepartmentName	
Salary		AnnualSalary	

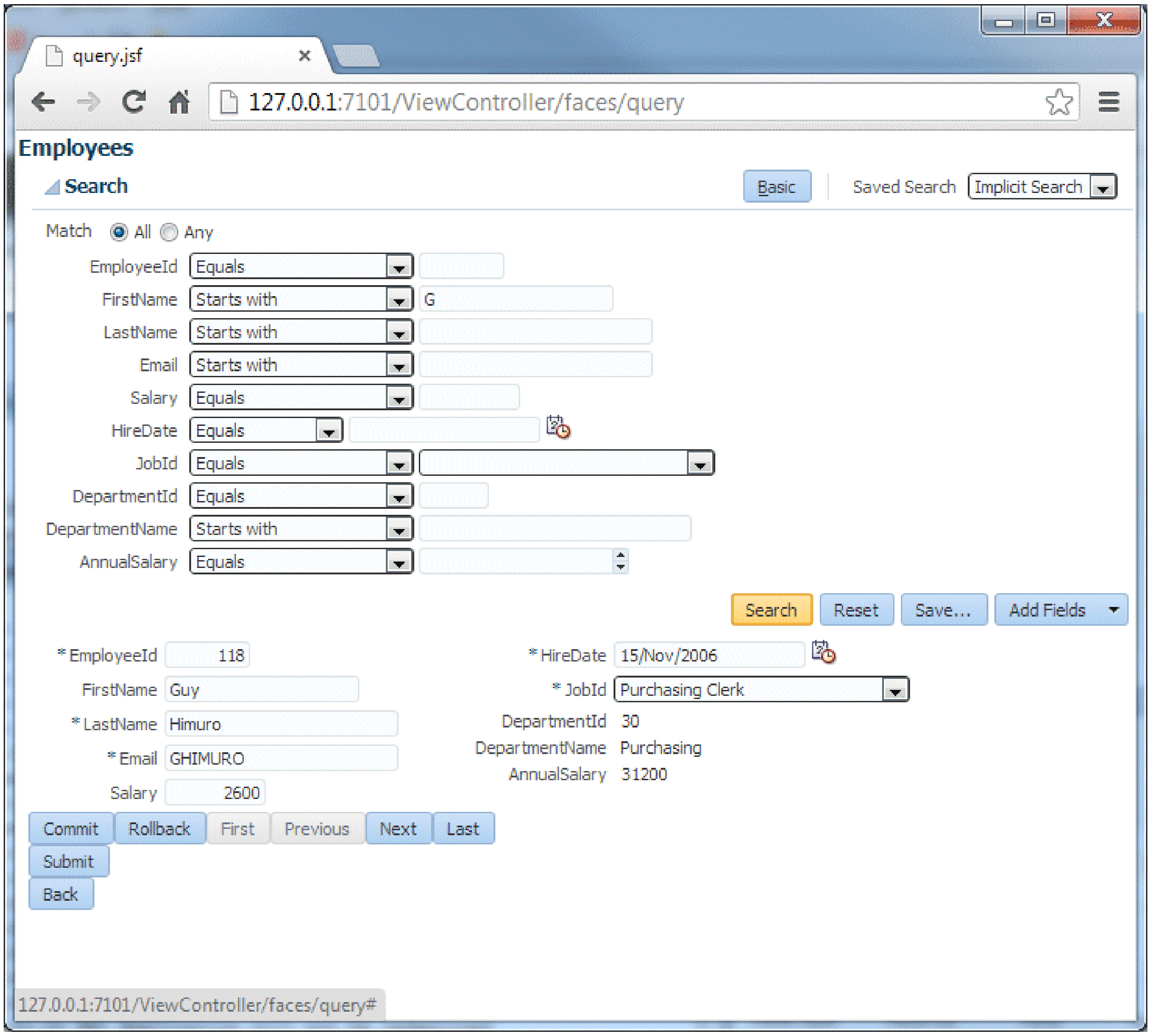
Commit | Rollback | First | Previous | Next | Last

Submit

Back

127.0.0.1:7101/ViewController/faces/query#

23. In the **FirstName** field accept the default **Starts with**, and type the letter **G**. Press **Search**. The form below displays the record for Guy Himuro.



Employees

Search

Saved Search: Implicit Search

Match (● All) Any

EmployeeId	Equals	
FirstName	Starts with	G
LastName	Starts with	
Email	Starts with	
Salary	Equals	
HireDate	Equals	
JobId	Equals	
DepartmentId	Equals	
DepartmentName	Starts with	
AnnualSalary	Equals	

* EmployeeId	118	* HireDate	15/Nov/2006
FirstName	Guy	* JobId	Purchasing Clerk
* LastName	Himuro	DepartmentId	30
* Email	GHIMURO	DepartmentName	Purchasing
Salary	2600	AnnualSalary	31200

24. Experiment with the form, saving your search criteria, creating more complex queries and updating data for employees. Note how this form displays a view of the data that matches the definition in the view object you created - including information for Department name as well as a list of values for the Job id and the employee's annual salary.
You can also make changes to the data and **commit** and **rollback** your transactions as needed.

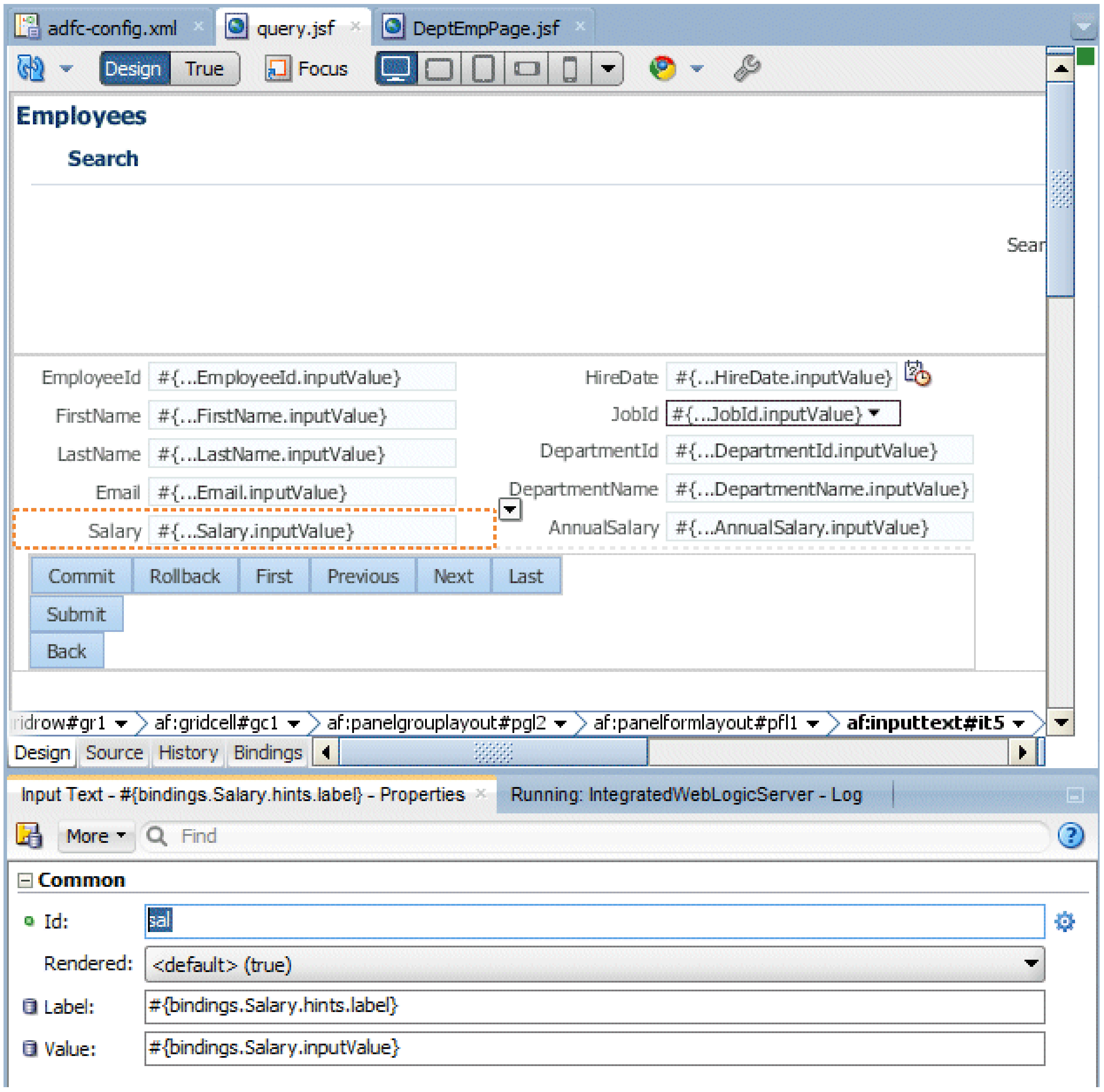
When you are finished **close** the browser window.

Step 2: Use Partial Page Refresh

In the next steps you become acquainted with some of the features of the ADF Faces Framework. You enhance your pages with additional Ajax functionality leveraging the declarative development features offered by ADF Faces components.

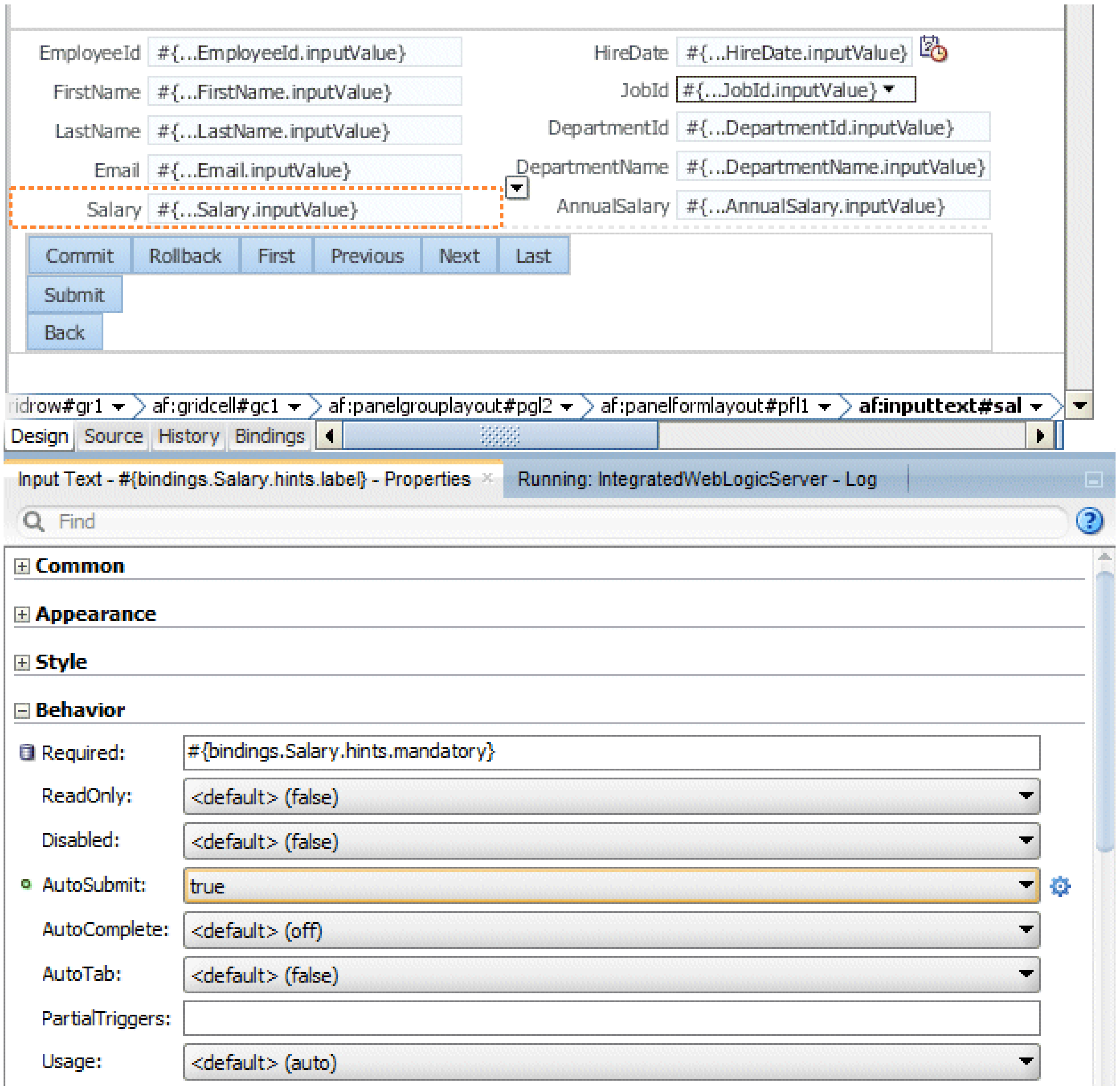
In the next few steps you use the partial page refresh capability offered by ADF Faces. You want to make sure that if the Salary field is updated, that the Annual Salary is recalculated. In addition, you don't want the entire page to refresh but rather just the affected fields. You will use an ADF Faces feature called Partial Page Rendering or PPR to accomplish that behavior.

1. Open the **query.jsf** file in the Design Editor, if it is not already open. Select the **Salary** field. In the Properties window set the value of the **Id** property to **sal**.


adfc-config.xml
query.jsf
DeptEmpPage.jsf
Design
True
Focus
Employees
Search
Sear
EmployeeId #{...EmployeeId.inputValue}
FirstName #{...FirstName.inputValue}
LastName #{...LastName.inputValue}
Email #{...Email.inputValue}
Salary #{...Salary.inputValue}
HireDate #{...HireDate.inputValue}
JobId #{...JobId.inputValue}
DepartmentId #{...DepartmentId.inputValue}
DepartmentName #{...DepartmentName.inputValue}
AnnualSalary #{...AnnualSalary.inputValue}
Commit
Rollback
First
Previous
Next
Last
Submit
Back
idrow#gr1 af:gridcell#gc1 af:panelgrouplayout#pgl2 af:panelformlayout#pfl1 af:inputtext#it5
Design
Source
History
Bindings
Input Text - #{bindings.Salary.hints.label} - Properties
Running: IntegratedWebLogicServer - Log
More
Find
Common

Id: sal
Rendered: <default> (true)
Label: #{bindings.Salary.hints.label}
Value: #{bindings.Salary.inputValue}

2. Still in the Properties window expand the **Behavior** node and set the **AutoSubmit** property to **True**.

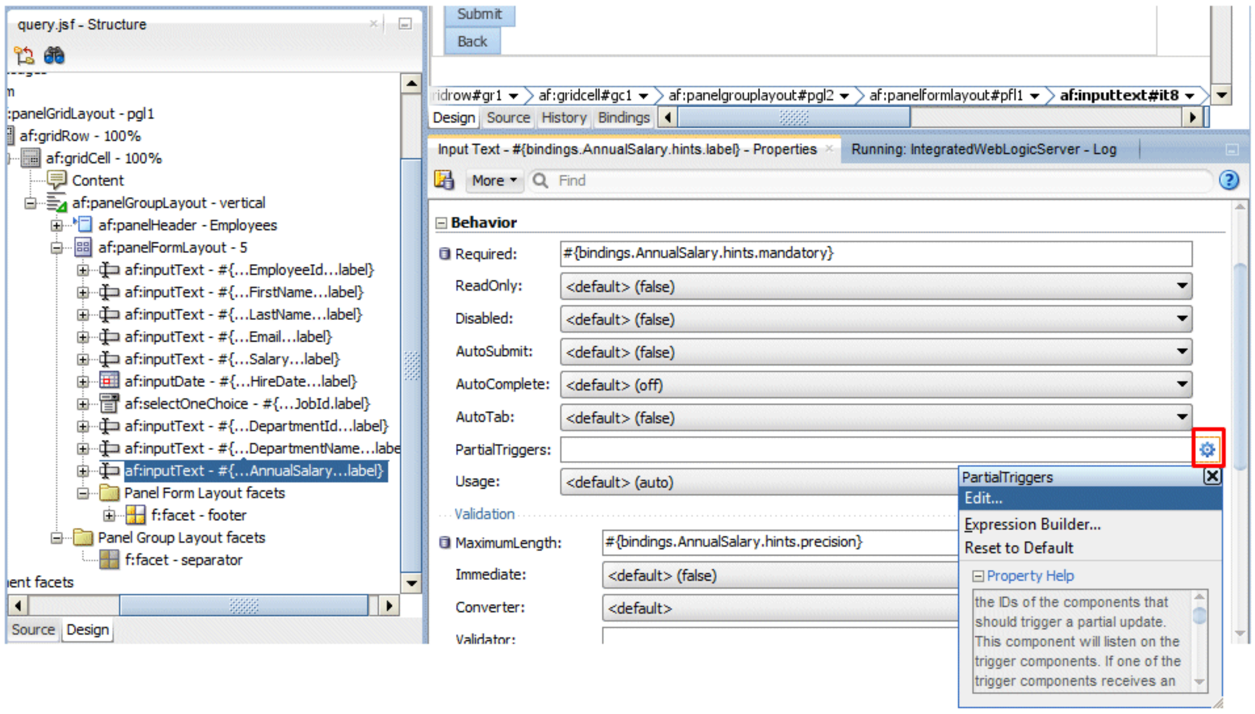


Design Source History Bindings
Input Text - #{bindings.Salary.hints.label} - Properties
Running: IntegratedWebLogicServer - Log
Find
Common
Appearance
Style
Behavior
Commit Rollback First Previous Next Last
Submit
Back
idrow#gr1 af:gridcell#gc1 af:panelgrouplayout#pgl2 af:panelformlayout#pfl1 af:inputtext#sal

EmployeeId	#{...EmployeeId.inputValue}
FirstName	#{...FirstName.inputValue}
LastName	#{...LastName.inputValue}
Email	#{...Email.inputValue}
Salary	#{...Salary.inputValue}
HireDate	#{...HireDate.inputValue}
JobId	#{...JobId.inputValue}
DepartmentId	#{...DepartmentId.inputValue}
DepartmentName	#{...DepartmentName.inputValue}
AnnualSalary	#{...AnnualSalary.inputValue}

Required:	#{bindings.Salary.hints.mandatory}
ReadOnly:	<default> (false)
Disabled:	<default> (false)
AutoSubmit:	true
AutoComplete:	<default> (off)
AutoTab:	<default> (false)
PartialTriggers:	
Usage:	<default> (auto)

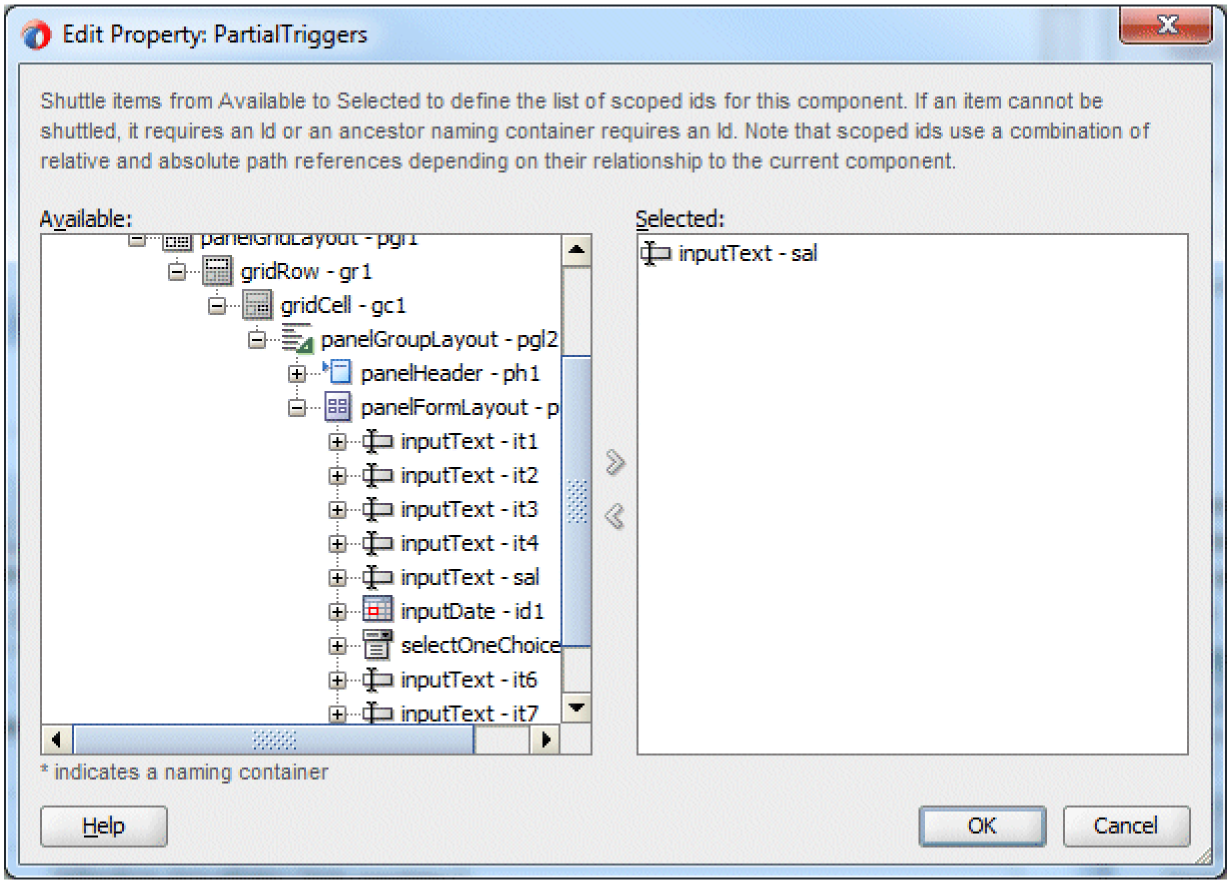
3. Either in the Design Editor or the Structure window locate the **AnnualSalary** field. Locate the **PartialTriggers** property under the **Behavior** section and click the gear icon to its right to choose **Edit**.


query.jsf - Structure
:panelGridLayout - pgl1
af:gridRow - 100%
af:gridCell - 100%
Content
af:panelGroupLayout - vertical
af:panelHeader - Employees
af:panelFormLayout - 5
af:inputText - #{...EmployeeId...label}
af:inputText - #{...FirstName...label}
af:inputText - #{...LastName...label}
af:inputText - #{...Email...label}
af:inputText - #{...Salary...label}
af:inputDate - #{...HireDate...label}
af:selectOneChoice - #{...JobId.label}
af:inputText - #{...DepartmentId...label}
af:inputText - #{...DepartmentName...labe
af:inputText - #{...AnnualSalary...label}
Panel Form Layout facets
f:facet - footer
Panel Group Layout facets
f:facet - separator
ent facets
Source
Design
Submit
Back
idrow#gr1
af:gridcell#gc1
af:panelgrouplayout#pgl2
af:panelformlayout#pfl1
af:inputtext#it8
Design
Source
History
Bindings
Input Text - #{bindings.AnnualSalary.hints.label} - Properties
Running: IntegratedWebLogicServer - Log
More
Find
Behavior
Validation
PartialTriggers
Edit...
Expression Builder...
Reset to Default
Property Help

Required: #{bindings.AnnualSalary.hints.mandatory}
ReadOnly: <default> (false)
Disabled: <default> (false)
AutoSubmit: <default> (false)
AutoComplete: <default> (off)
AutoTab: <default> (false)
PartialTriggers:
Usage: <default> (auto)
MaximumLength: #{bindings.AnnualSalary.hints.precision}
Immediate: <default> (false)
Converter: <default>
Validator:

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

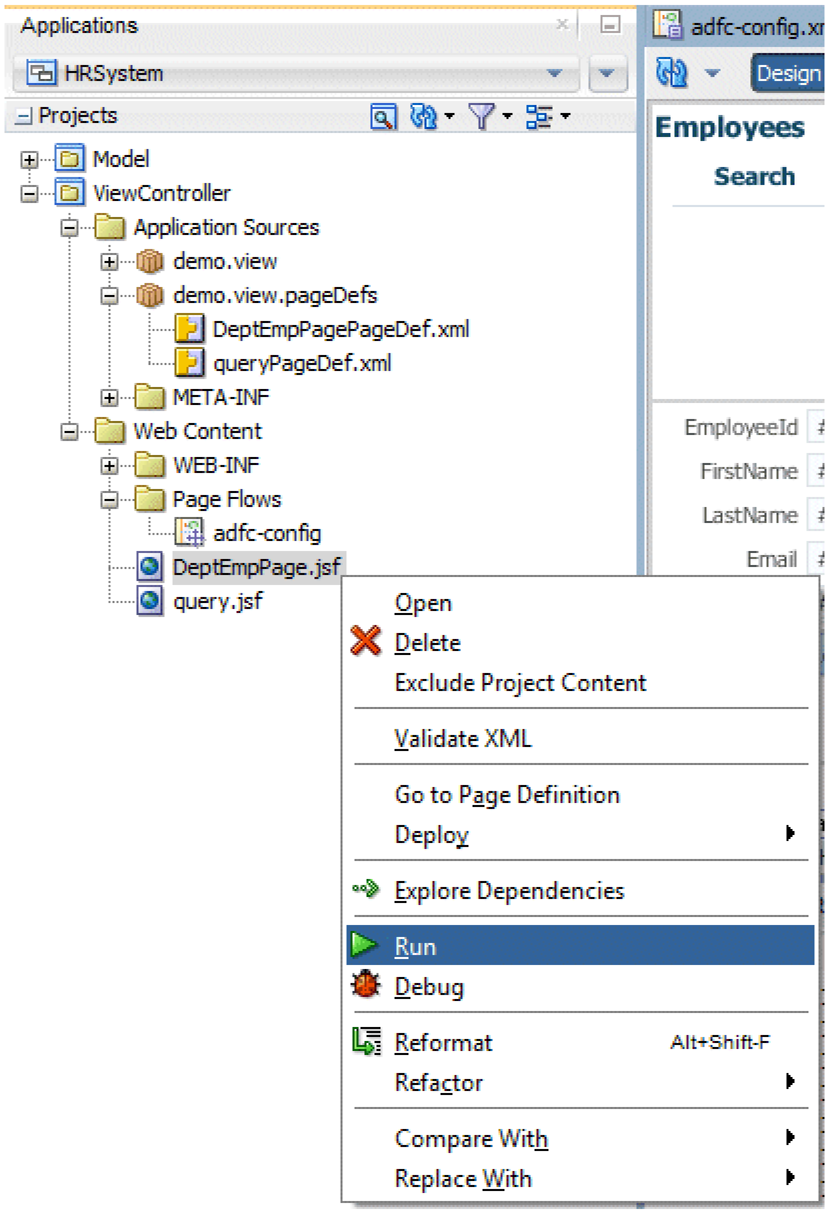
4. In the Edit Property dialog locate the **Salary** field and shuttle it to the right using the blue arrow. Click **OK**.



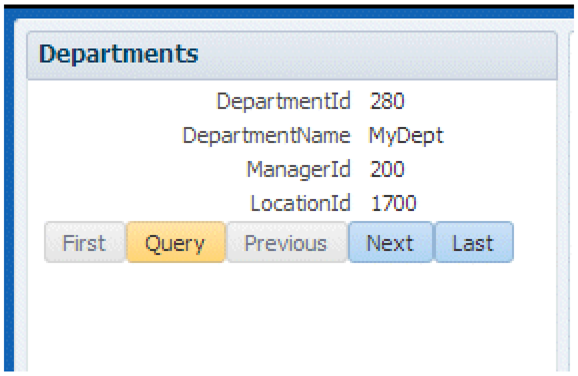
Shuttle items from Available to Selected to define the list of scoped ids for this component. If an item cannot be shuttled, it requires an Id or an ancestor naming container requires an Id. Note that scoped ids use a combination of relative and absolute path references depending on their relationship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

* indicates a naming container

5. Click the **Save All** icon on the JDeveloper menu bar to save your work, and **Run DeptEmpPage.jsf**.

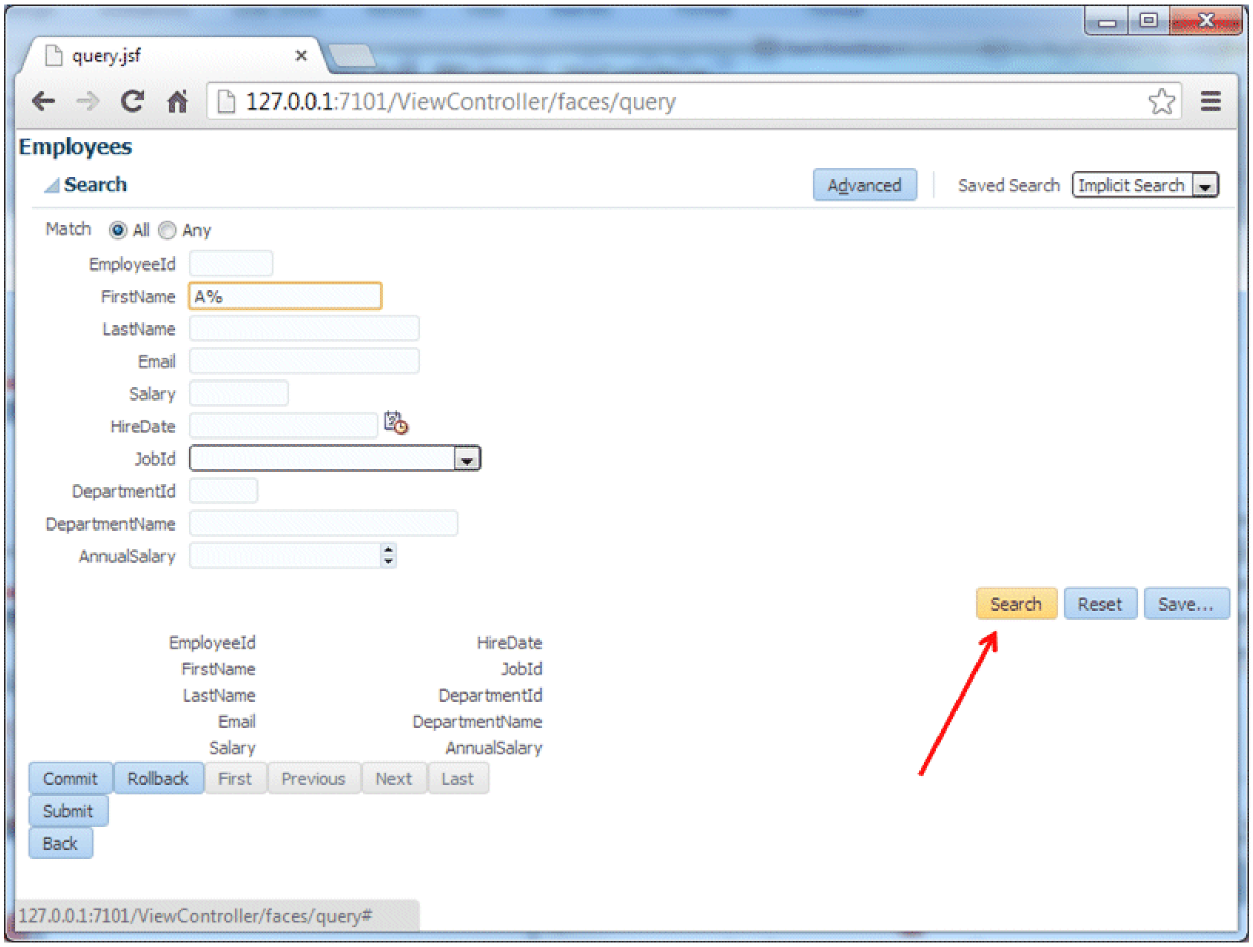


6. When the page displays, click the **Query** button in the Departments panel.



DepartmentId	280
DepartmentName	MyDept
ManagerId	200
LocationId	1700

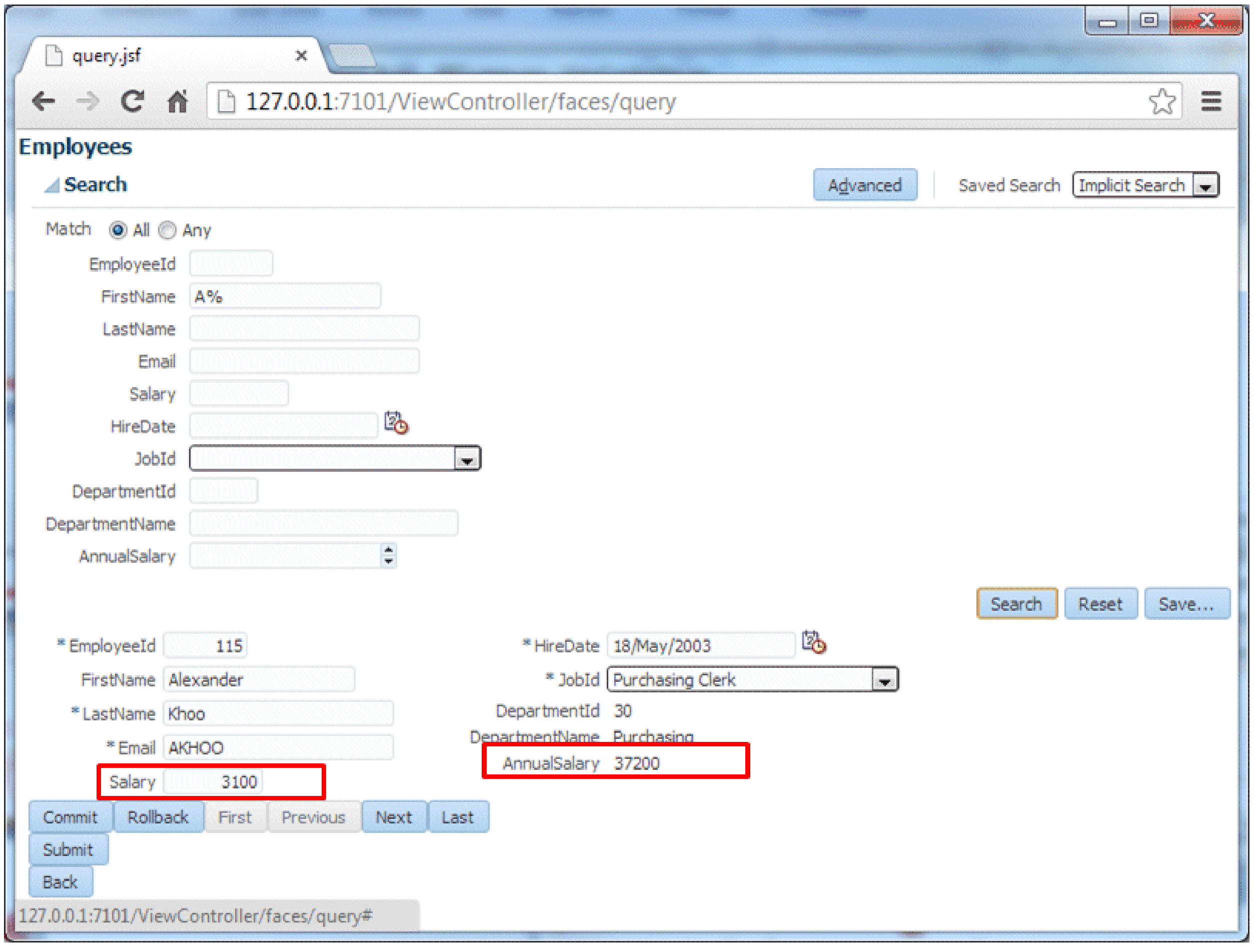
7. In the Search page search for employees whose first name begins with **A%**.



127.0.0.1:7101/ViewController/faces/query

Match	[✓ All] Any
EmployeeId	
FirstName	A%
LastName	
Email	
Salary	
HireDate	
JobId	
DepartmentId	
DepartmentName	
AnnualSalary	

Note the salary and annual salary values.



Employees

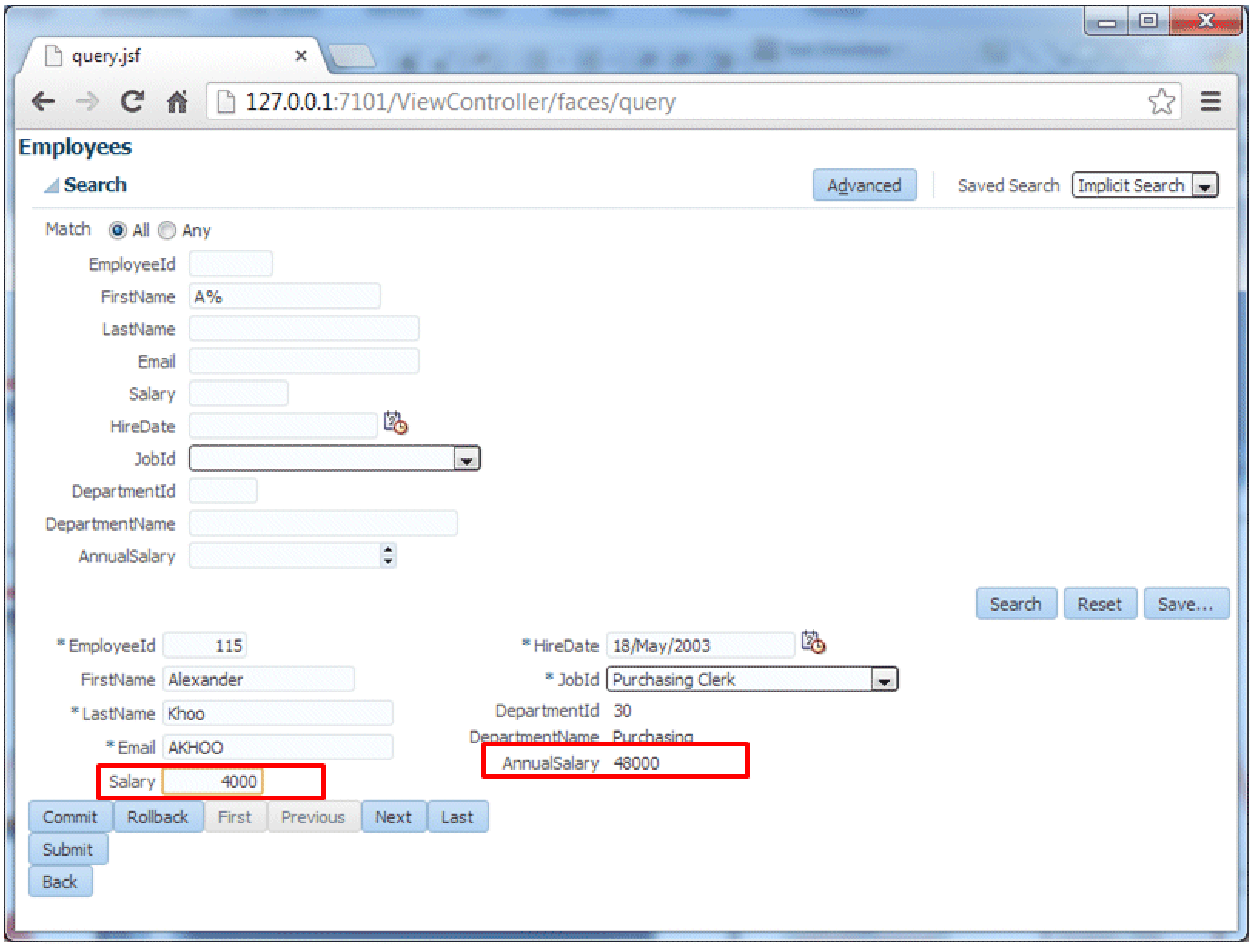
Search

Match [✓ All] Any

EmployeeId	
FirstName	A%
LastName	
Email	
Salary	
HireDate	
JobId	
DepartmentId	
DepartmentName	
AnnualSalary	

* EmployeeId	115	* HireDate	18/May/2003
FirstName	Alexander	* JobId	Purchasing Clerk
* LastName	Khoo	DepartmentId	30
* Email	AKHOO	DepartmentName	Purchasing
Salary	3100	AnnualSalary	37200

8. Update the **Salary** field and then tab out of it. Notice the immediate change to the **AnnualSalary** field once you leave the Salary field. However this is the only field that is refreshed (not the whole page).



Match [✓ All] Any

EmployeeId	
FirstName	A%
LastName	
Email	
Salary	
HireDate	
JobId	
DepartmentId	
DepartmentName	
AnnualSalary	

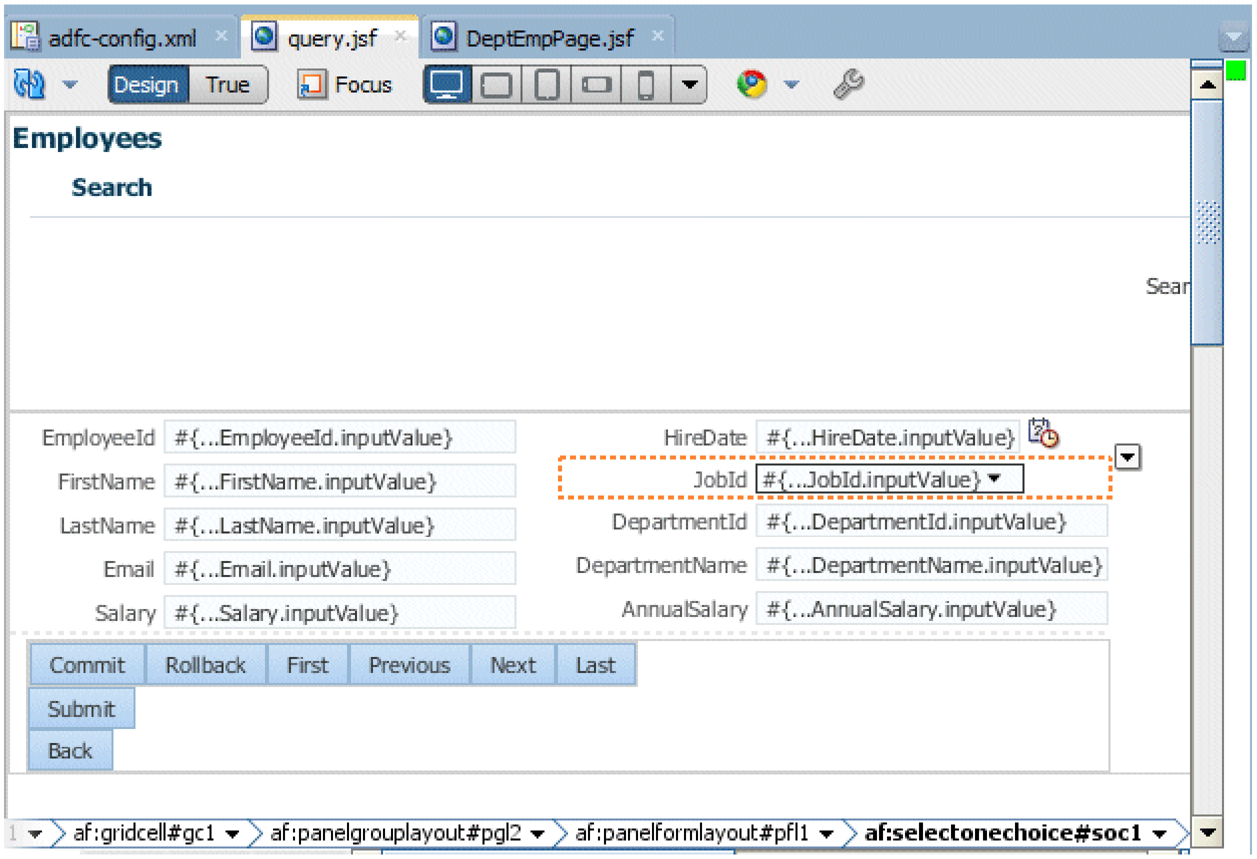
* EmployeeId	115	* HireDate	18/May/2003
FirstName	Alexander	* JobId	Purchasing Clerk
* LastName	Khoo	DepartmentId	30
* Email	AKHOO	DepartmentName	Purchasing
Salary	4000	AnnualSalary	48000

9. Close the browser.

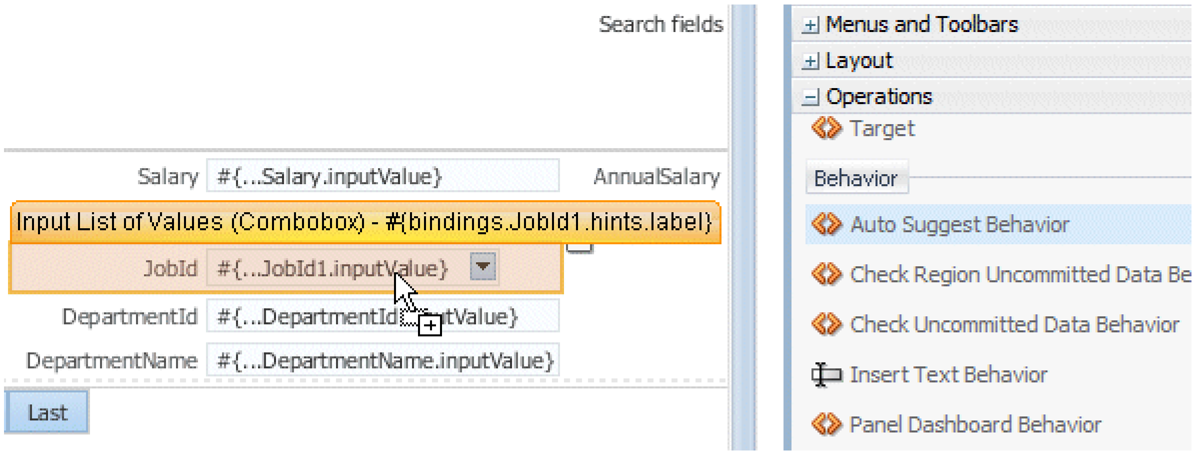
Step 3: Use the ADF Auto Suggest Behavior

The **af:autoSuggestBehavior** component displays a drop down list of suggested items when the user types into an input component. To use the auto-suggest functionality in a declarative way you need a model-driven list of values on your model project, which will be the base for the **suggestedItems** list. Earlier you added a list of values to the **JobId** field so in this example you use that field.

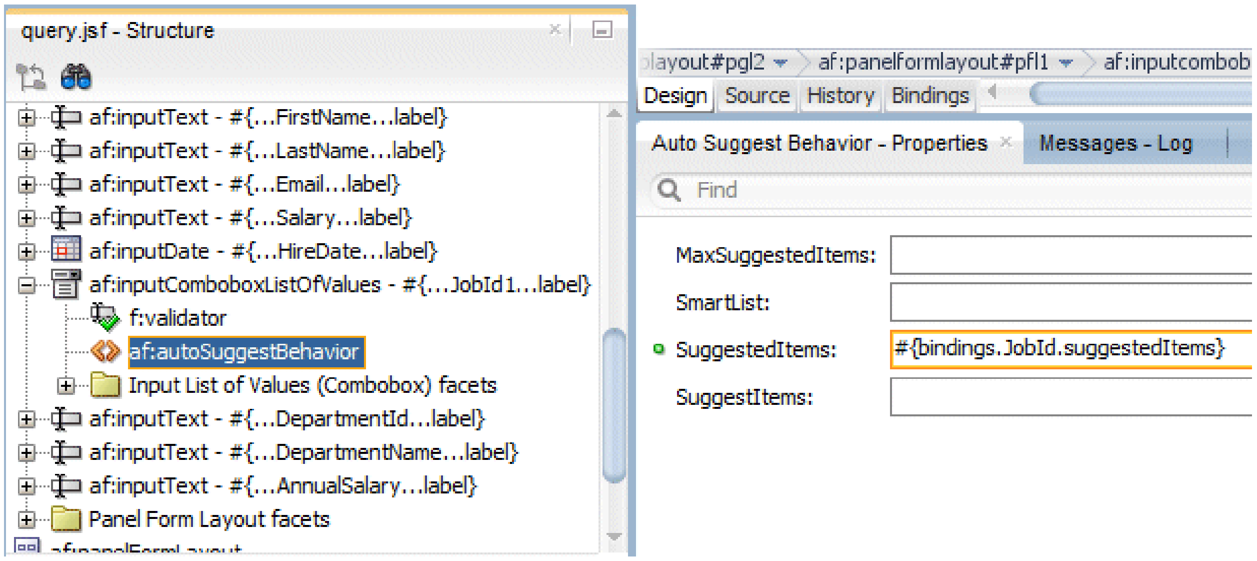
1. In the **query** page, select the **JobId** field.


adfc-config.xml
query.jsf
DeptEmpPage.jsf
Design
True
Focus
Employees
Search
Sear
EmployeeId #{...EmployeeId.inputValue}
FirstName #{...FirstName.inputValue}
LastName #{...LastName.inputValue}
Email #{...Email.inputValue}
Salary #{...Salary.inputValue}
HireDate #{...HireDate.inputValue}
JobId #{...JobId.inputValue}
DepartmentId #{...DepartmentId.inputValue}
DepartmentName #{...DepartmentName.inputValue}
AnnualSalary #{...AnnualSalary.inputValue}
Commit
Rollback
First
Previous
Next
Last
Submit
Back
af:gridcell#gc1
af:panelgrouplayout#pgl2
af:panelformlayout#pfl1
af:selectonechoice#soc1

2. In the Components window expand the **Operations** node, and locate **Auto Suggest Behavior** in the Behavior section. Drag and drop the **Auto Suggest Behavior** operation onto the **JobId** field.


Search fields
Salary #{...Salary.inputValue}
AnnualSalary
Input List of Values (Combobox) - #{bindings.JobId1.hints.label}
JobId #{...JobId1.inputValue}
DepartmentId #{...DepartmentId.inputValue}
DepartmentName #{...DepartmentName.inputValue}
Last
Menus and Toolbars
Layout
Operations
Target
Behavior
Auto Suggest Behavior
Check Region Uncommitted Data Be
Check Uncommitted Data Behavior
Insert Text Behavior
Panel Dashboard Behavior

3. Select the **af:autoSuggestBehavior** component in the Structure window. In the Properties window, set the SuggestedItems property to **#{bindings.JobId.suggestedItems}.** You can enter the value or use the Method Expression Builder by clicking the gear icon next to the property.

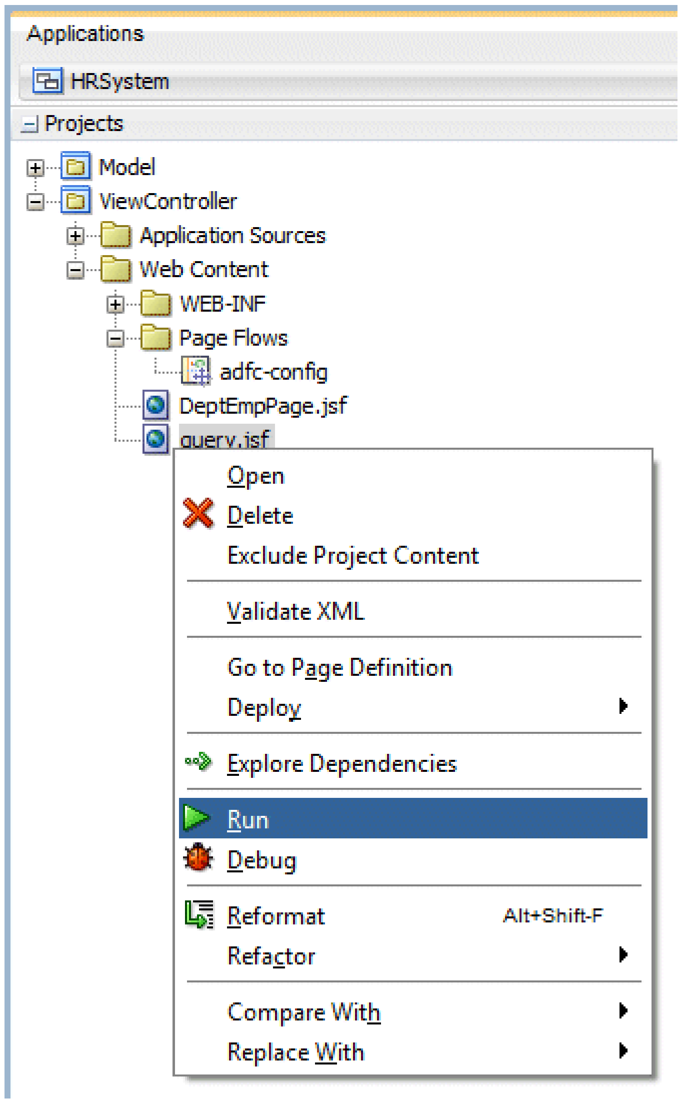

query.jsf - Structure

```
af:inputText - #{...FirstName...label}
af:inputText - #{...LastName...label}
af:inputText - #{...Email...label}
af:inputText - #{...Salary...label}
af:inputDate - #{...HireDate...label}
af:inputComboboxListOfValues - #{...JobId1...label}
    f:validator
    af:autoSuggestBehavior
    Input List of Values (Combobox) facets
af:inputText - #{...DepartmentId...label}
af:inputText - #{...DepartmentName...label}
af:inputText - #{...AnnualSalary...label}
Panel Form Layout facets
```

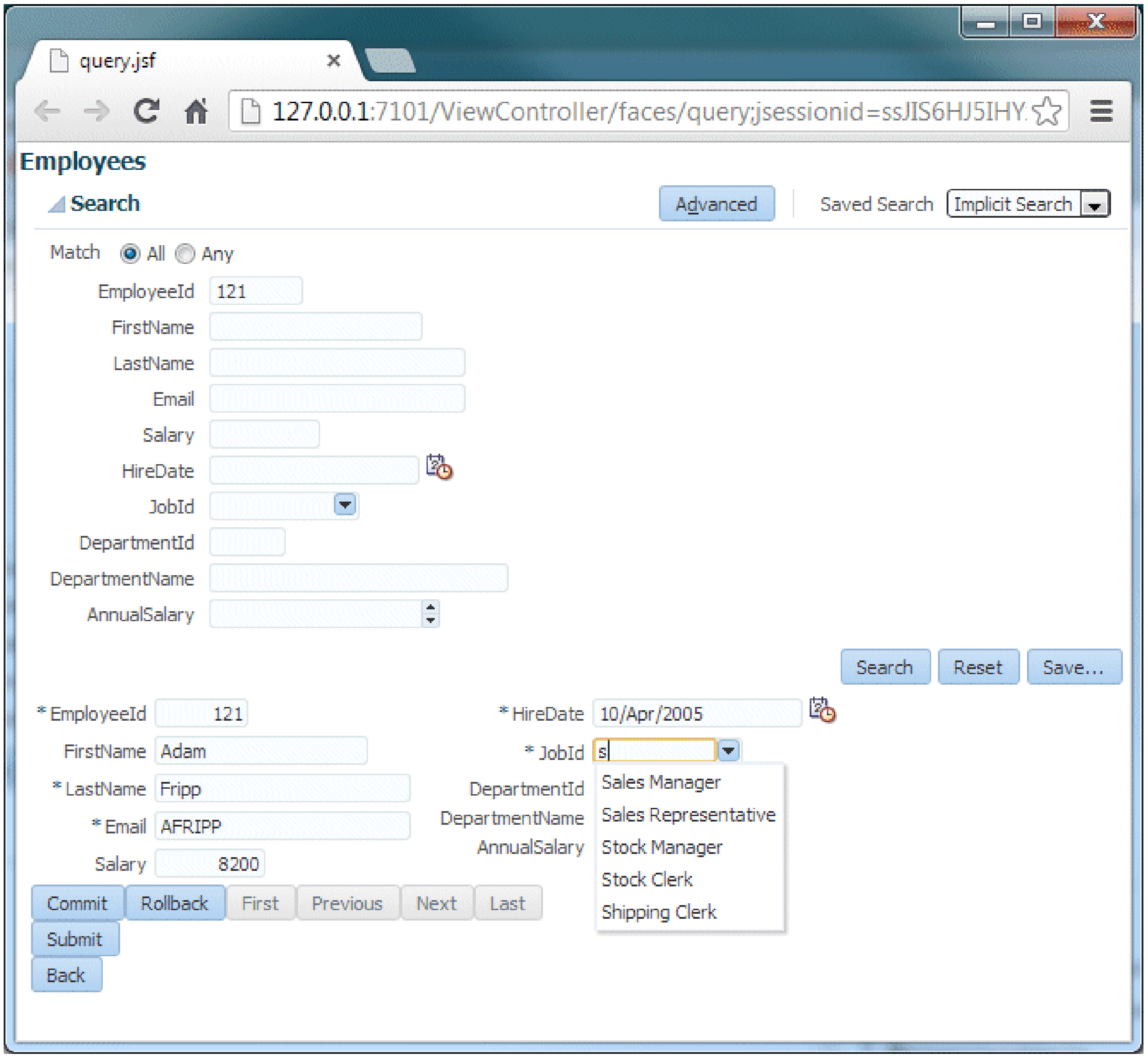
Auto Suggest Behavior - Properties

MaxSuggestedItems:	
SmartList:	
SuggestedItems:	#{bindings.JobId.suggestedItems}
SuggestItems:	

4. **Save** your work and then **Run** the **query** page.



5. Type **121** in the EmployeeId field, and click the **Search** button. In the record for Adam Fripp update the **JobId** field by typing **'s'** in it. A number of jobs beginning with 's' are suggested.


query.jsf

127.0.0.1:7101/ViewController/faces/query;jsessionid=ssJIS6HJ5IHY

Employees

Search

Advanced | Saved Search: Implicit Search

Match [✓ All] Any

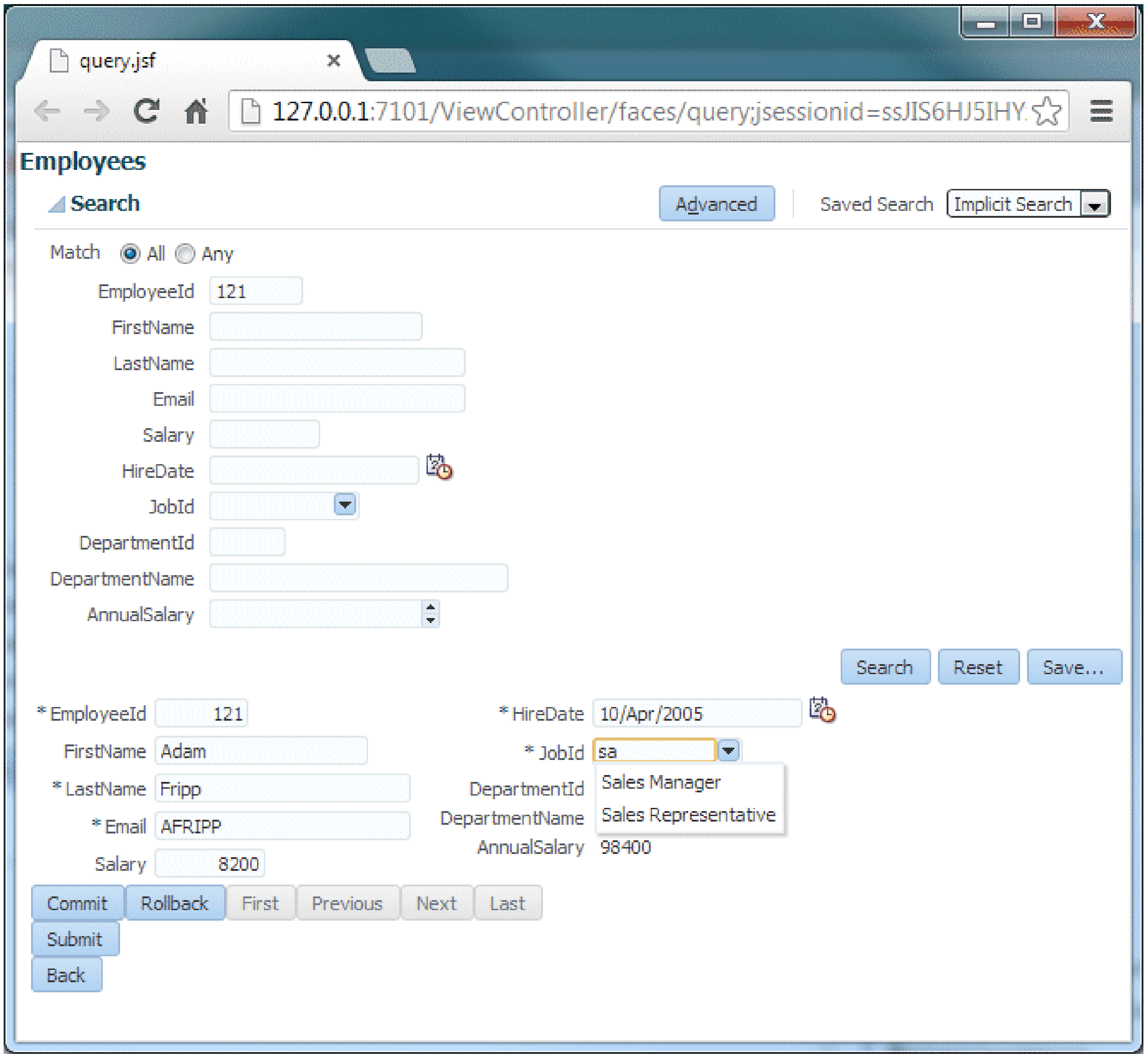
EmployeeId	121
FirstName	
LastName	
Email	
Salary	
HireDate	
JobId	
DepartmentId	
DepartmentName	
AnnualSalary	

Search | Reset | Save...

* EmployeeId	121	* HireDate	10/Apr/2005
FirstName	Adam	* JobId	s
* LastName	Fripp	DepartmentId	
* Email	AFRIPP	DepartmentName	
Salary	8200	AnnualSalary	

Sales Manager
Sales Representative
Stock Manager
Stock Clerk
Shipping Clerk

6. Add **'a'** after the **'s'** and see the list of suggestions modified accordingly.



Employees

Search

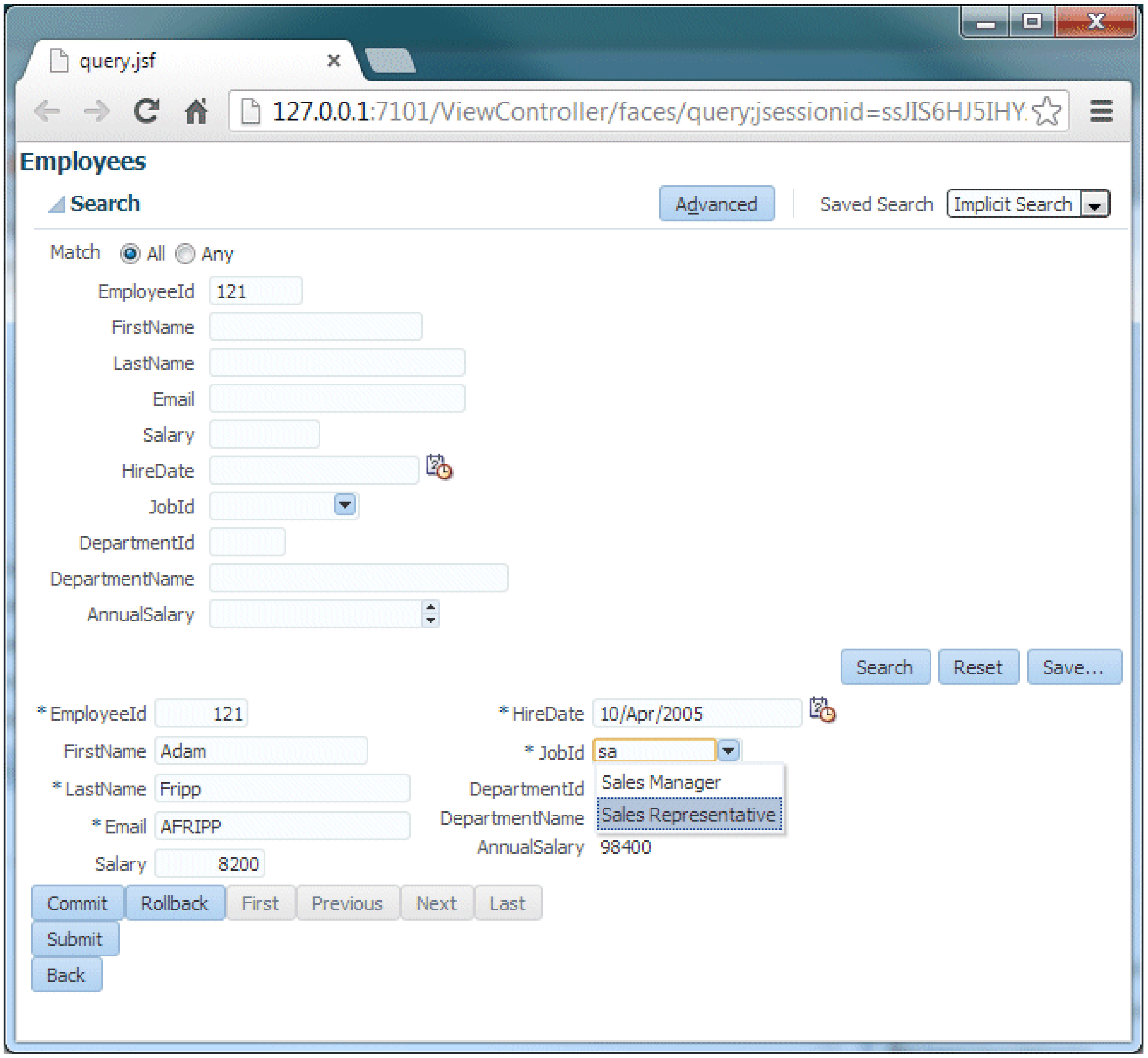
Saved Search: Implicit Search

Match [✓ All] Any

EmployeeId	121
FirstName	
LastName	
Email	
Salary	
HireDate	
JobId	
DepartmentId	
DepartmentName	
AnnualSalary	

* EmployeeId	121	* HireDate	10/Apr/2005
FirstName	Adam	* JobId	sa
* LastName	Fripp	DepartmentId	Sales Manager
* Email	AFRIPP	DepartmentName	Sales Representative
Salary	8200	AnnualSalary	98400

7. Choose **Sales Representative** from the remaining options, to populate the field.


query.jsf

127.0.0.1:7101/ViewController/faces/query;jsessionid=ssJIS6HJ5IHY

Employees

Search

Advanced | Saved Search: Implicit Search

Match [✓ All] Any

EmployeeId	121
FirstName	
LastName	
Email	
Salary	
HireDate	
JobId	
DepartmentId	
DepartmentName	
AnnualSalary	

Search | Reset | Save...

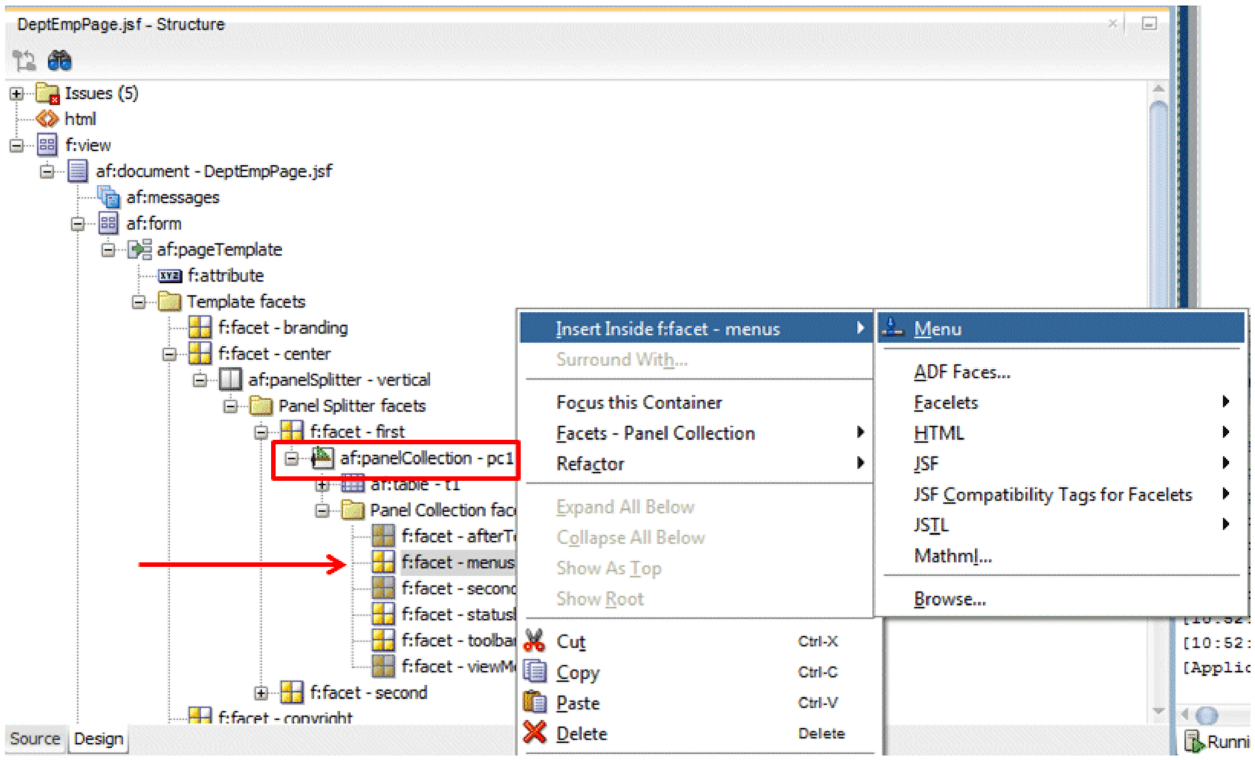
* EmployeeId	121	* HireDate	10/Apr/2005
FirstName	Adam	* JobId	sa
* LastName	Fripp	DepartmentId	
* Email	AFRIPP	DepartmentName	
Salary	8200	AnnualSalary	98400

8. Close the browser without saving the change.

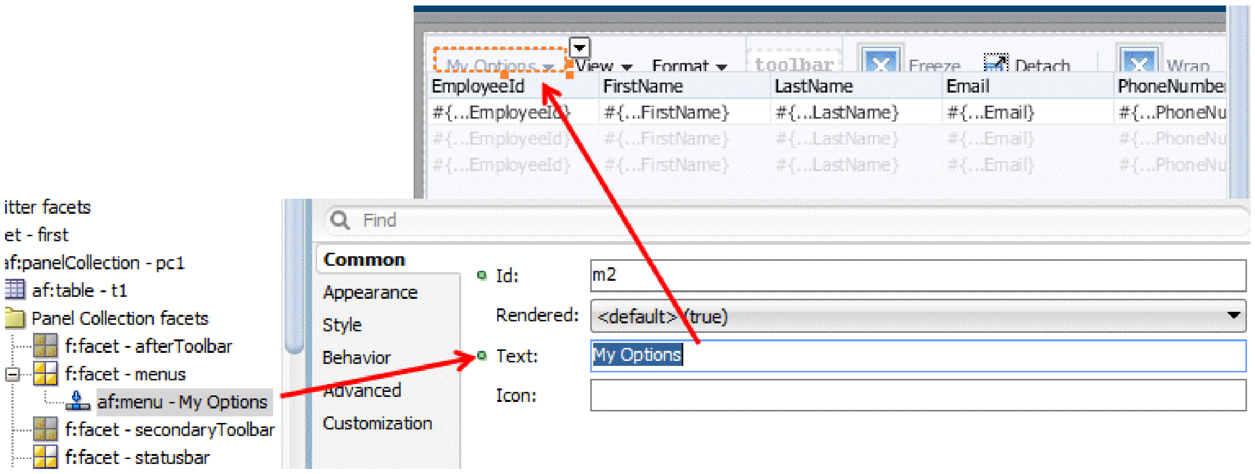
Step 4: Use Drop Down Menus and Operation Components

In this step you add a drop down menu to a page and use a couple of ADF Faces operation components to add Javascript-based operations to the page. On component will export table data into an Excel spreadsheet and the other will create a printable page.

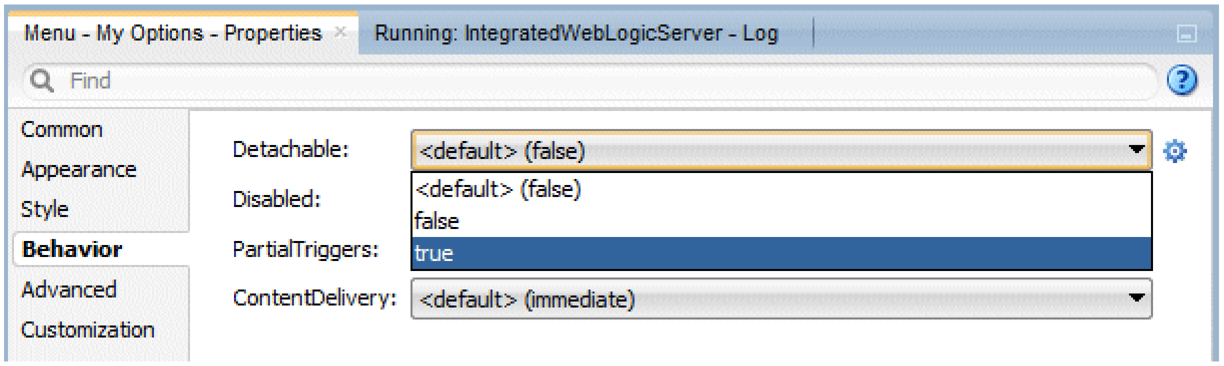
1. In the **DeptEmpPage.jsf** file click inside the **menus** facet in the panel collection surrounding the Employees table. Right-click and from the context menu choose **Insert Inside Facet menus > Menu**.



2. In the Properties window set the **Text** property to **My Options**.



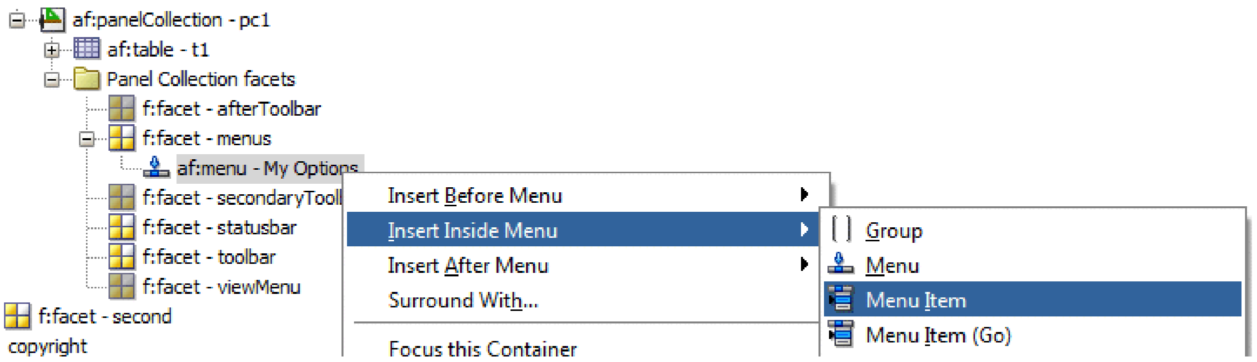
3. In the Properties window expand the **Behavior** node and set the **Detachable** property to **true**.


Menu - My Options - Properties
Running: IntegratedWebLogicServer - Log
Find
Common
Appearance
Style
Behavior
Advanced
Customization

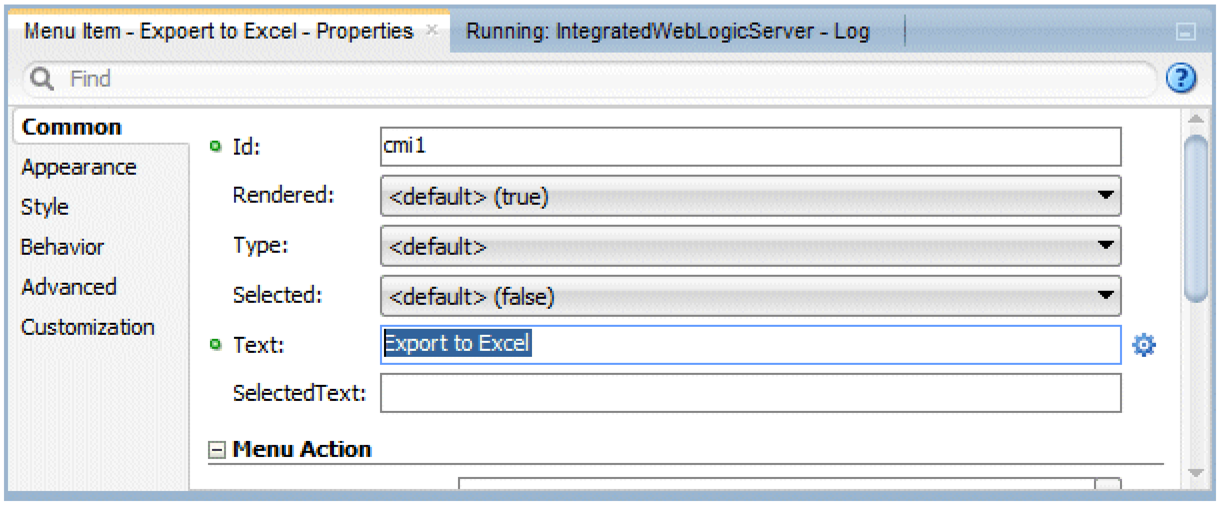
Detachable: <default> (false)
Disabled:
PartialTriggers:
ContentDelivery: <default> (immediate)

<default> (false)
false
true

4. In the Structure window right-click the menu component and choose **Insert inside af:menu - My Options > Menu Item**.

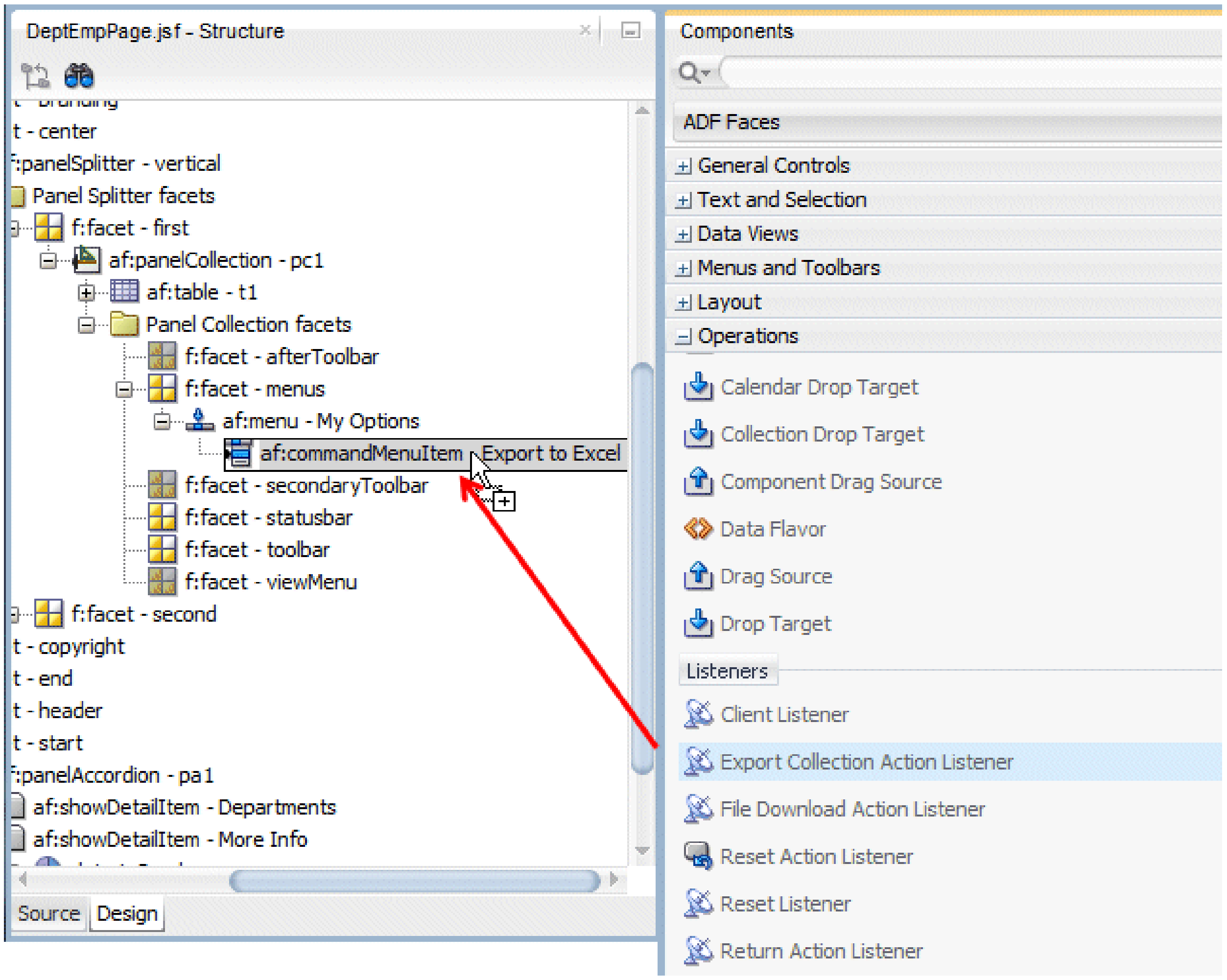


5. In the Properties window set the **Text** property of the new menu item to **Export to Excel**.

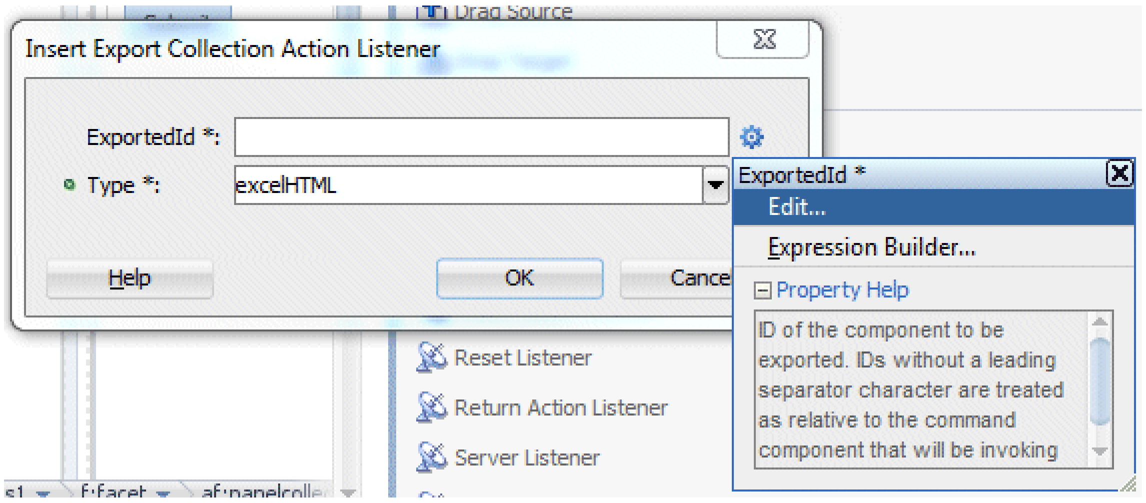

Menu Item - Expoert to Excel - Properties
Running: IntegratedWebLogicServer - Log
Find
Common
Appearance
Style
Behavior
Advanced
Customization
Menu Action

Id: cmi1
Rendered: <default> (true)
Type: <default>
Selected: <default> (false)
Text: Export to Excel
SelectedText:

6. With the new Export to Excel menu item still selected in the Structure window, expand the **Listeners** section of the **Operations** node of the the ADF Faces components in the Components window.
Locate the **Export Collection Action Listener** component and drag it onto the **Export to Excel** menu option in the Structure window.

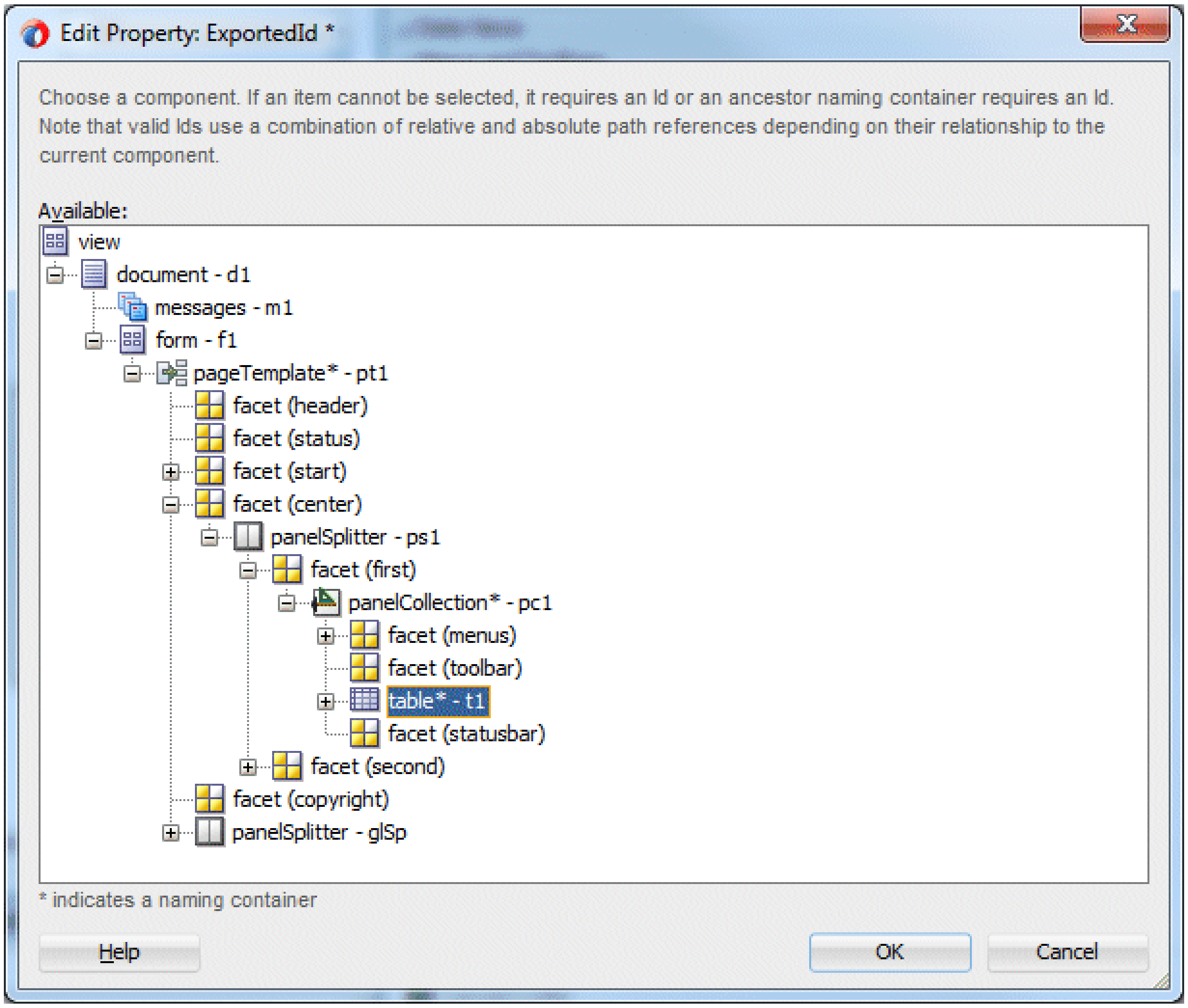


7. In the **Insert Export Collection Action Listener** dialog click the gear icon next to **ExportedId** field and choose **Edit**.



ID of the component to be exported. IDs without a leading separator character are treated as relative to the command component that will be invoking

8. In the Edit Property dialog navigate through the page structure to locate the **table - t1** component and select it. Click **OK**.


Edit Property: ExportedId *

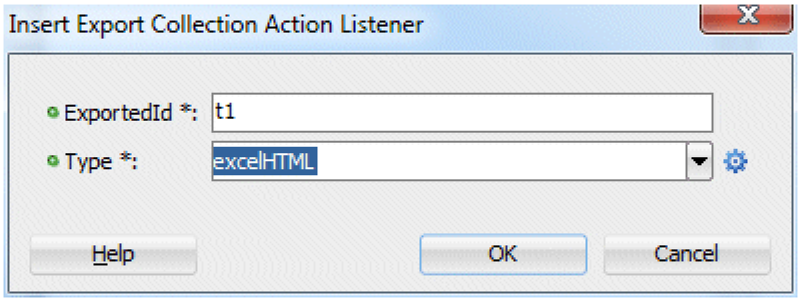
Choose a component. If an item cannot be selected, it requires an Id or an ancestor naming container requires an Id. Note that valid Ids use a combination of relative and absolute path references depending on their relationship to the current component.

Available:

```
view
document - d1
    messages - m1
    form - f1
        pageTemplate* - pt1
            facet (header)
            facet (status)
            facet (start)
            facet (center)
                panelSplitter - ps1
                    facet (first)
                        panelCollection* - pc1
                            facet (menus)
                            facet (toolbar)
                            table* - t1
                            facet (statusbar)
                    facet (second)
            facet (copyright)
            panelSplitter - glSp
```

* indicates a naming container

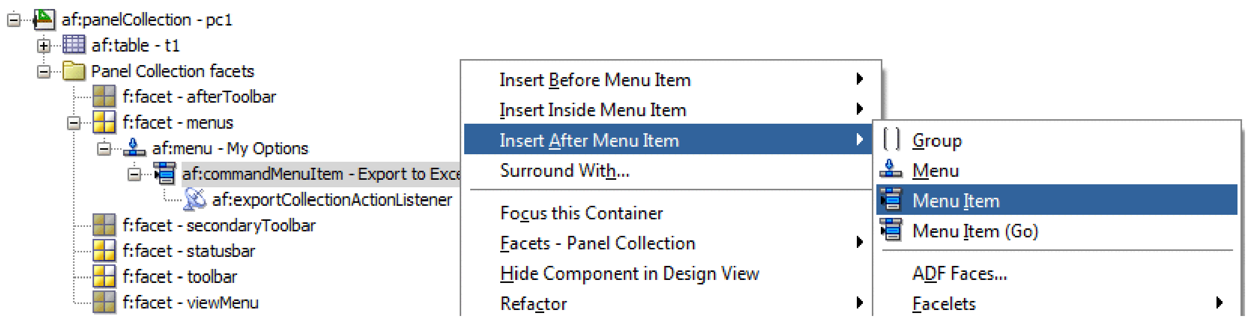
9. From the **Type** drop down list select **excelHTML** and click **OK**.


Insert Export Collection Action Listener

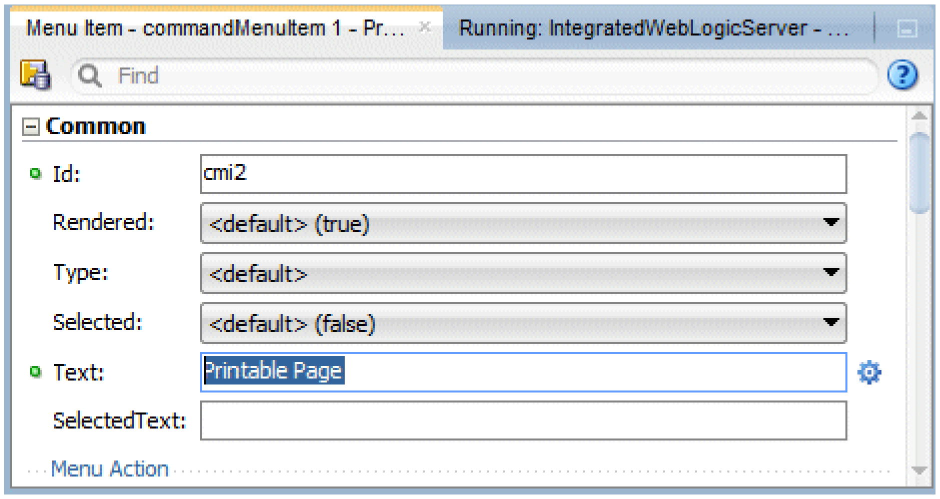
ExportedId *: t1

Type *: excelHTML

10. Add another menu option to the menu. In the Structure window right-click the **Export to Excel** menu component and from the context menu choose **Insert After Menu Item > Menu Item**.

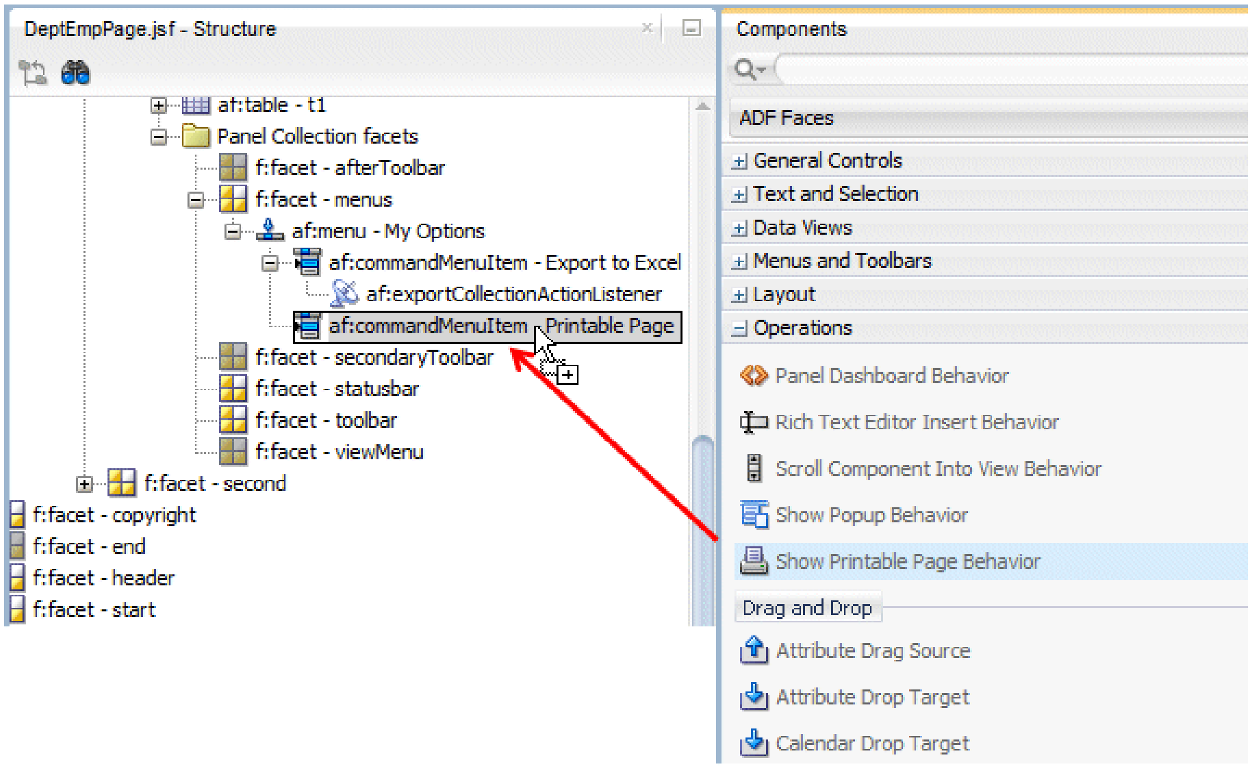


11. Set the **Text** property of this new menu option to **Printable Page**.


Menu Item - commandMenuItem 1 - Pr...
Running: IntegratedWebLogicServer - ...
Find
Common
Menu Action

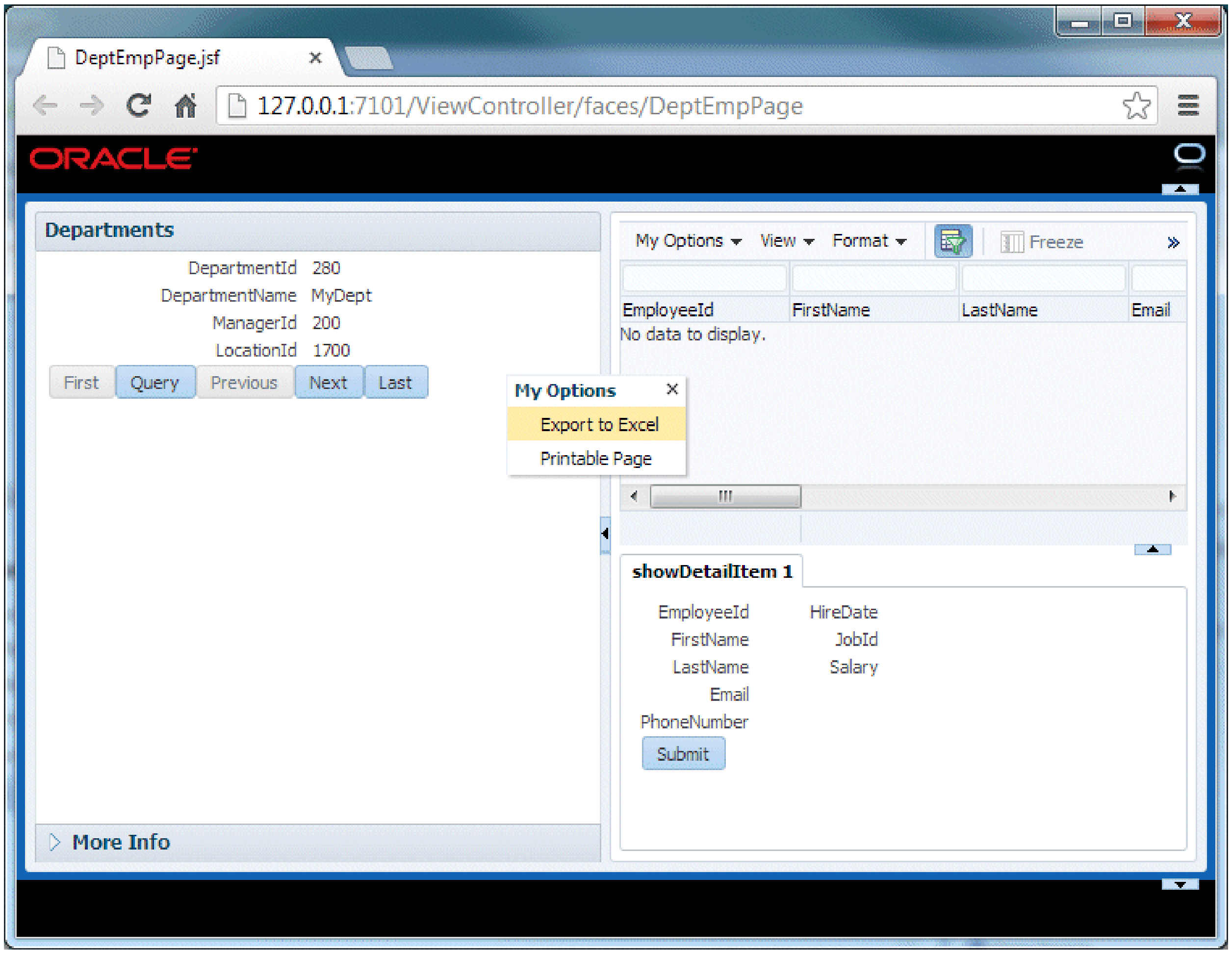
Id: cmi2
Rendered: <default> (true)
Type: <default>
Selected: <default> (false)
Text: Printable Page
SelectedText:

12. In the Behavior section under the Operations node of the Components window select the **Show Printable Page Behavior** operation to add it to the new menu item. Drag and drop it onto the new menu option you created.

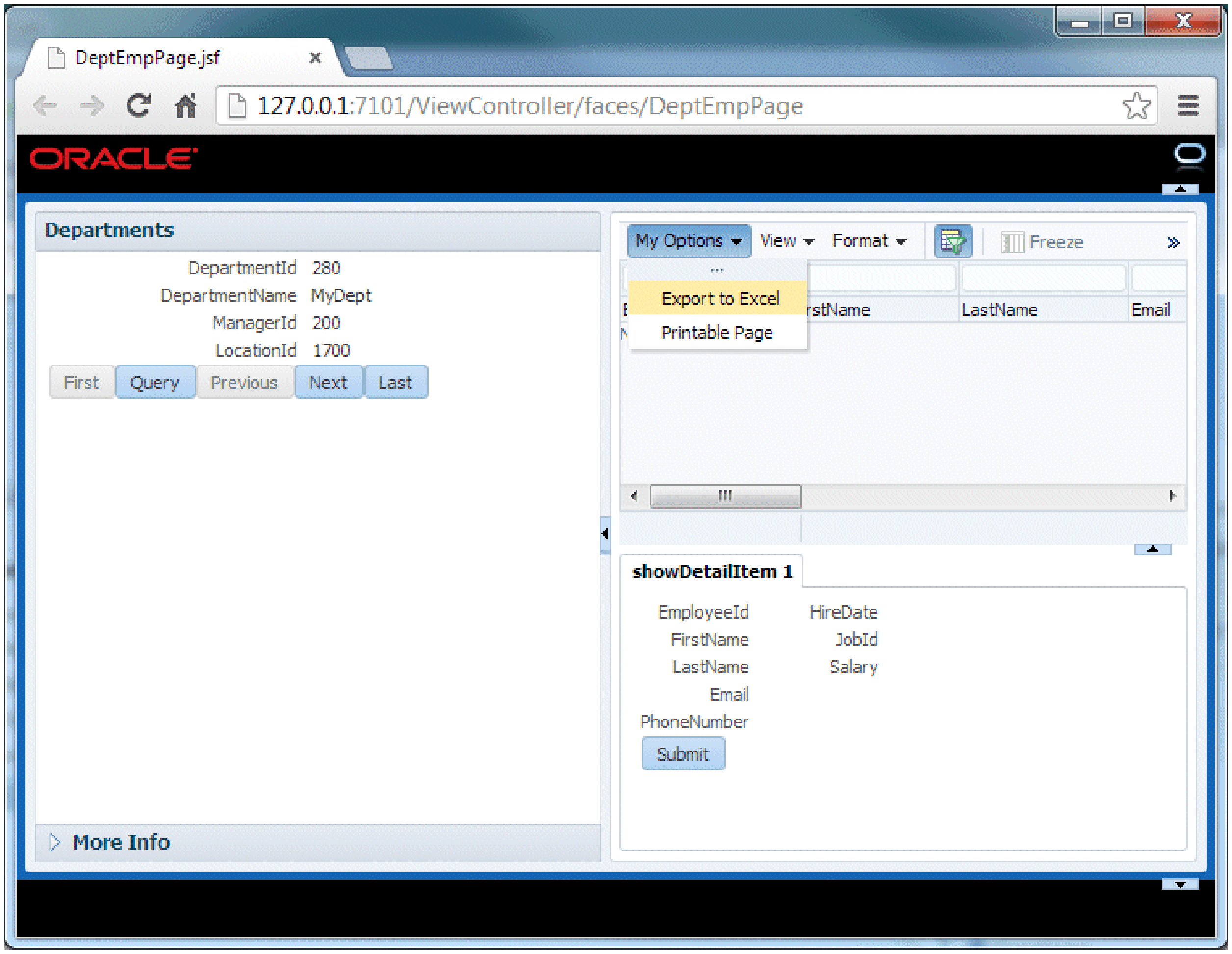


13. Click the **Save All** icon on the JDeveloper menu bar to save your work, and then choose **Run**.
14. When the page displays click the new menu and detach it.

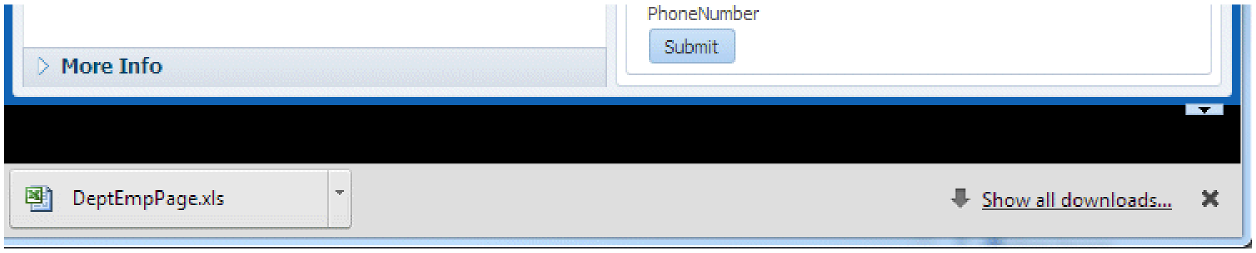
    Click the '**x**' to close the menu.



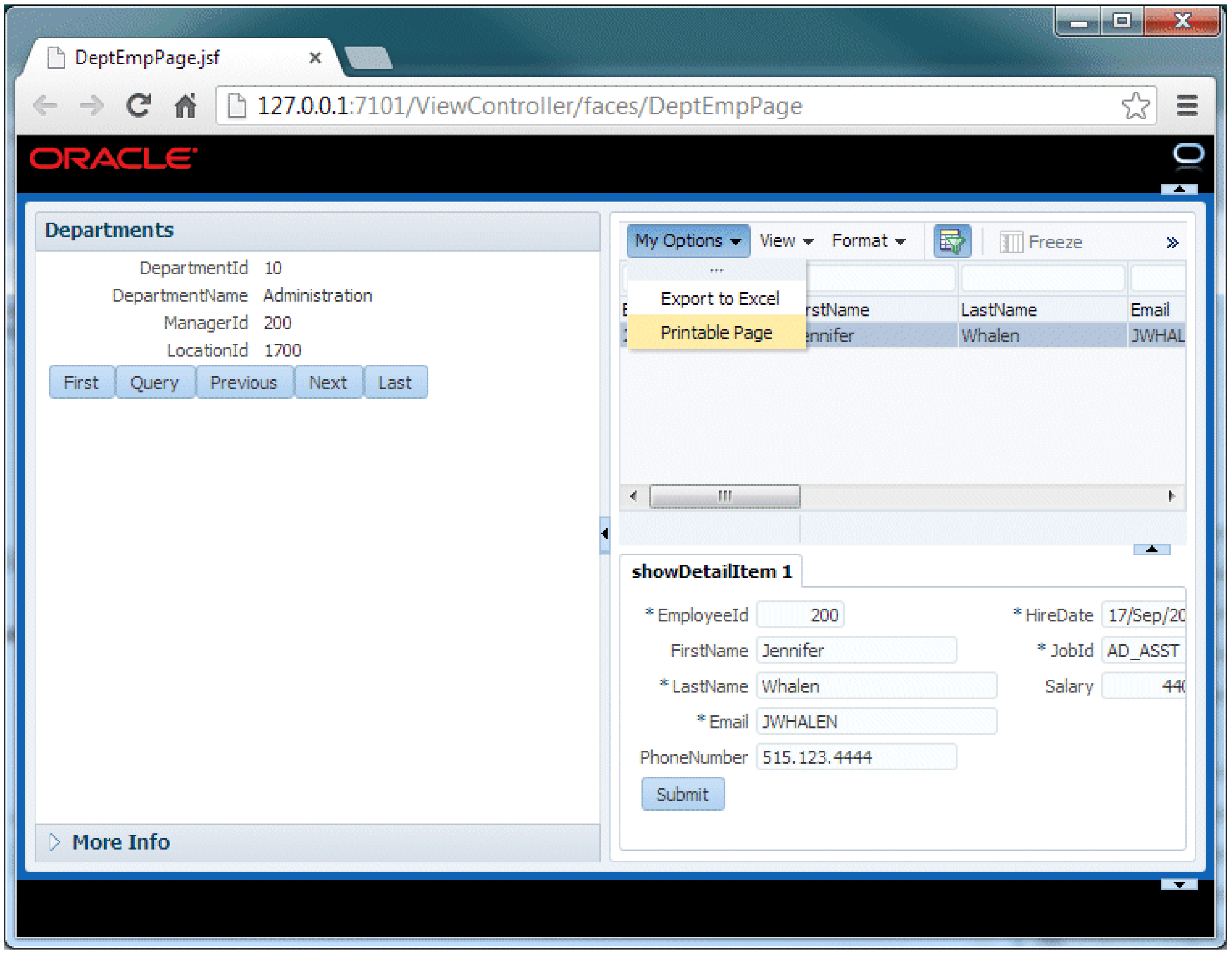
15. Then invoke each one of the menu options you created, for example **Export to Excel**.



You may need to accept the download of the file in the browser window to be able to access the Excel file, and it should be found in your default **'download'**. directory..



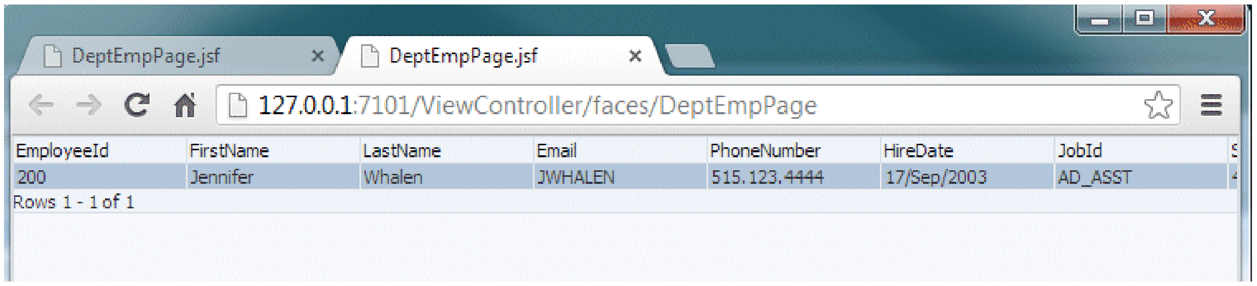
16. Try the **Printable Page** menu option.



DepartmentId	10
DepartmentName	Administration
ManagerId	200
LocationId	1700

* EmployeeId	200	* HireDate	17/Sep/20
FirstName	Jennifer	* JobId	AD_ASST
* LastName	Whalen	Salary	44
* Email	JWHALEN		
PhoneNumber	515.123.4444		

The page is ready for printing.



EmployeeId	FirstName	LastName	Email	PhoneNumber	HireDate	JobId
200	Jennifer	Whalen	JWHALEN	515.123.4444	17/Sep/2003	AD_ASST

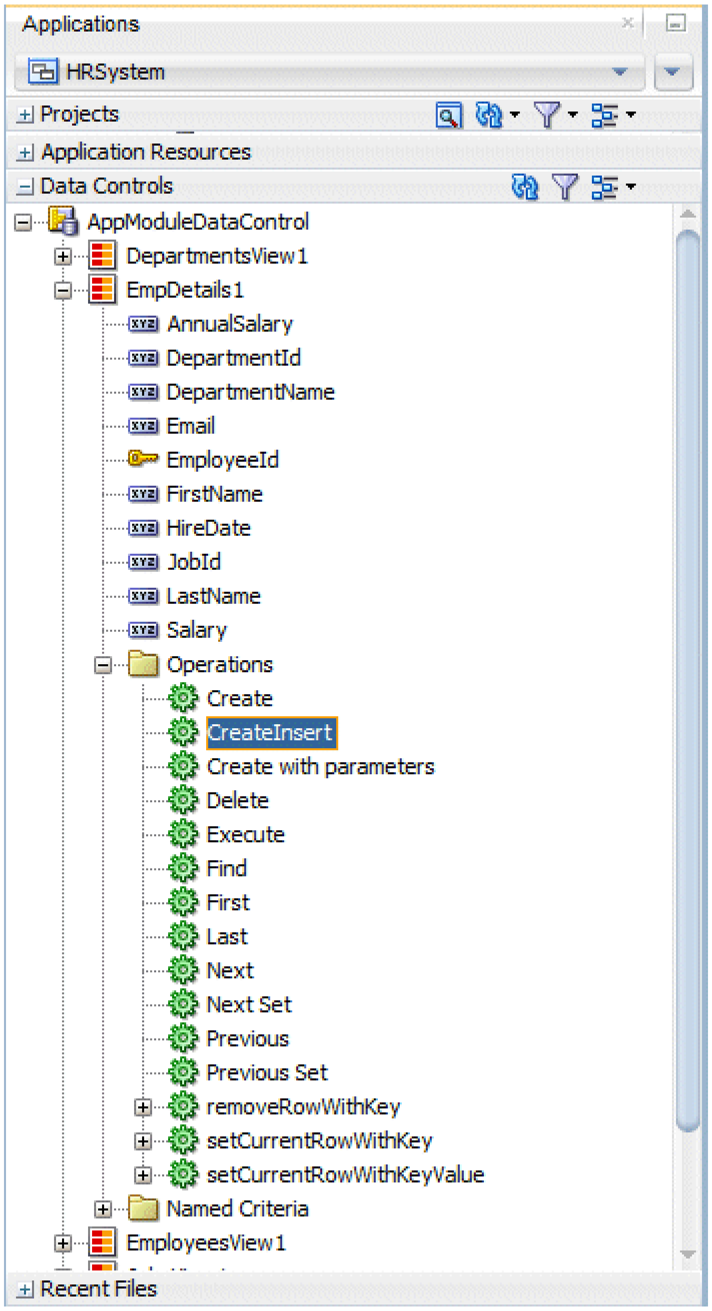
Rows 1 - 1 of 1

Close the browser window.

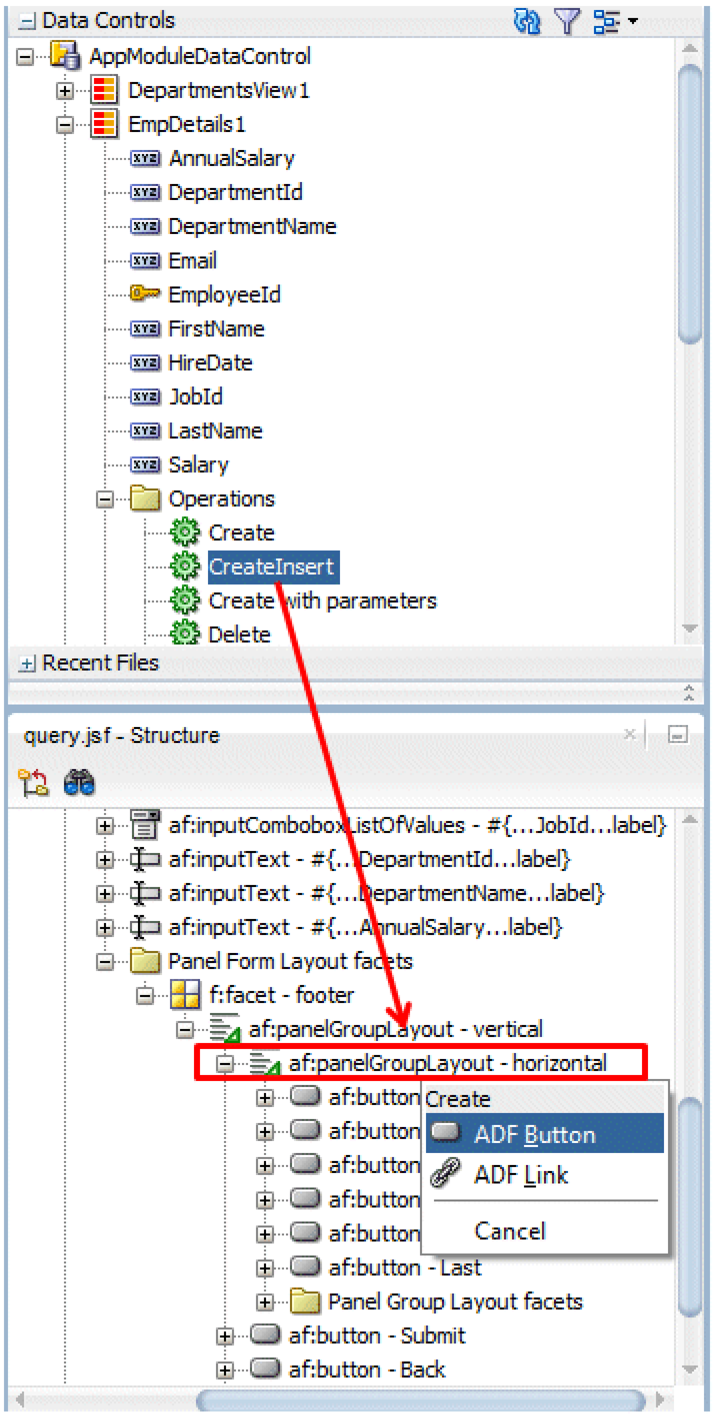
Step 5: Add CRUD Operation Components to your Page

The next few tasks examine some of the data operations that JDeveloper makes available to view objects. You see how to add a Delete operation and a CreateInsert operation. When the user clicks the CreateInsert button to insert the new row you want the table to refresh to display the new empty row. To do this you again use the Partial Page Refresh feature that was covered in Step 2 of this part of the tutorial.The view object uses a bind variable to pass the employee's email into the query.

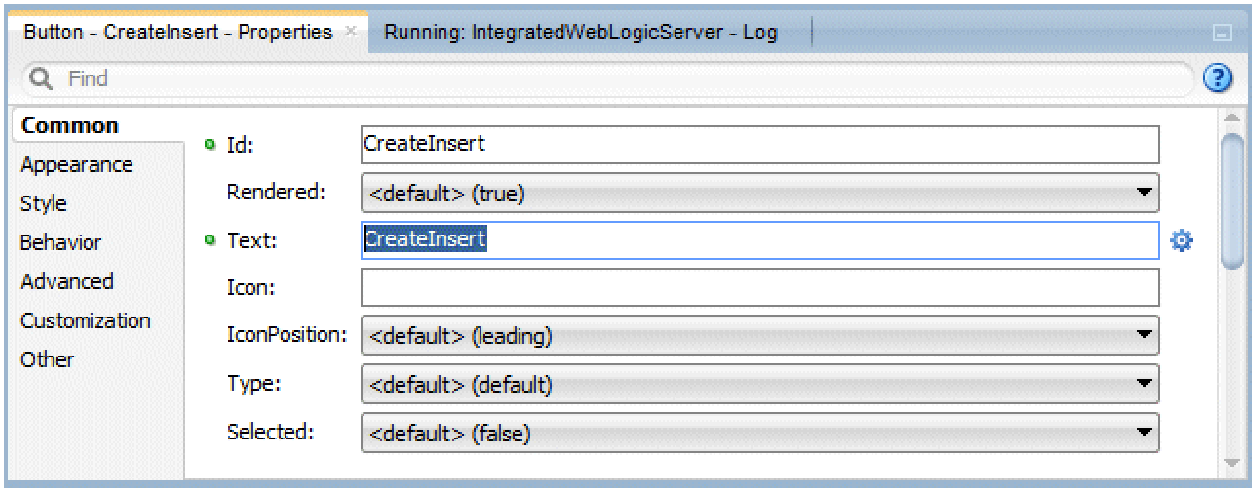
1. Click the **query.jsf** tab to return to the Query page, opening the Structure window. In the Data Controls accordion expand the **EmpDetails1** node and then the **Operations** node below it. Select the **CreateInsert** operation.



2. Drag the **CreateInsert** operation in the Structure window onto the **af:panelGroupLayout - horizontal** in the footer facet of the employees form. Create it as an **ADF Button**. Hint: expand af:panelFormlayout -5 > Panel Form Layout Facets > af:panelGroupLayout - vertical > af:panelGroupLayout - horizontal.

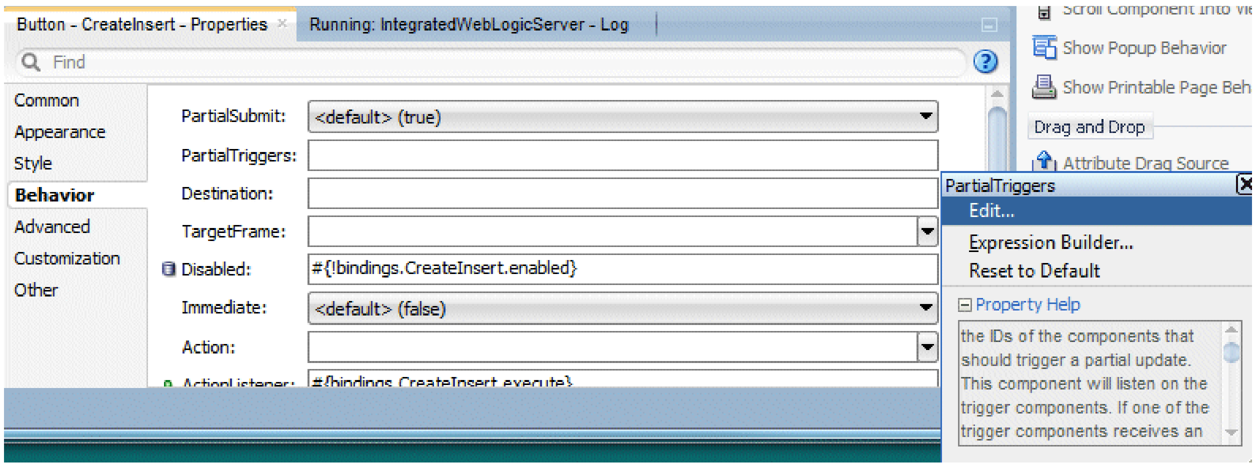


3. In the Properties window set the **Id** property for the button to **CreateInsert**.


Button - CreateInsert - Properties
Running: IntegratedWebLogicServer - Log
Find
Common
Appearance
Style
Behavior
Advanced
Customization
Other

Id: CreateInsert
Rendered: <default> (true)
Text: CreateInsert
Icon:
IconPosition: <default> (leading)
Type: <default> (default)
Selected: <default> (false)

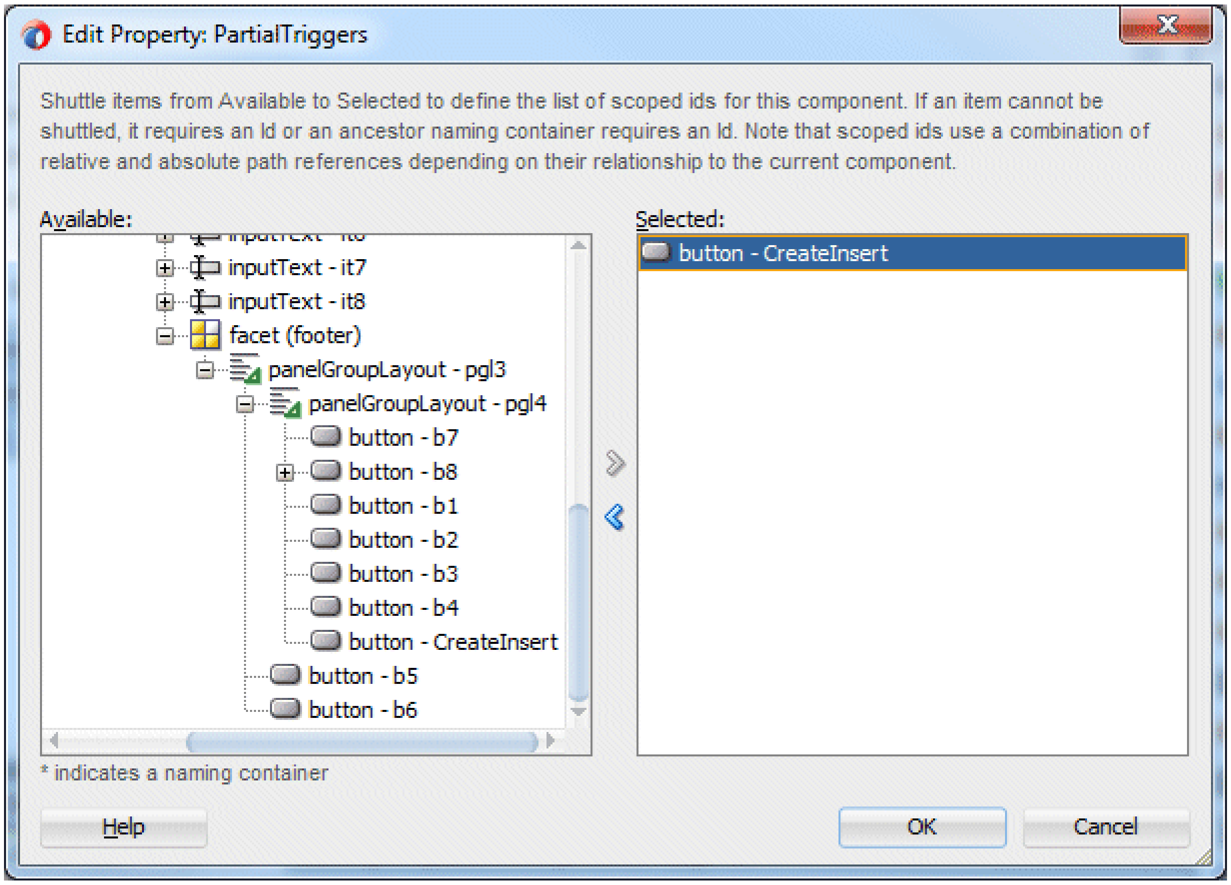
4. Still in the Properties window expand the **Behavior** node and in the **PartialTriggers** property choose **Edit** from the drop down list.



PartialSubmit: <default> (true)
PartialTriggers:
Destination:
TargetFrame:
Disabled: #{!bindings.CreateInsert.enabled}
Immediate: <default> (false)
Action:
ActionListener: #{bindings.CreateInsert.execute}

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

5. In the Edit window scroll through the page's components until you find the **CreateInsert** button. Shuttle it into the **Selected** pane.


Edit Property: PartialTriggers

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.

Available:

- inputText - it7
- inputText - it8
- facet (footer)
  - panelGroupLayout - pgl3
    - panelGroupLayout - pgl4
      - button - b7
      - button - b8
      - button - b1
      - button - b2
      - button - b3
      - button - b4
      - button - CreateInsert
    - button - b5
    - button - b6

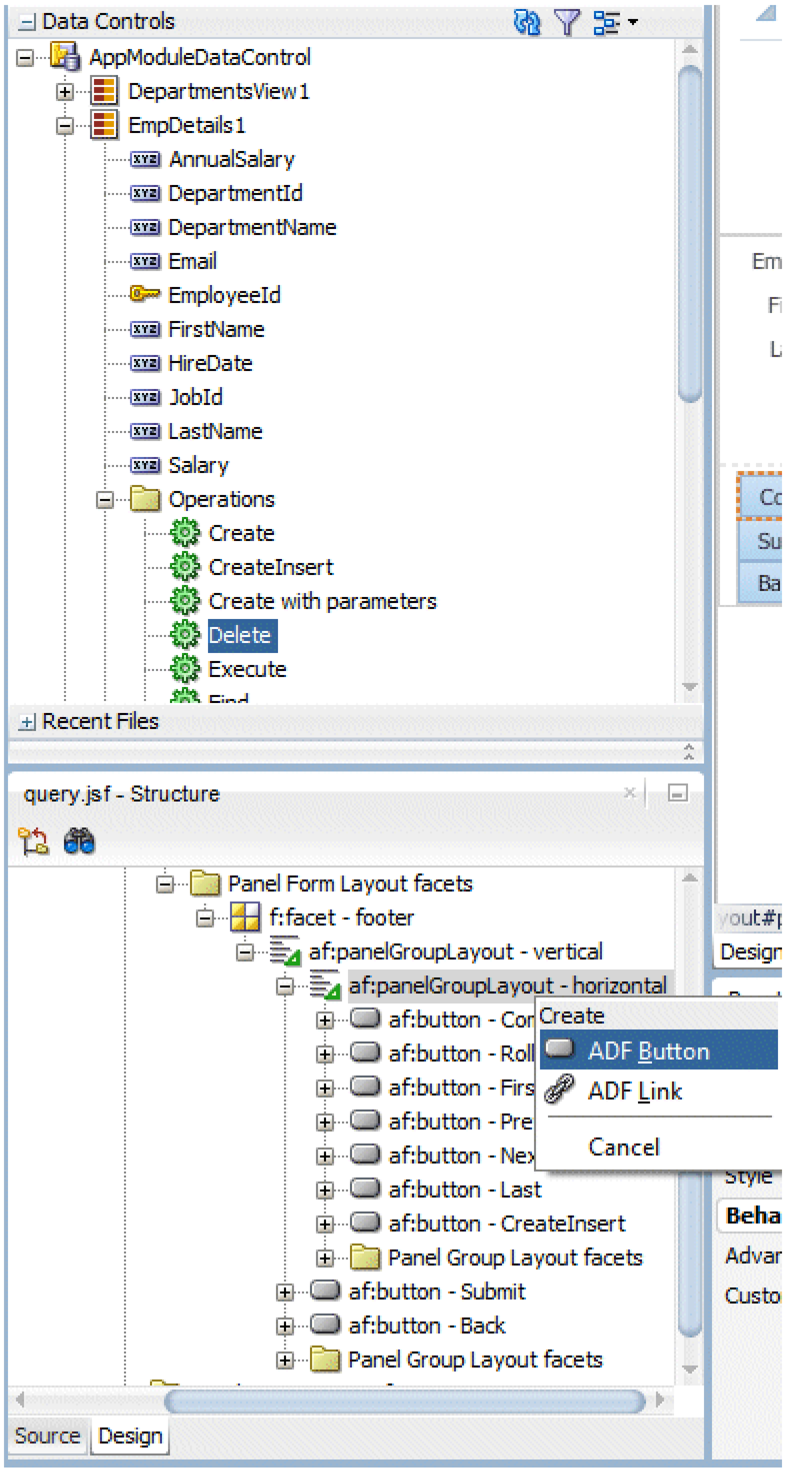
Selected:

- button - CreateInsert

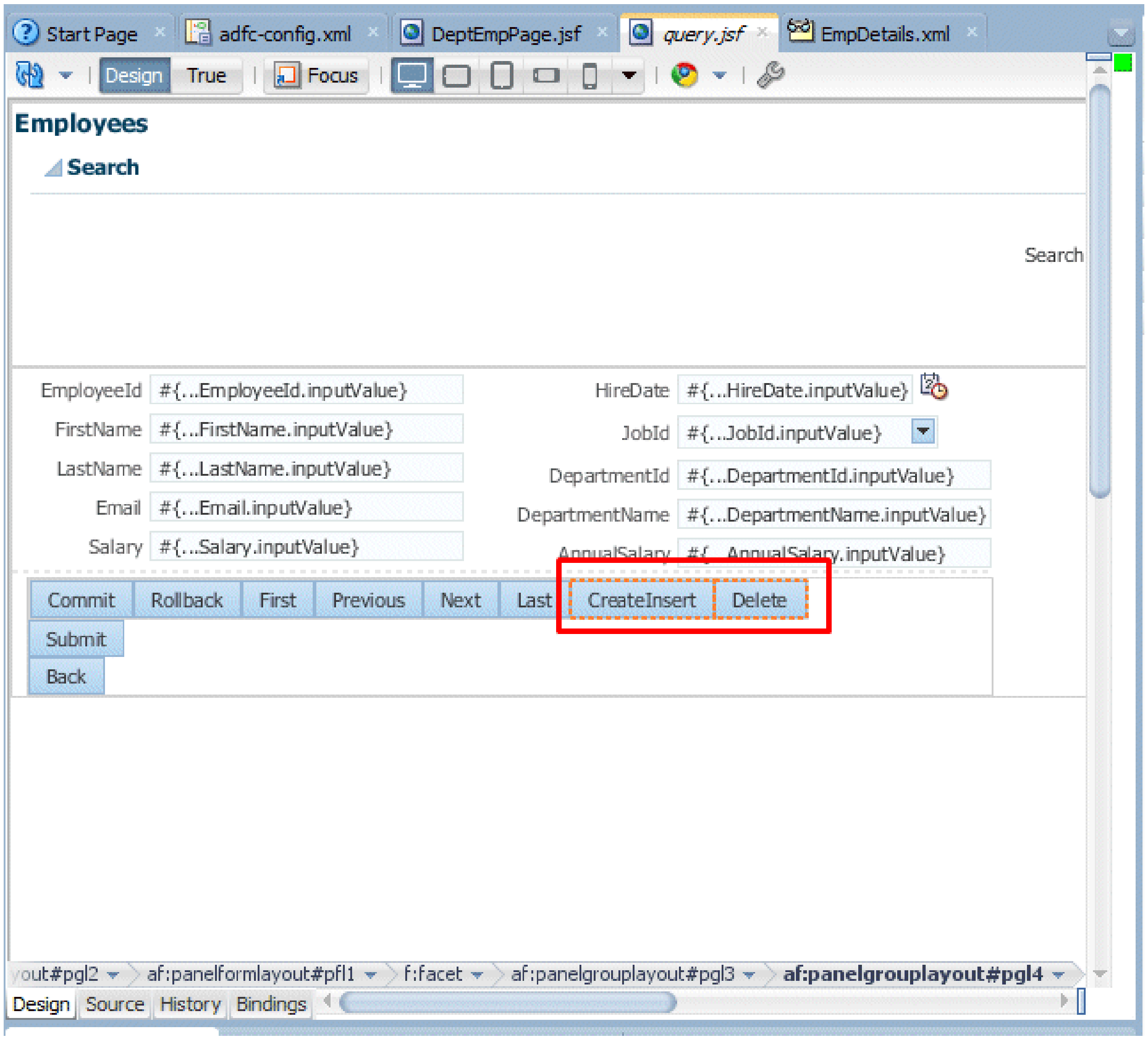
* indicates a naming container

Click **OK**. This action defines the **CreateInsert** component as the trigger that will cause the table to refresh.

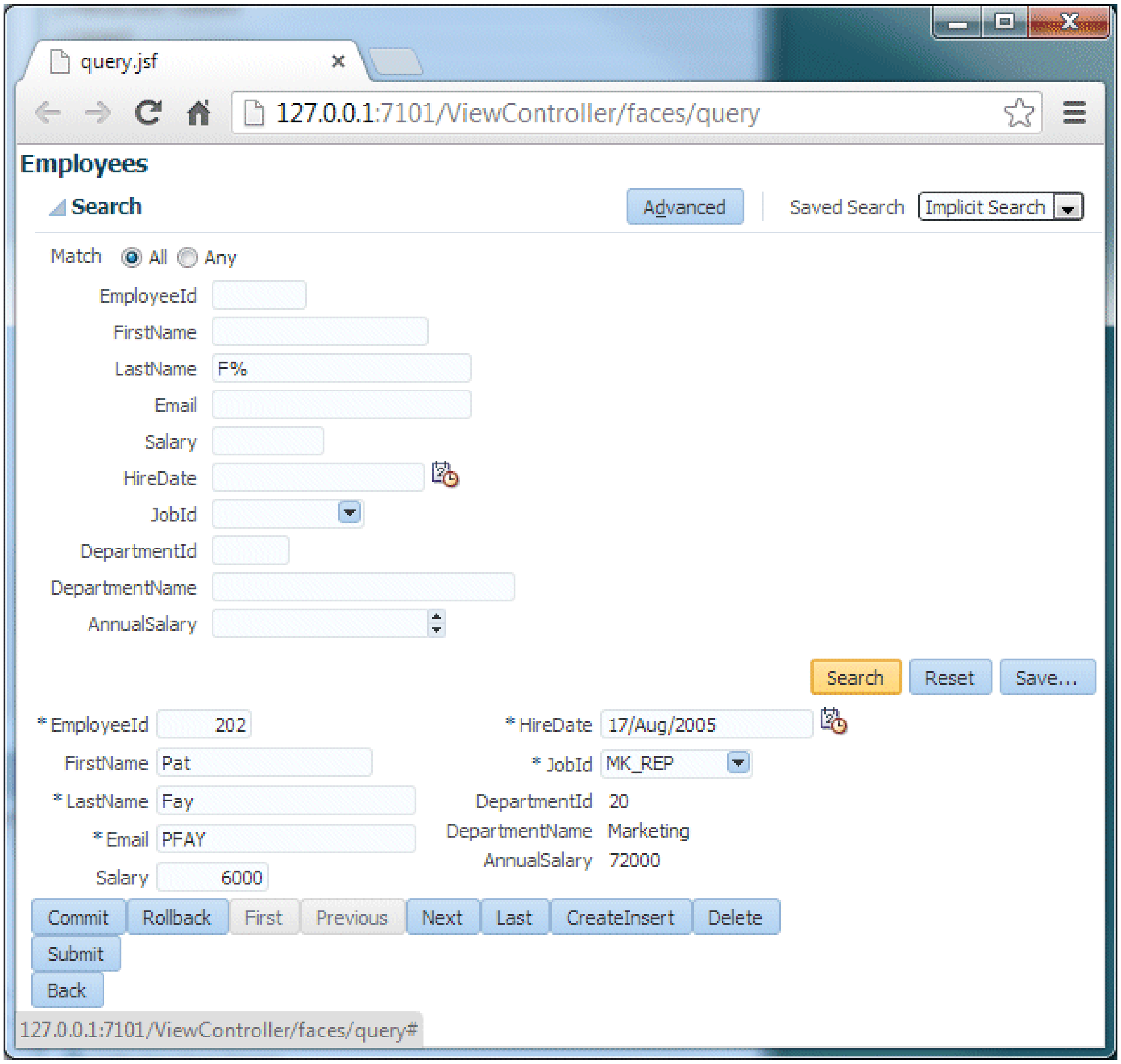
6. In the same way add a Delete operation by dropping the **Delete** operation from **EmpDetails1** onto the panelGroupLayout in the page footer. As before, create it as an ADF button.



7. The two new buttons display at the bottom of the query page.


Start Page
adfc-config.xml
DeptEmpPage.jsf
query.jsf
EmpDetails.xml
Design
True
Focus
Employees
Search
Search
EmployeeId #{...EmployeeId.inputValue}
FirstName #{...FirstName.inputValue}
LastName #{...LastName.inputValue}
Email #{...Email.inputValue}
Salary #{...Salary.inputValue}
HireDate #{...HireDate.inputValue}
JobId #{...JobId.inputValue}
DepartmentId #{...DepartmentId.inputValue}
DepartmentName #{...DepartmentName.inputValue}
AnnualSalary #{...AnnualSalary.inputValue}
Commit
Rollback
First
Previous
Next
Last
CreateInsert
Delete
Submit
Back
yout#pgl2 > af:panelformlayout#pfl1 > f:facet > af:panelgrouplayout#pgl3 > af:panelgrouplayout#pgl4
Design
Source
History
Bindings

8. **Save** your work and then **Run** the Query page.
9. When the page displays type **F%** in the **LastName** field and click the **Search** button. The first F% employee record displays.


query.jsf

127.0.0.1:7101/ViewController/faces/query

Employees

Search

Saved Search: Implicit Search

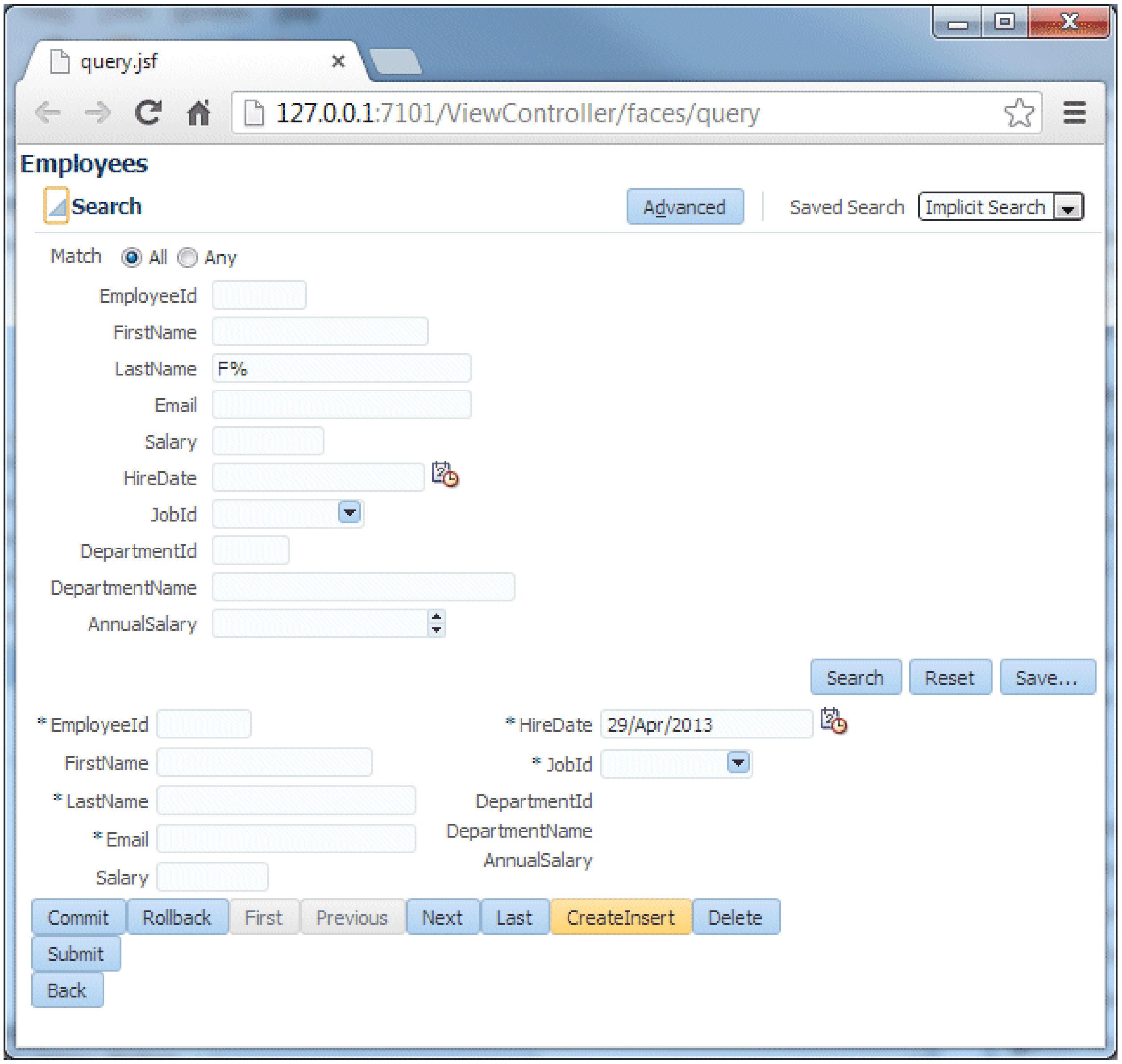
Match: [✓ All] Any

EmployeeId	
FirstName	
LastName	F%
Email	
Salary	
HireDate	
JobId	
DepartmentId	
DepartmentName	
AnnualSalary	

* EmployeeId	202	* HireDate	17/Aug/2005
FirstName	Pat	* JobId	MK_REP
* LastName	Fay	DepartmentId	20
* Email	PFAY	DepartmentName	Marketing
Salary	6000	AnnualSalary	72000

127.0.0.1:7101/ViewController/faces/query#

10. Click the **CreateInsert** button. The page refreshes and the fields are cleared (except for the HireDate field, which you set to default to the current date) so that a new record can be inserted.


query.jsf

127.0.0.1:7101/ViewController/faces/query

Employees

Search

Saved Search: Implicit Search

Match [✓ All] Any

EmployeeId	
FirstName	
LastName	F%
Email	
Salary	
HireDate	
JobId	
DepartmentId	
DepartmentName	
AnnualSalary	

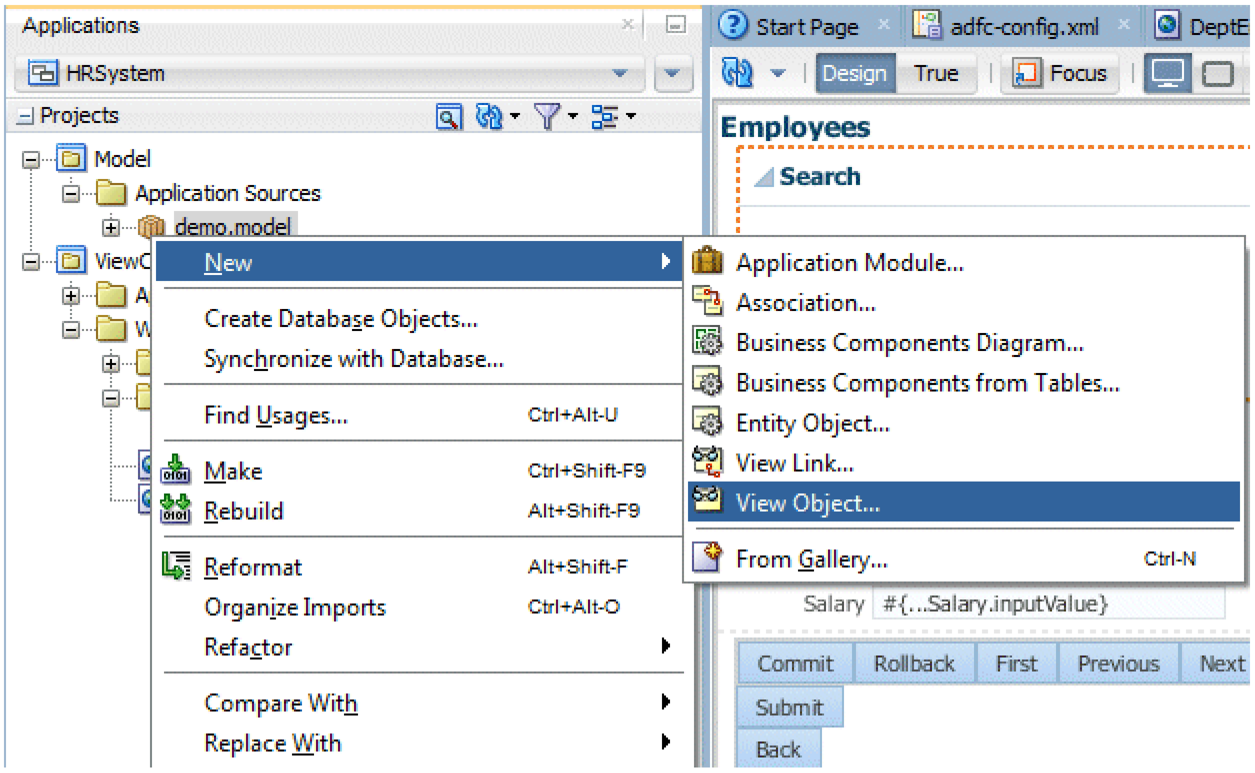
* EmployeeId		* HireDate	29/Apr/2013
FirstName		* JobId	
* LastName		DepartmentId	
* Email		DepartmentName	
Salary		AnnualSalary	

11. Close the browser without committing.

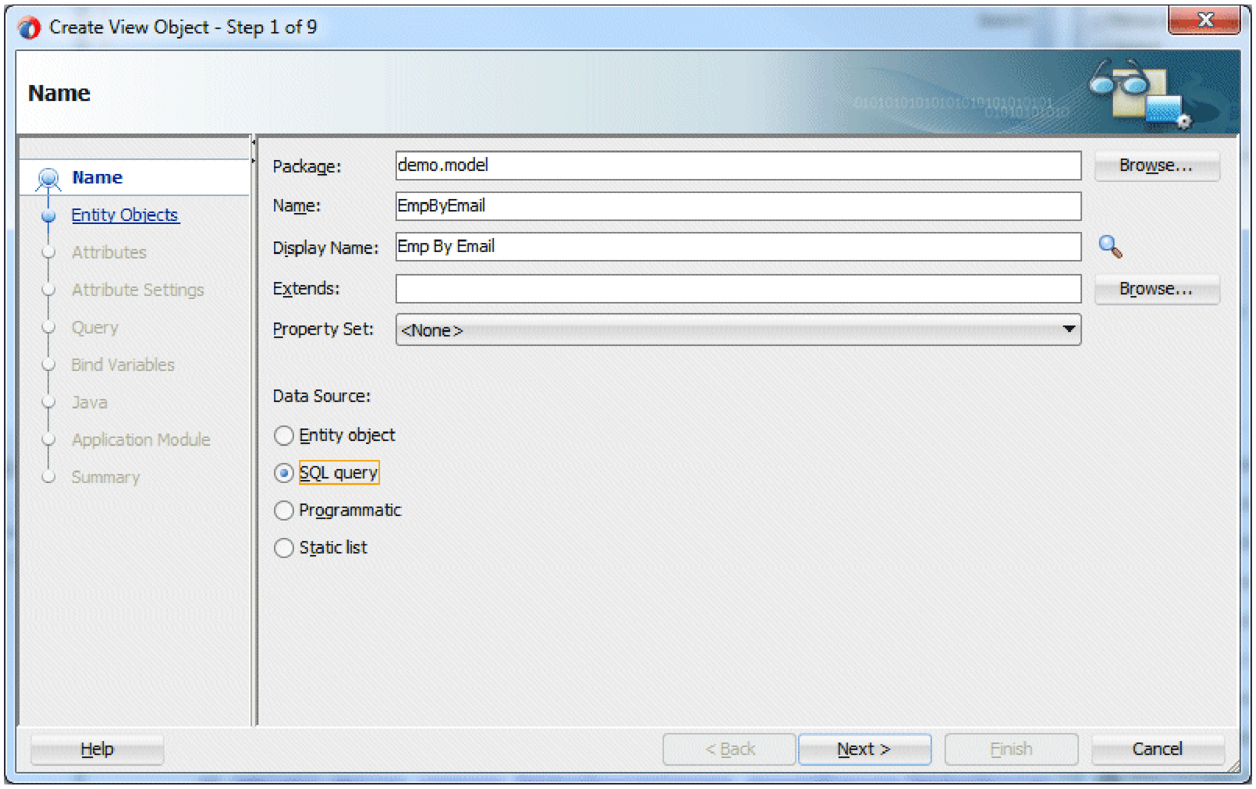
Step 6: Create a Query-only Business Service Based on Parameters

In this step you create a view object that allows end users to search for an employee's name based on their email address. The view object uses a bind variable to pass the employee's email into the query.

1. In the Applications window locate the **demo.model** package and right-click it to choose **New View Object...**.



2. In the Create View Object wizard set the **Name** property to **EmpByEmail** and choose the **SQL Query** radio button as the data source. Click **Next**.



Package: demo.model
Name: EmpByEmail
Display Name: Emp By Email
Extends:
Property Set: <None>

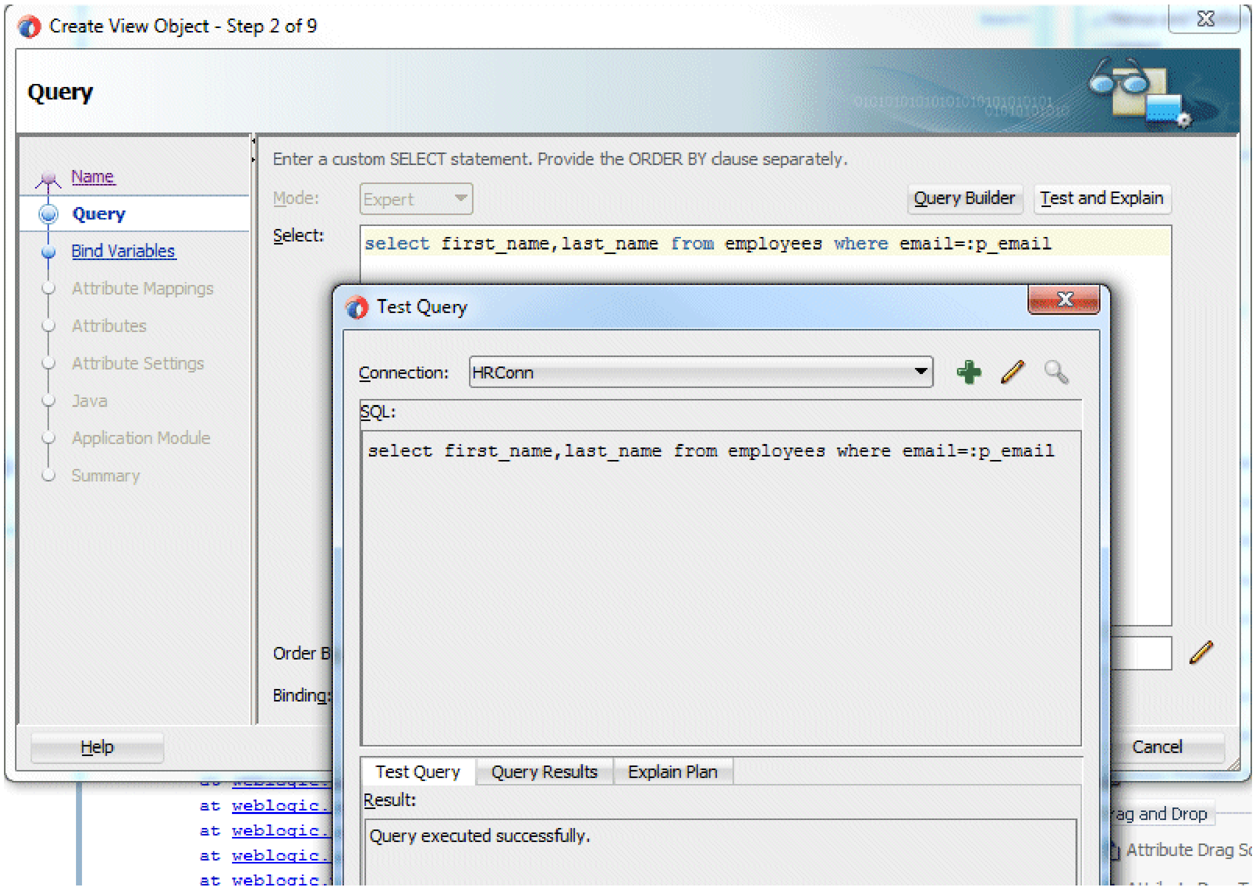
Data Source:
Entity object
[✓ SQL query]
Programmatic
Static list

3. In Step 2 of the Create View Object wizard type the following query:

```
select first_name, last_name from employees where email = :p_email
```

The **':'** before **p_email** means that it is a variable that will be passed to the query.

Click the **Test and Explain** button to verify your query.



Create View Object - Step 2 of 9

Query

Enter a custom SELECT statement. Provide the ORDER BY clause separately.

Mode: Expert

Select:

```
select first_name,last_name from employees where email=:p_email
```

Test Query

Connection: HRConn

SQL:

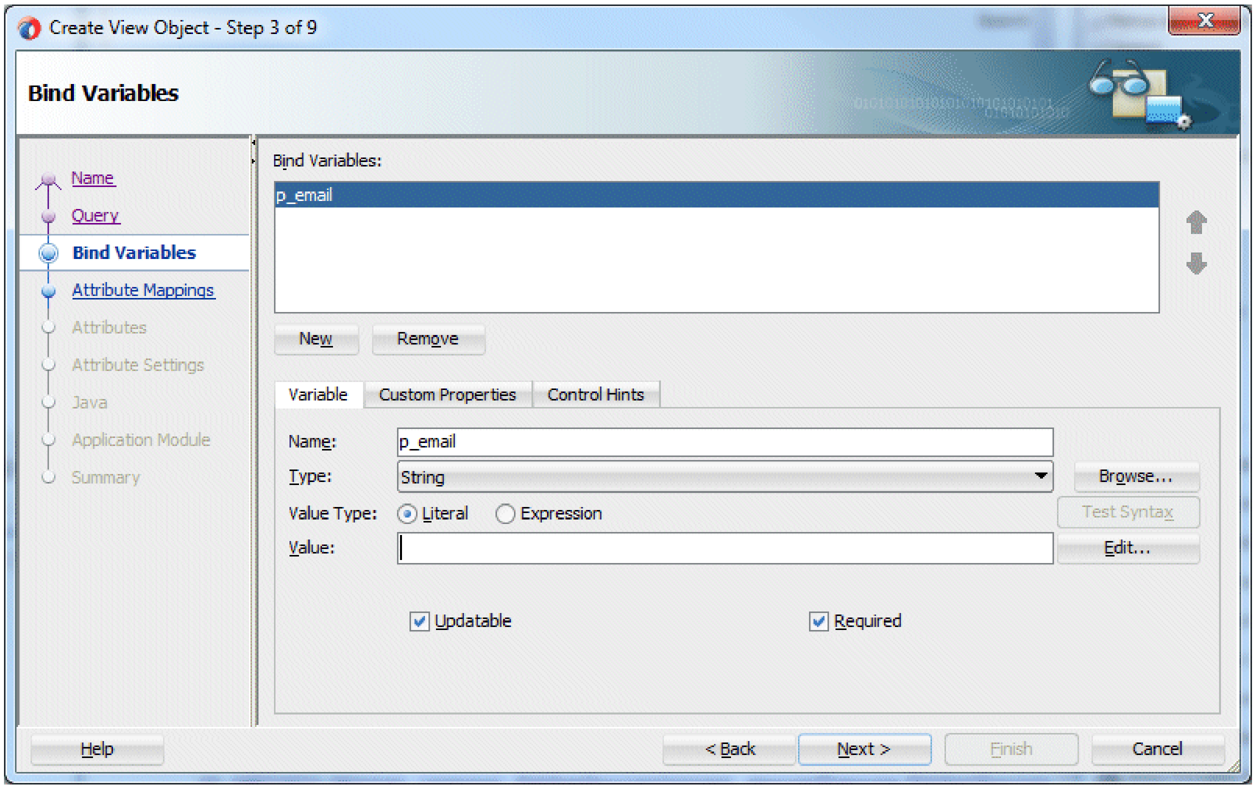
```
select first_name,last_name from employees where email=:p_email
```

Result:

Query executed successfully.

Click **Close** and then click **Next**.

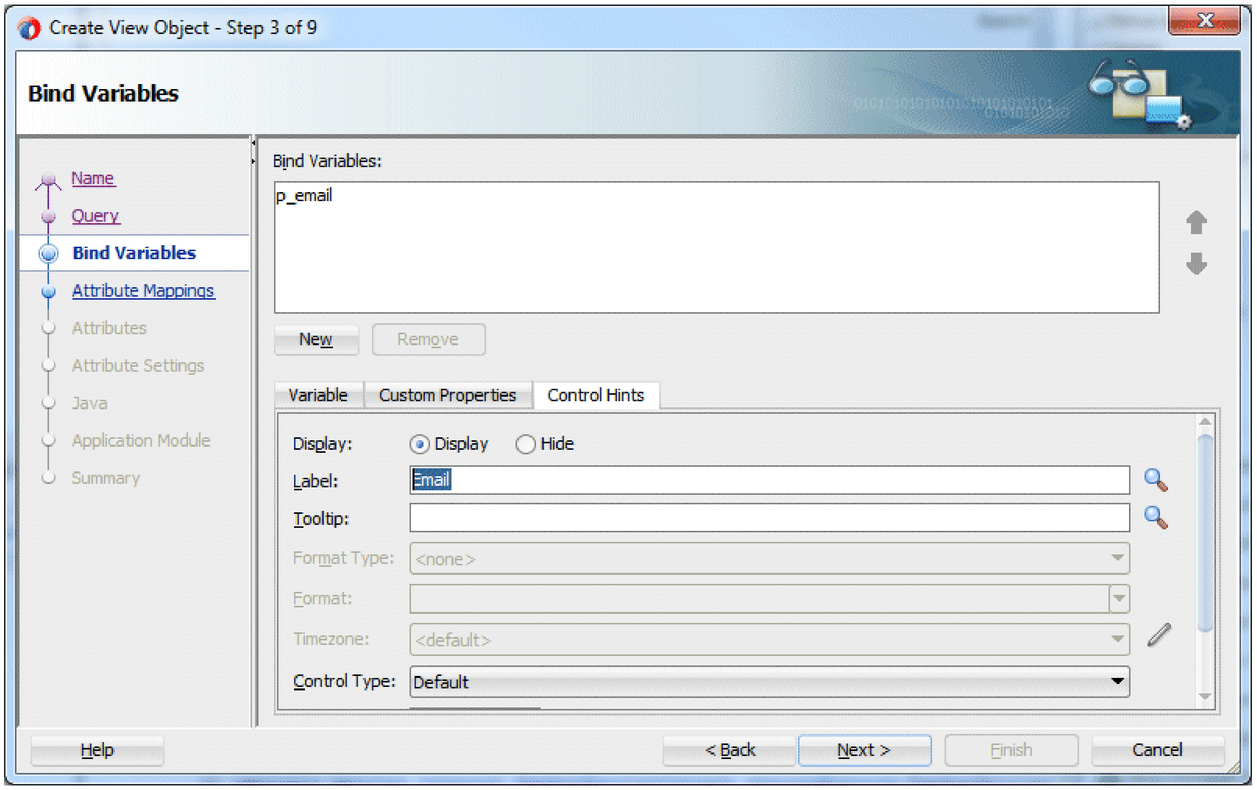
4. In Step 3 of the Create View Object wizard, click the **New** button to define a new bind variable. Set the **Name** property to **p_email**.



Bind Variables: p_email

Name: p_email
Type: String
Value Type: (●) Literal ( ) Expression
Value:
[✓] Updatable [✓] Required

Click the **Control Hints** tab and set the **Label Text** to **Email**.



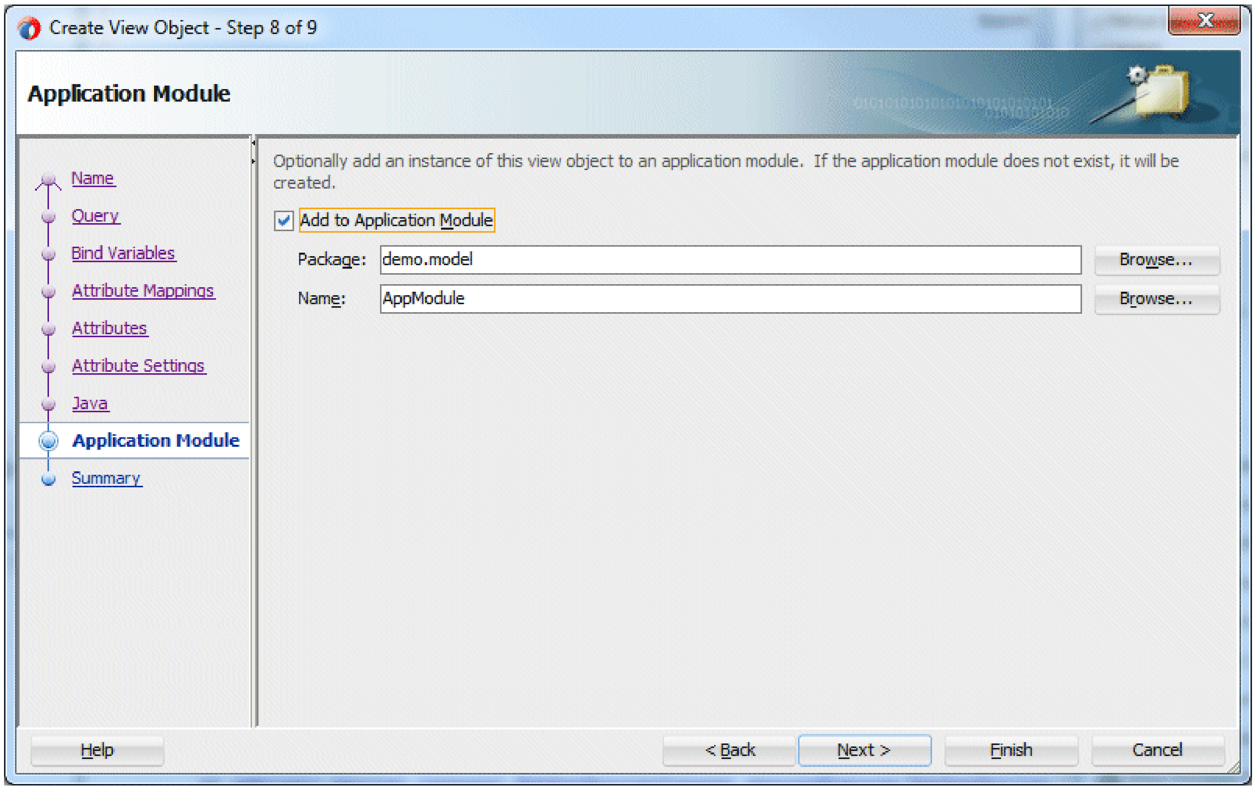
Bind Variables: p_email

Display: (●) Display ( ) Hide
Label: Email
Tooltip:
Format Type: <none>
Format:
Timezone: <default>
Control Type: Default

5. Click **Next** a few more times to accept all the defaults, until you get to step 8 of the wizard.

   Do **NOT** specify a Key Attribute, when prompted.

   Here check the **Add to Application Module** check box to include your new view in the data model.
   Click the **Finish** button.



Create View Object - Step 8 of 9

Application Module

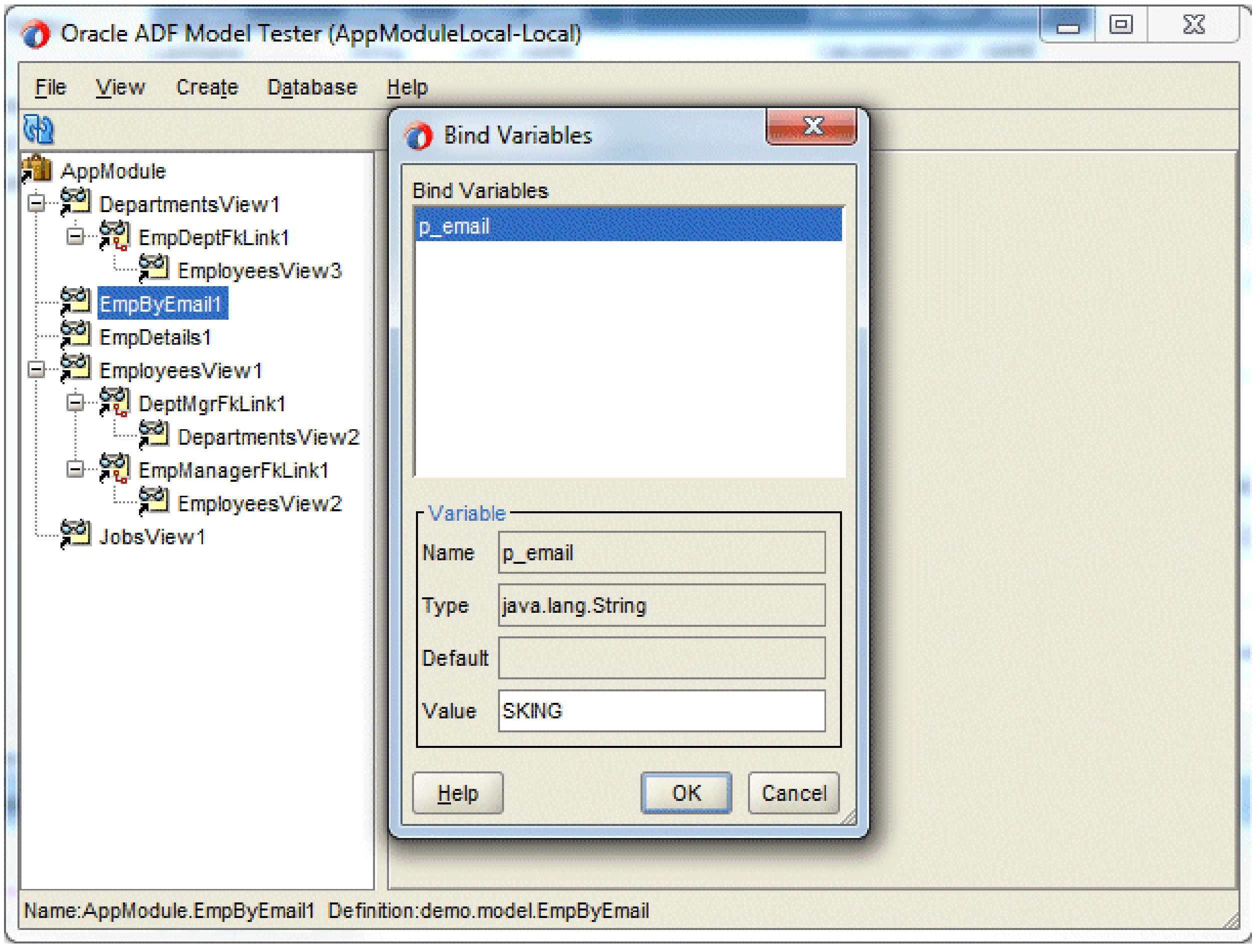
Optionally add an instance of this view object to an application module. If the application module does not exist, it will be created.

☑ Add to Application Module

Package: demo.model

Name: AppModule

6. **Save** your work and then **run** the Business Component Browser to test the new view. Double-click the new **EmpByEmail1** view and when prompted to insert a value for the variable enter **SKING** and press **OK** to get the results for this email address. (To run the Business Components Tester, right-click the application module and select Run)



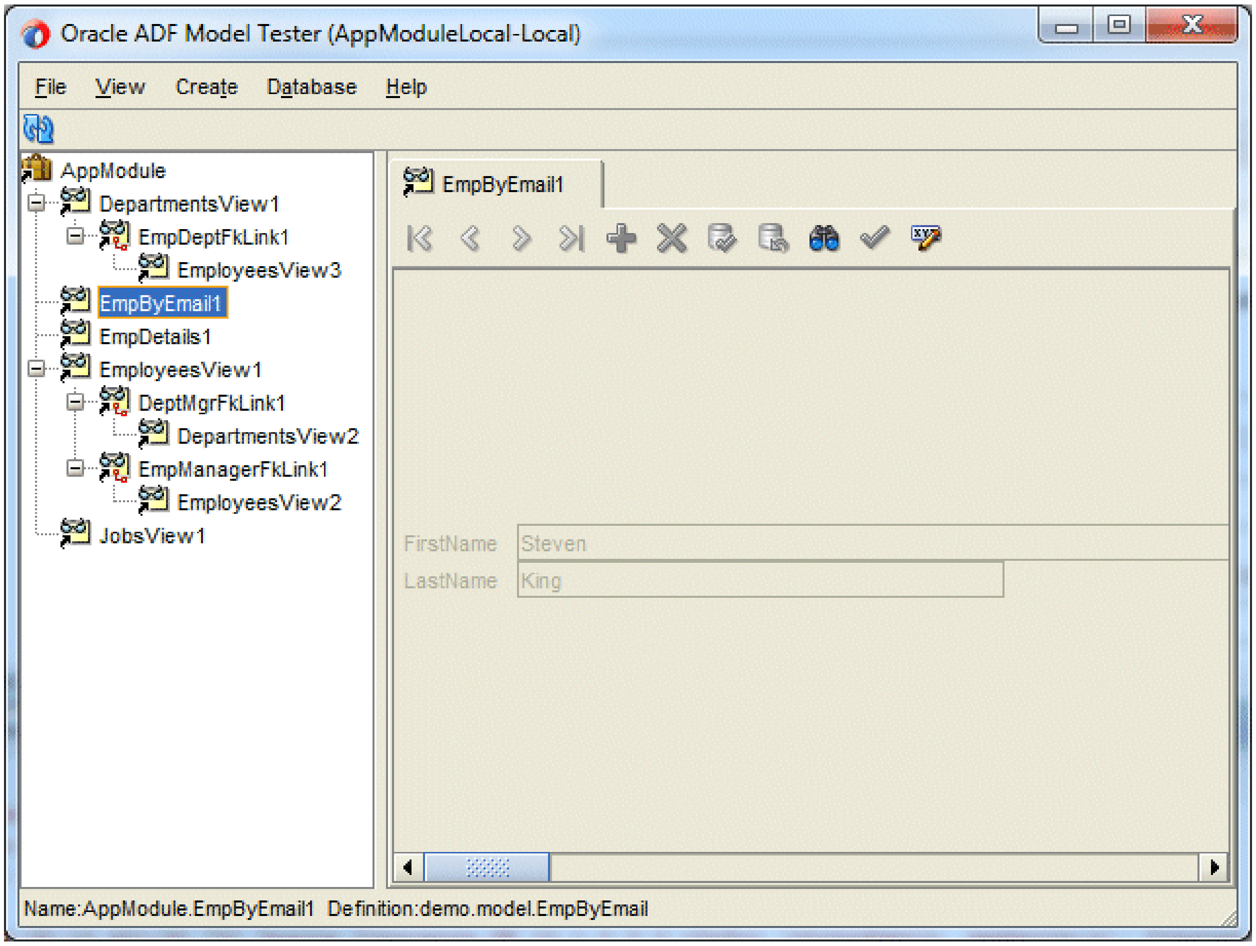
Bind Variables

p_email

Variable

Name	p_email
Type	java.lang.String
Default	
Value	SKING

Name:AppModule.EmpByEmail1 Definition:demo.model.EmpByEmail



FirstName Steven
LastName King

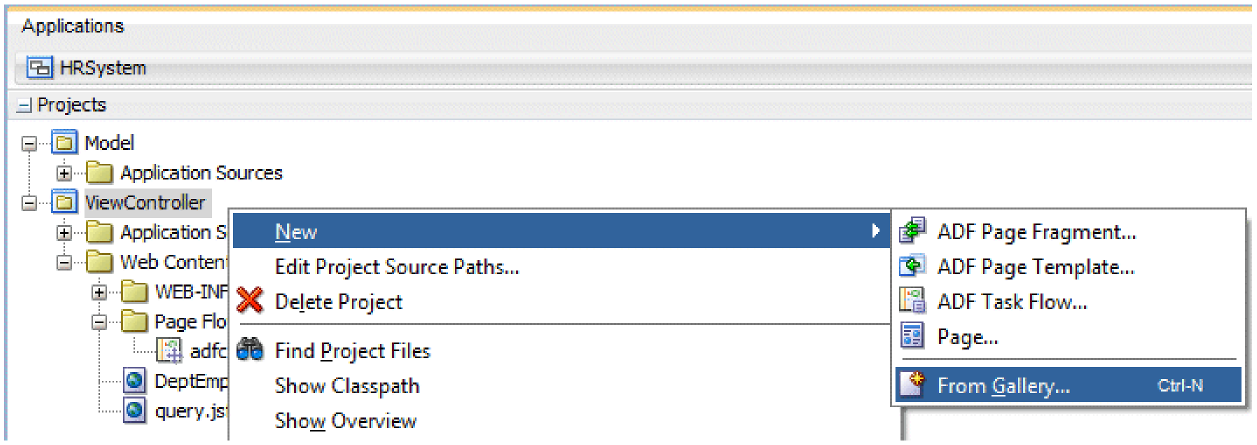
Name:AppModule.EmpByEmail1 Definition:demo.model.EmpByEmail

7. Notice that the Business Components Browser shows only King.
8. **Close** the Business Component Browser.

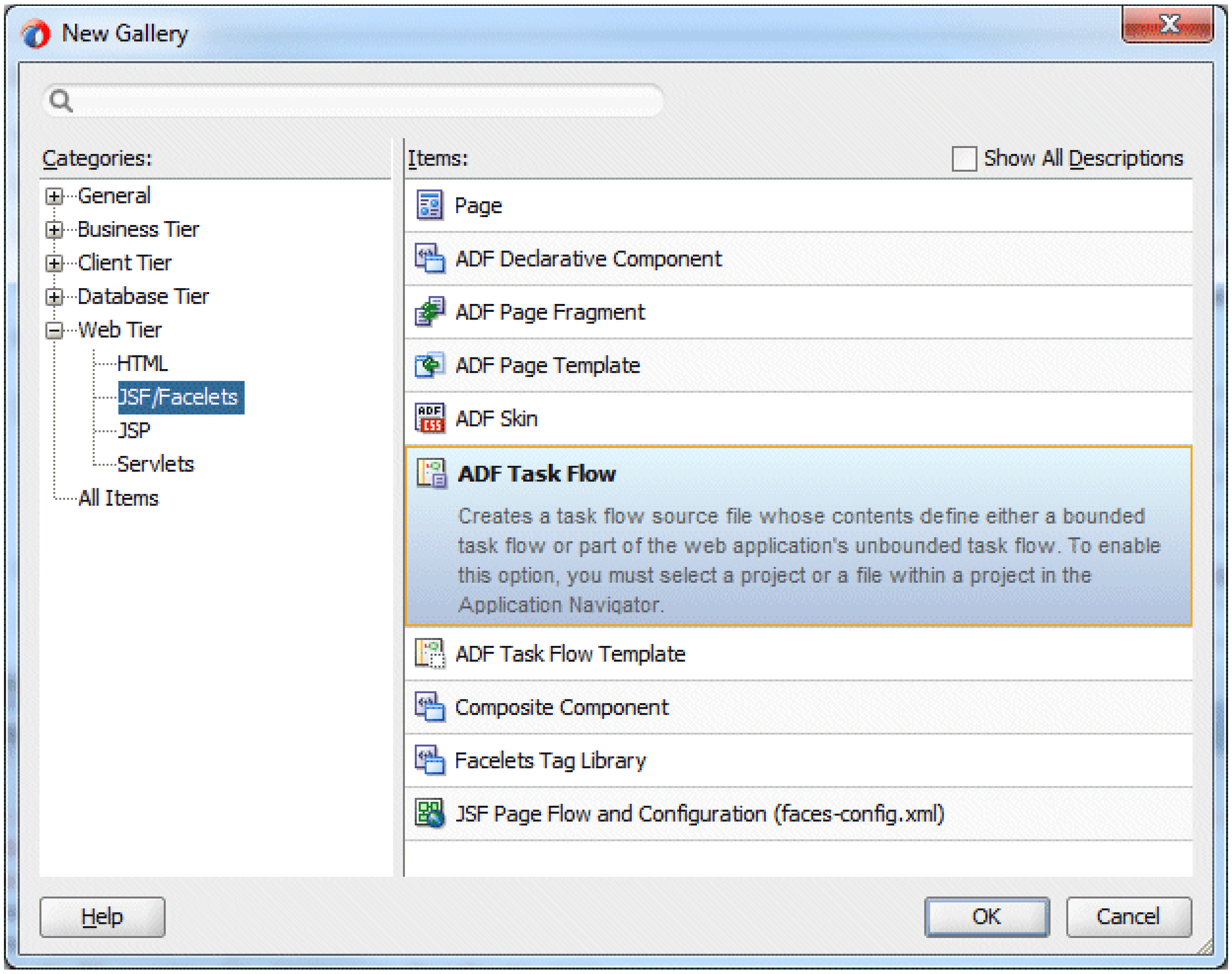
Step 7: Create a Reusable Page Fragment

In this step you create a reusable page fragment. You then embed the view object you created in the last step into the page fragment, and finally you use the page fragment in the DeptEmpPage page

1. First create a new task flow specifically for this page. In the Applications Navigator right-click the **ViewController** project and choose **New > From Gallery.**



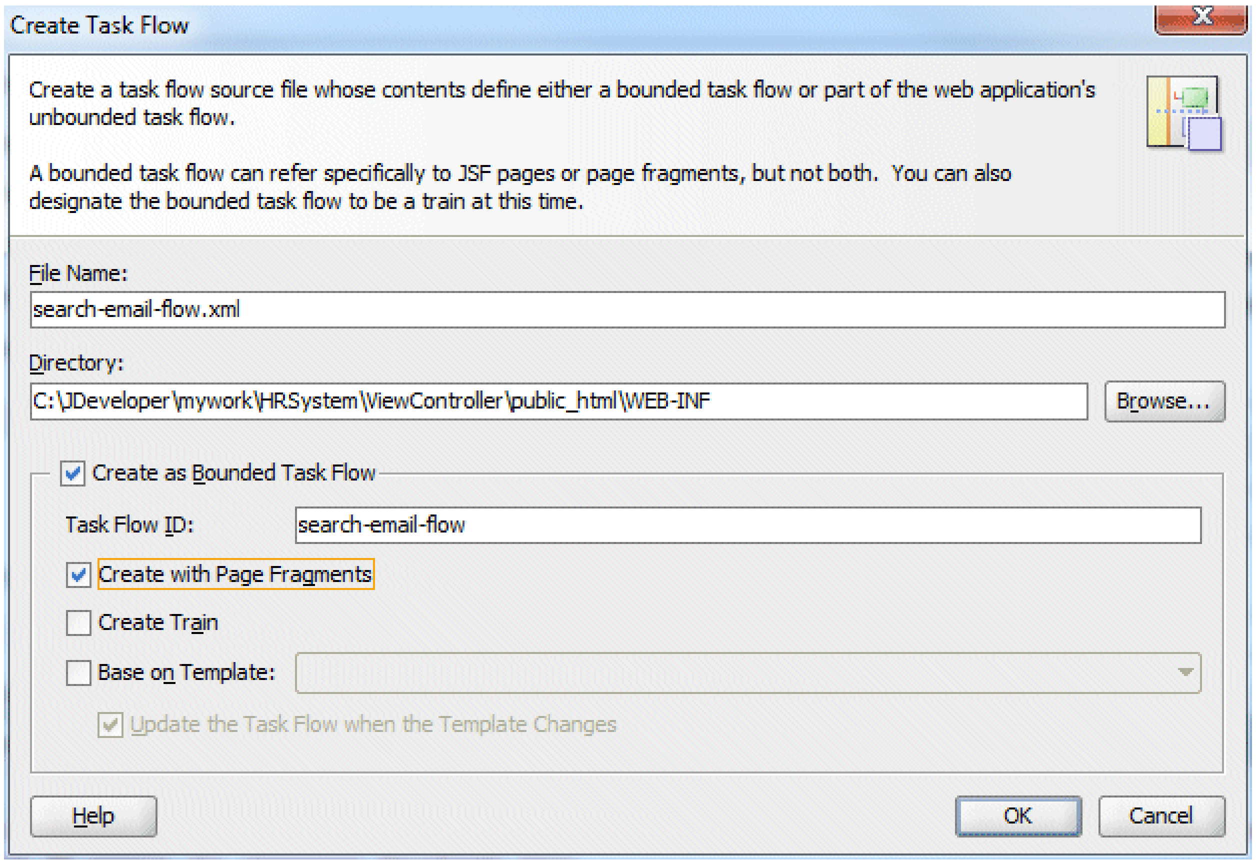
2. In the **Web Tier > JSF/Facelets** category choose **ADF Task Flow**.



Creates a task flow source file whose contents define either a bounded task flow or part of the web application's unbounded task flow. To enable this option, you must select a project or a file within a project in the Application Navigator.

Click **OK**.

3. In the Create Task Flow dialog set the **File Name** property to **search-email-flow.xml**.
Verify that the **Create As Bounded Task Flow** and **Create with Page Fragments** checkboxes are both checked.


Create Task Flow

Create a task flow source file whose contents define either a bounded task flow or part of the web application's unbounded task flow.

A bounded task flow can refer specifically to JSF pages or page fragments, but not both. You can also designate the bounded task flow to be a train at this time.

File Name: search-email-flow.xml

Directory: C:\JDeveloper\mywork\HRSystem\ViewController\public_html\WEB-INF

[✓] Create as Bounded Task Flow

Task Flow ID: search-email-flow

[✓] Create with Page Fragments

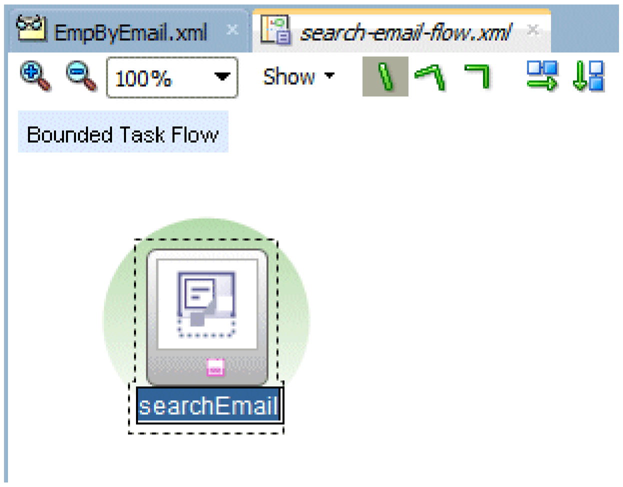
[ ] Create Train

[ ] Base on Template:

[✓] Update the Task Flow when the Template Changes

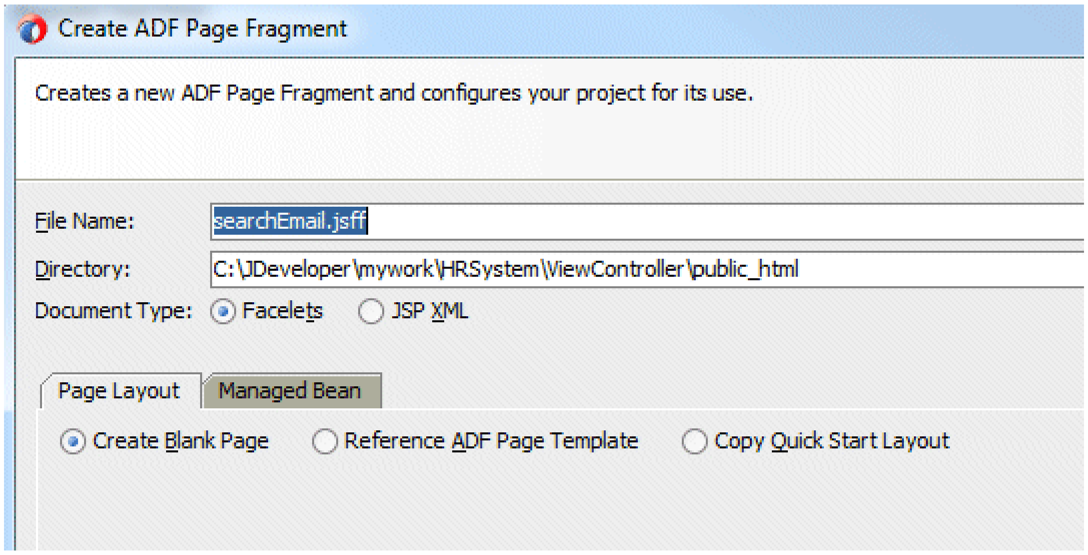
Click **OK**.

4. In the Design Editor drag a **View** component from the Components window onto the empty diagram and name it **searchEmail**. You only use a single page in this flow, but you can have bounded task flows with multiple pages and still include them in other JSF pages.



5. Double-click the new **searchEmail** view component to create the page fragment for it. Accept all the defaults in the dialog that displays and make sure the file name

is **searchEmail.jsff**. This creates the page as a page fragment that can be included in other JSF pages. Click **OK**.


Create ADF Page Fragment

Creates a new ADF Page Fragment and configures your project for its use.

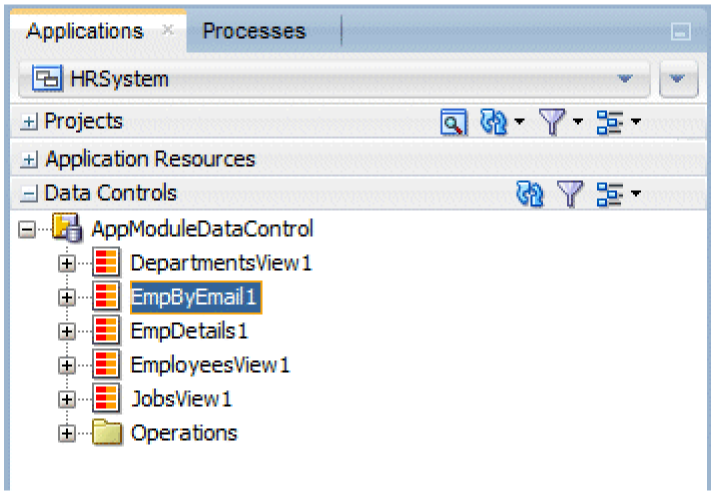
File Name: searchEmail.jsff

Directory: C:\JDeveloper\mywork\HRSystem\ViewController\public_html

Document Type: [✓ Facelets] JSP XML

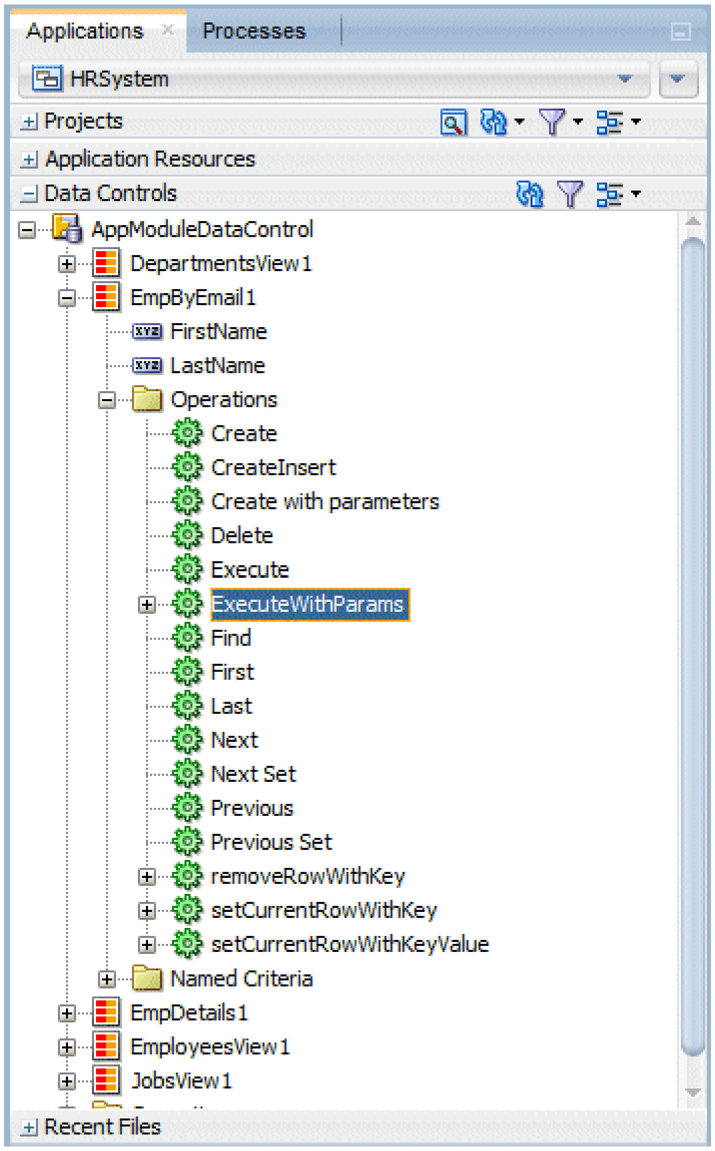
[✓ Create Blank Page] Reference ADF Page Template Copy Quick Start Layout

6. An empty page displays in the Design Editor. Expand the **Data Controls** accordion and, if necessary, click the **Refresh** button to ensure that the new **EmpByEmail1** data control appears in the list.



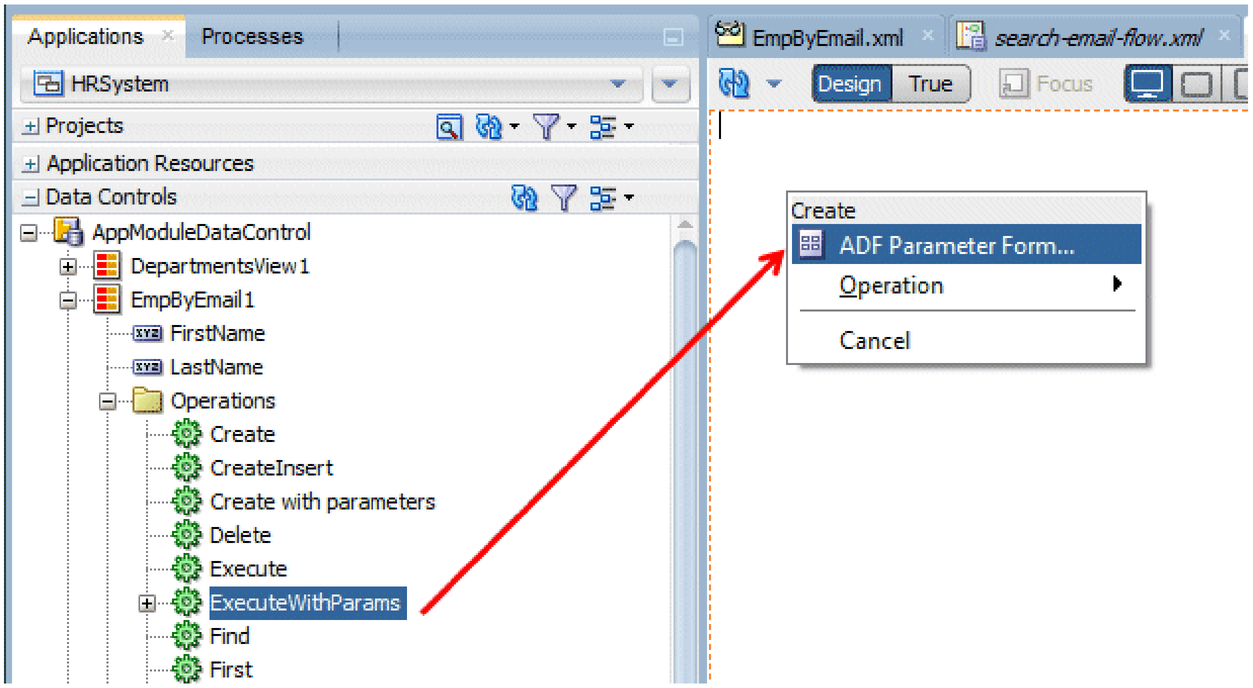
```
AppModuleDataControl
    DepartmentsView1
    EmpByEmail1
    EmpDetails1
    EmployeesView1
    JobsView1
    Operations
```

7. Expand the new **EmpByEmail1** view and the **Operations** node beneath it. Select the **ExecuteWithParams** operation. You are going to use this operation to execute the query for this view passing it the necessary parameter.

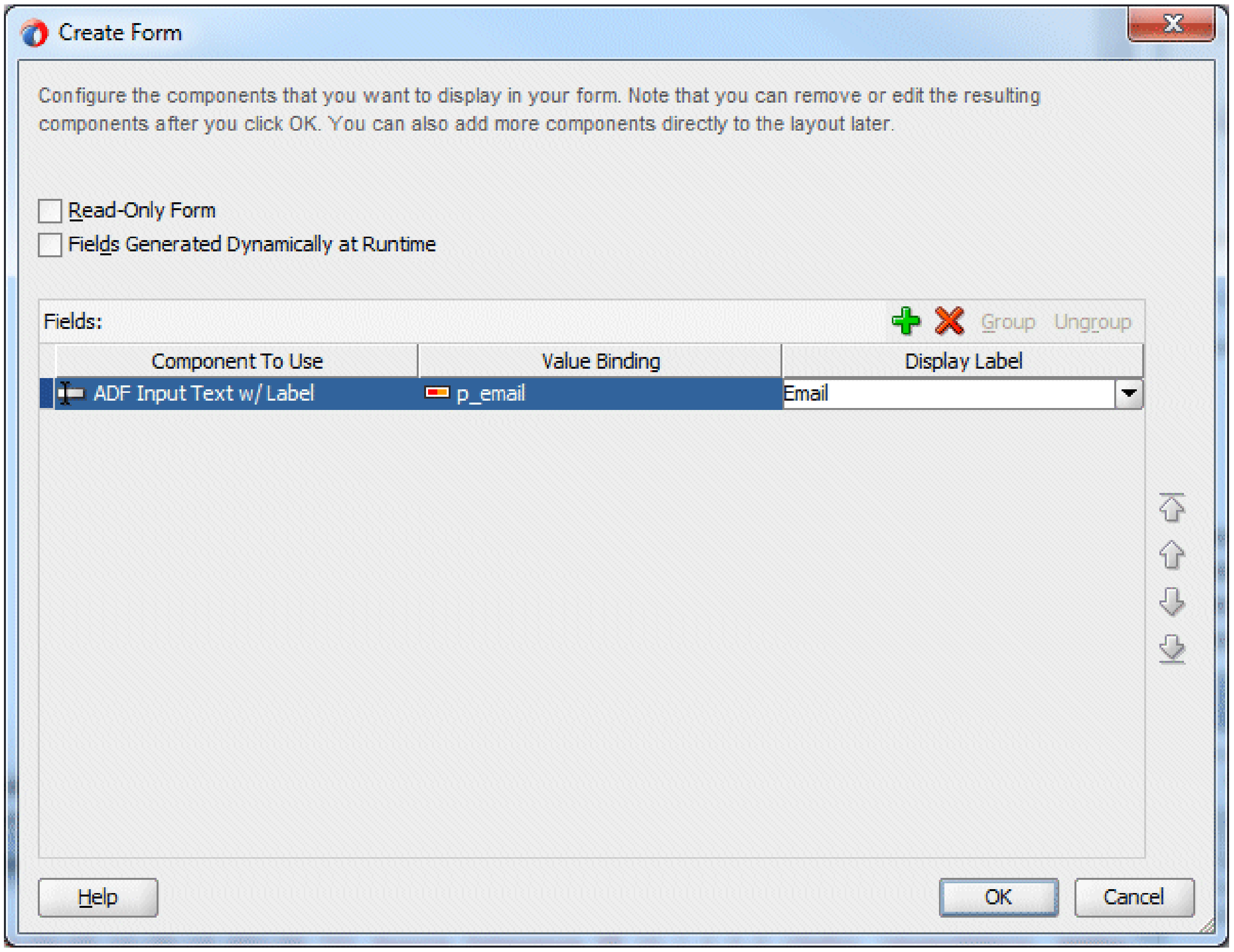


8.

Drag the **ExecuteWithParams** operation to your new page, and create it as an **ADF Parameter Form.**



9. In the Edit Form Fields dialog change the display label for the p_email value from default to **Email**. Click **OK**.


Create Form

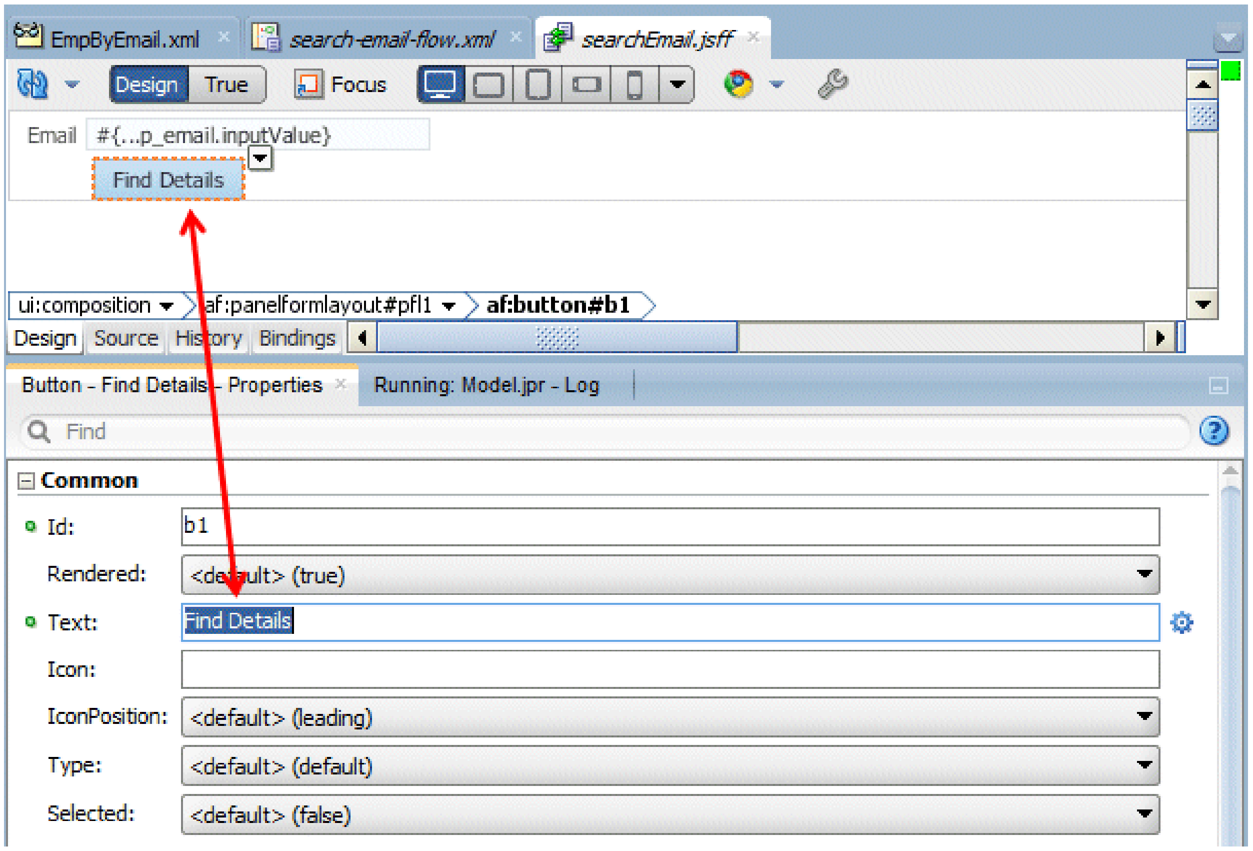
Configure the components that you want to display in your form. Note that you can remove or edit the resulting components after you click OK. You can also add more components directly to the layout later.

☐ Read-Only Form
☐ Fields Generated Dynamically at Runtime

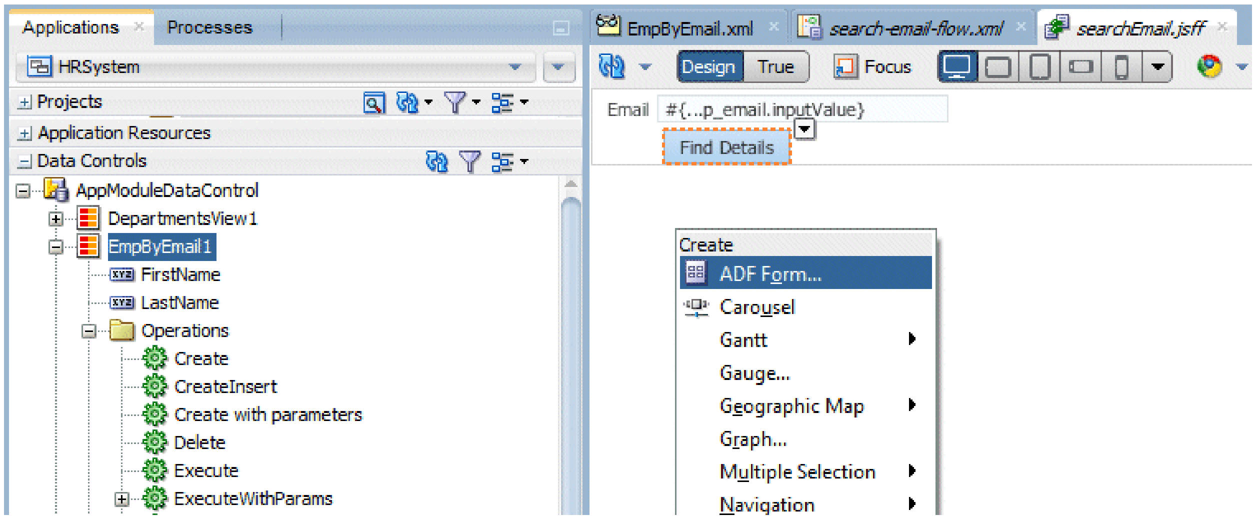
Fields:

Component To Use	Value Binding	Display Label
ADF Input Text w/ Label	p_email	Email

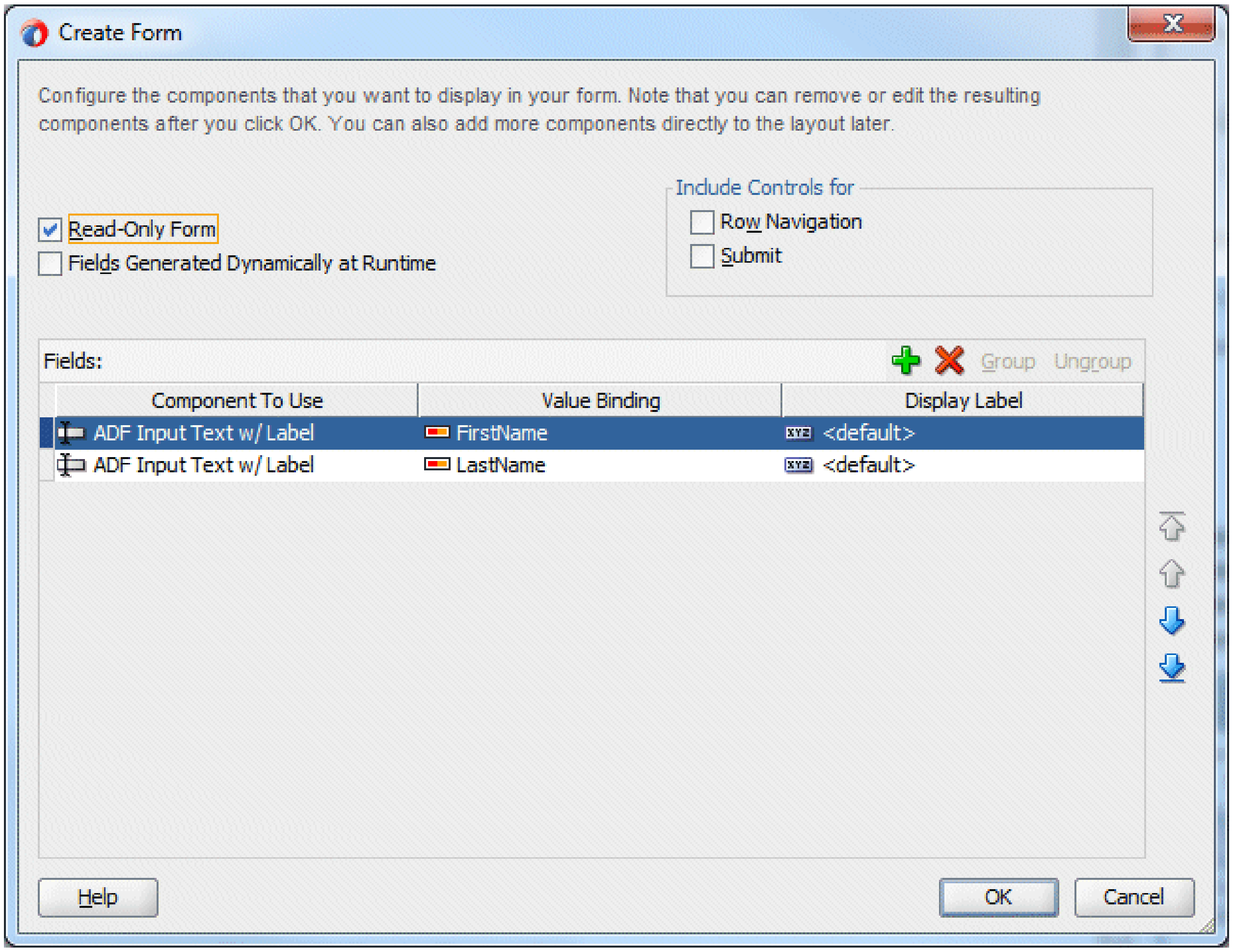
10. In the Design Editor for the page, click the **ExecuteWithParams** button and use the Properties window to change the **Text** property to **Find Details**.



11. From the Data Controls accordion, drag the **EmpByEmail1** collection onto the page beneath the button. Create it as a **Form > ADF Form...**.



12. Select **Read-Only Form** and accept the remaining defaults presented in the Edit Form Fields dialog. Then click **OK**.
Save your work.



Create Form

Configure the components that you want to display in your form. Note that you can remove or edit the resulting components after you click OK. You can also add more components directly to the layout later.

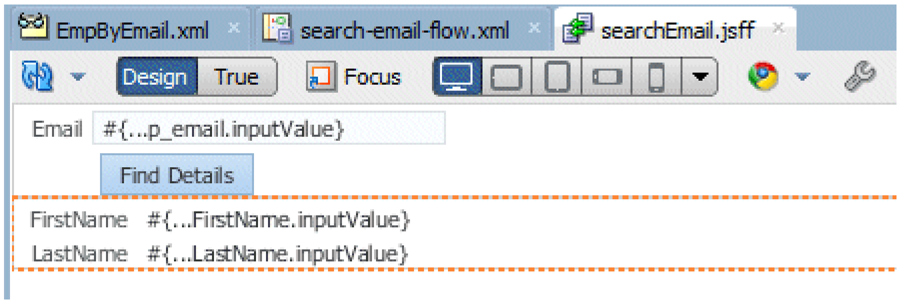
[✓] Read-Only Form
[ ] Fields Generated Dynamically at Runtime

Include Controls for
[ ] Row Navigation
[ ] Submit

Fields:

Component To Use	Value Binding	Display Label
ADF Input Text w/ Label	FirstName	<default>
ADF Input Text w/ Label	LastName	<default>

The page should look like this:

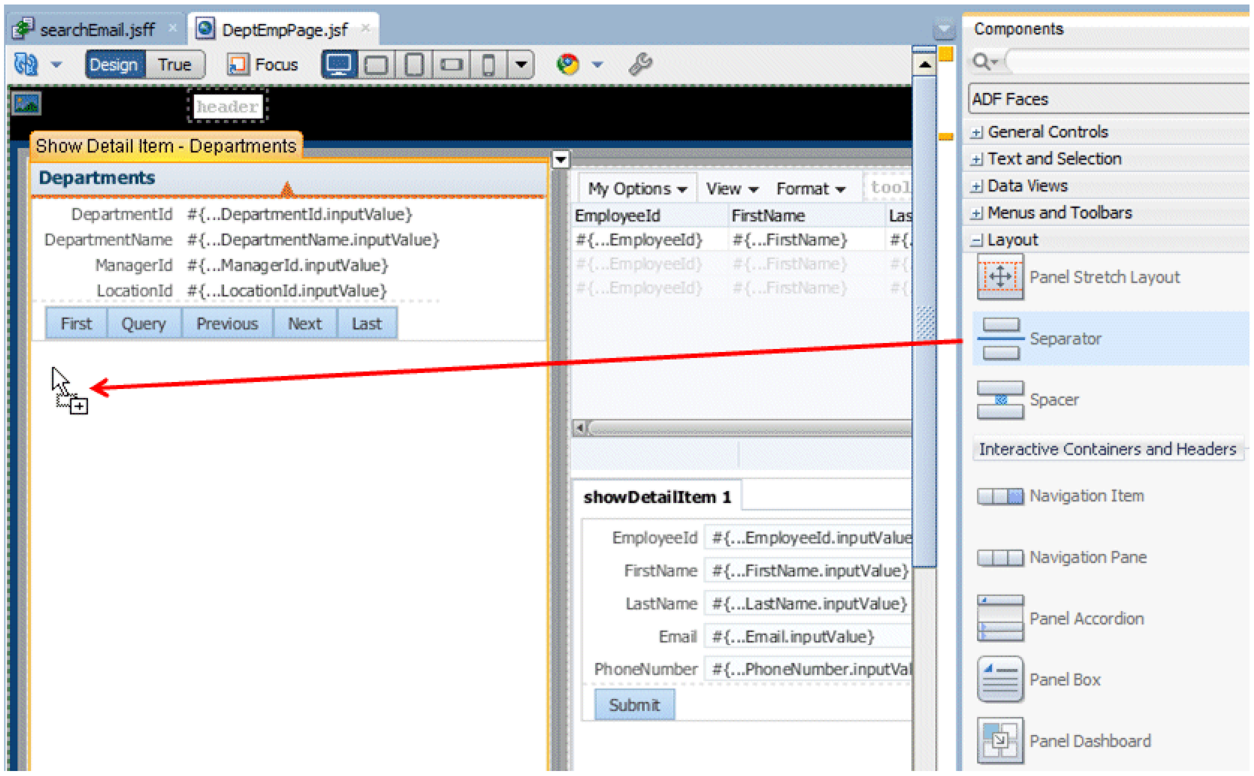


Email #{...p_email.inputValue}

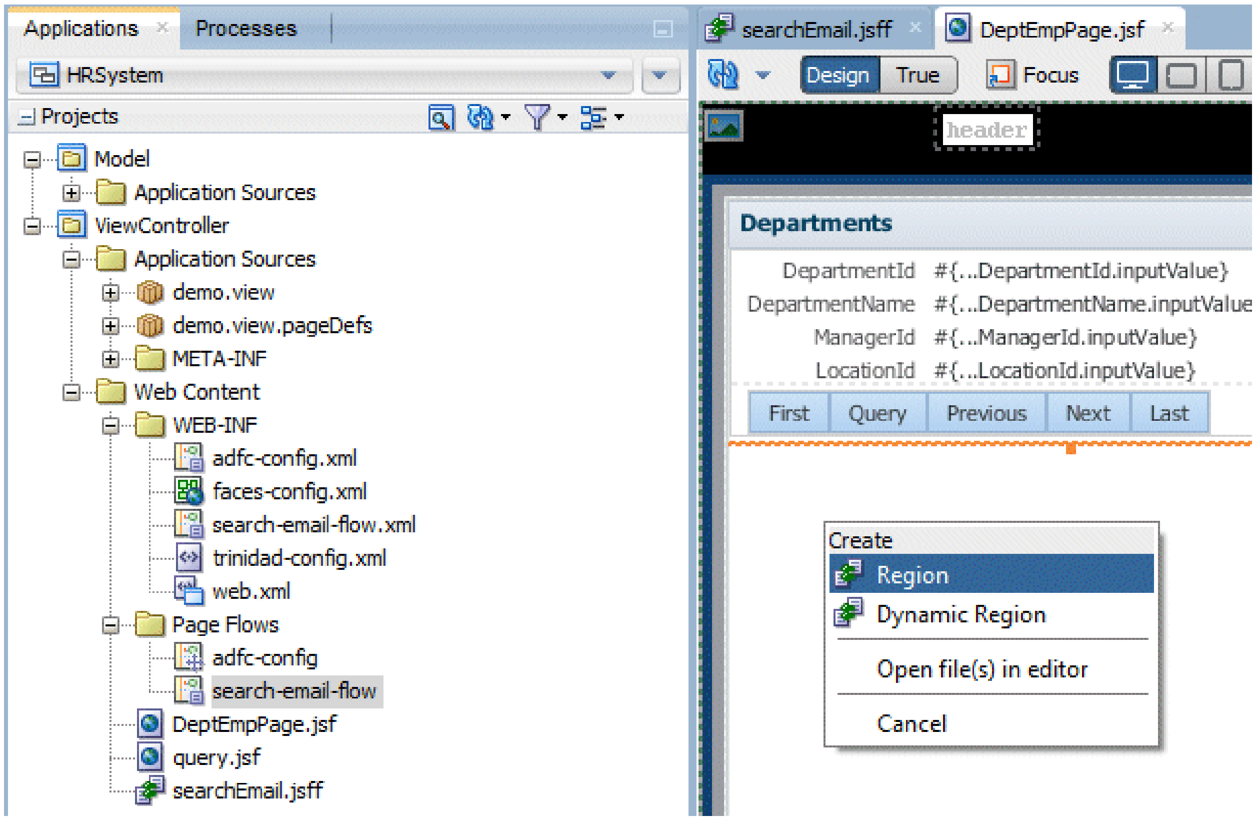
FirstName #{...FirstName.inputValue}

LastName #{...LastName.inputValue}

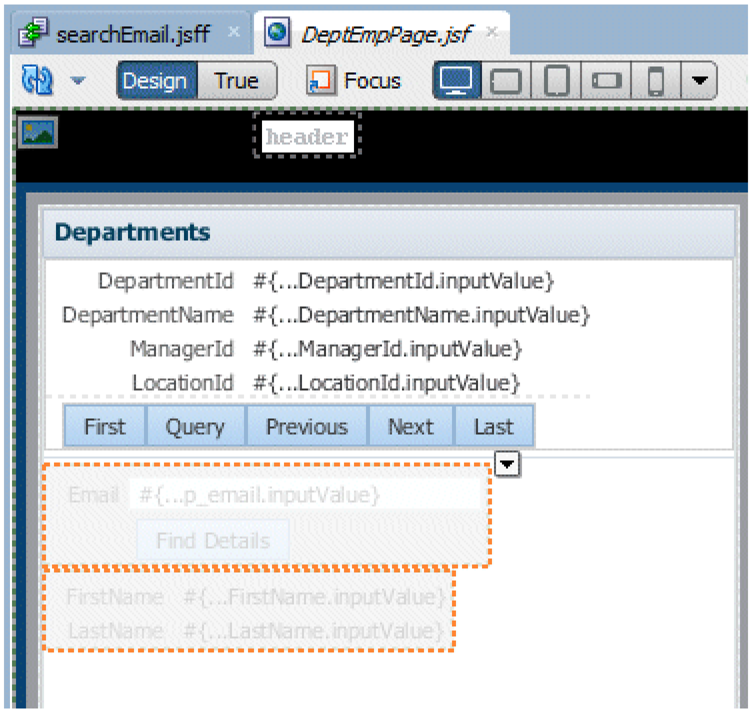
13. The new bounded task flow now contains a page fragment; next you include the complete bounded task flow inside another JSF page.
In the Applications window locate the **DeptEmpPage.jsf** file and open it in the Design Editor, if it is not already open. From the Components window Layout section drag and drop a **Separator** component into the left accordion in the **DeptEmpPage.jsf** page beneath the **Departments** form.


searchEmail.jsff
DeptEmpPage.jsf
Design
True
Focus
header
Show Detail Item - Departments
Departments
DepartmentId #{...DepartmentId.inputValue}
DepartmentName #{...DepartmentName.inputValue}
ManagerId #{...ManagerId.inputValue}
LocationId #{...LocationId.inputValue}
First
Query
Previous
Next
Last
My Options
View
Format
EmployeeId
FirstName
#{...EmployeeId}
#{...FirstName}
showDetailItem 1
EmployeeId #{...EmployeeId.inputValue
FirstName #{...FirstName.inputValue}
LastName #{...LastName.inputValue}
Email #{...Email.inputValue}
PhoneNumber #{...PhoneNumber.inputVal
Submit
Components
ADF Faces
General Controls
Text and Selection
Data Views
Menus and Toolbars
Layout
Panel Stretch Layout
Separator
Spacer
Interactive Containers and Headers
Navigation Item
Navigation Pane
Panel Accordion
Panel Box
Panel Dashboard

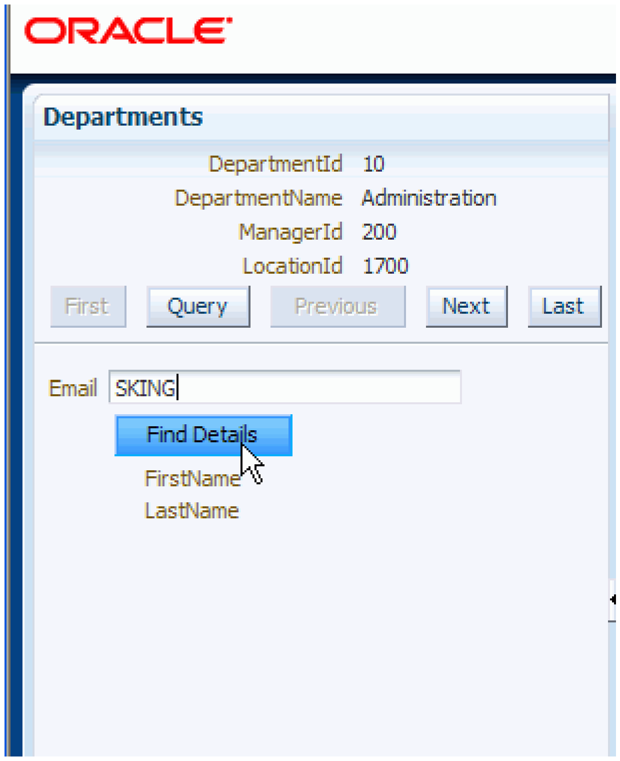
14. Add the new flow you created as a region to the existing page.
    From the Applications window drag and drop **search-email-flow.xml** into the left accordion in the **DeptEmpPage.jsf** page beneath the new **separator**. Create it as a **Region**.



15. Your page should now look like the screen shot below.

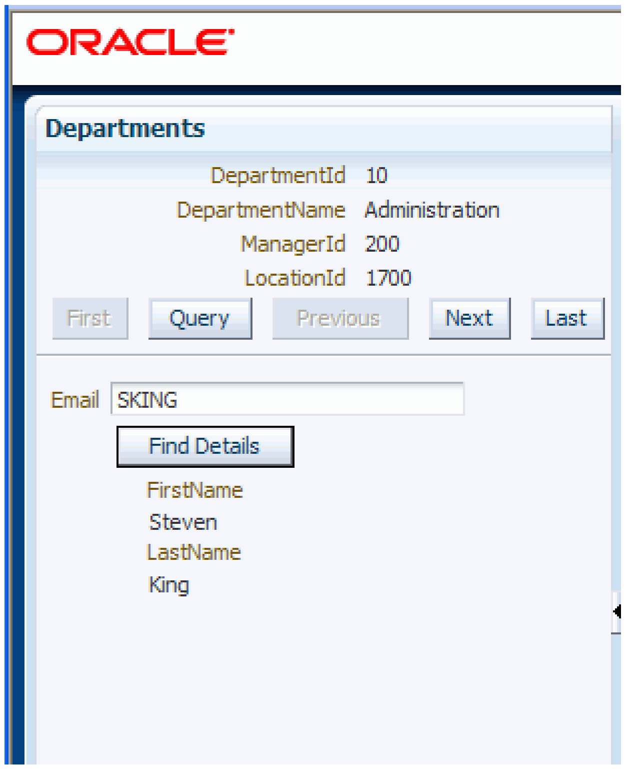

searchEmail.jsff
DeptEmpPage.jsf
Design
True
Focus
header
Departments
DepartmentId #{...DepartmentId.inputValue}
DepartmentName #{...DepartmentName.inputValue}
ManagerId #{...ManagerId.inputValue}
LocationId #{...LocationId.inputValue}
First
Query
Previous
Next
Last
Email #{...p_email.inputValue}
Find Details
FirstName #{...FirstName.inputValue}
LastName #{...LastName.inputValue}

16. Save your work and then **Run** the updated **DeptEmpPage.jsf** page.
    When the page displays in your browser test the new functionality by entering an email value (SKING) in the **Email** field and pressing the **Find Details** button.



DepartmentId 10
DepartmentName Administration
ManagerId 200
LocationId 1700
Email SKING

17. The employee name information is returned.



ORACLE

Departments

DepartmentId	10
DepartmentName	Administration
ManagerId	200
LocationId	1700

Email SKING

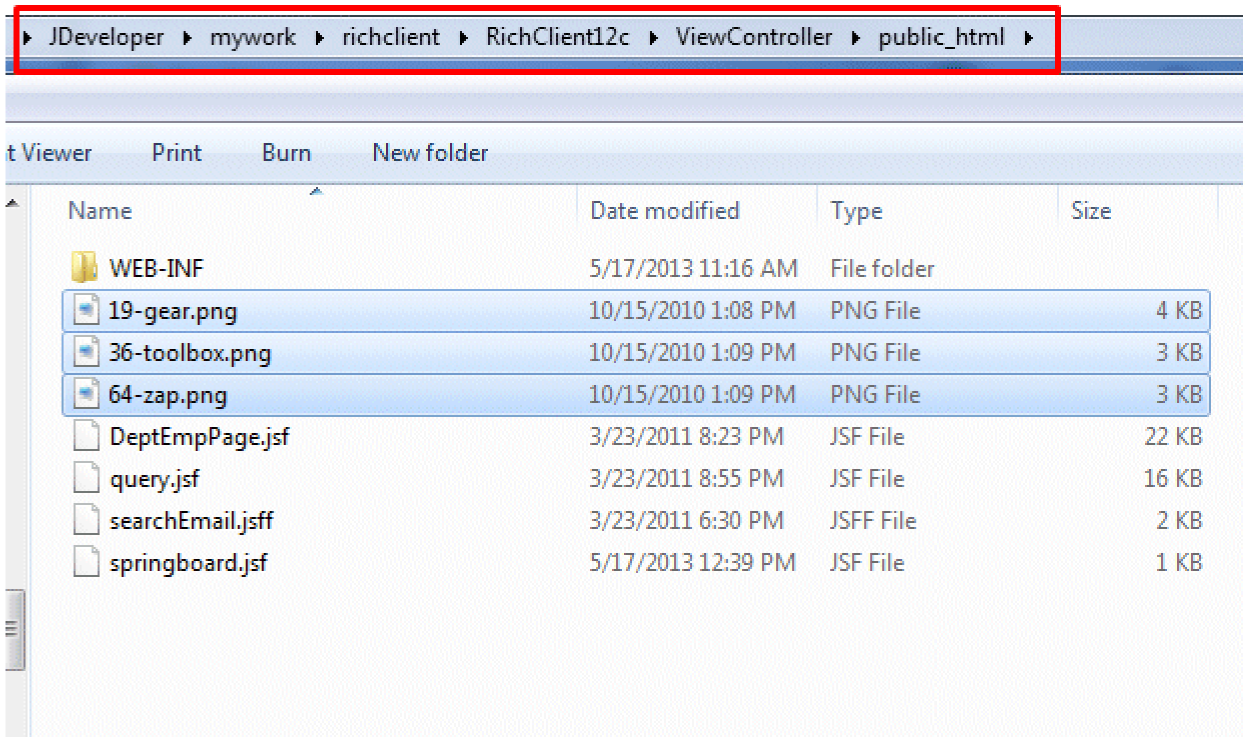
FirstName
Steven
LastName
King

18. Close the browser when you are done.

Step 8: Launching Pages with a Panel Springboard

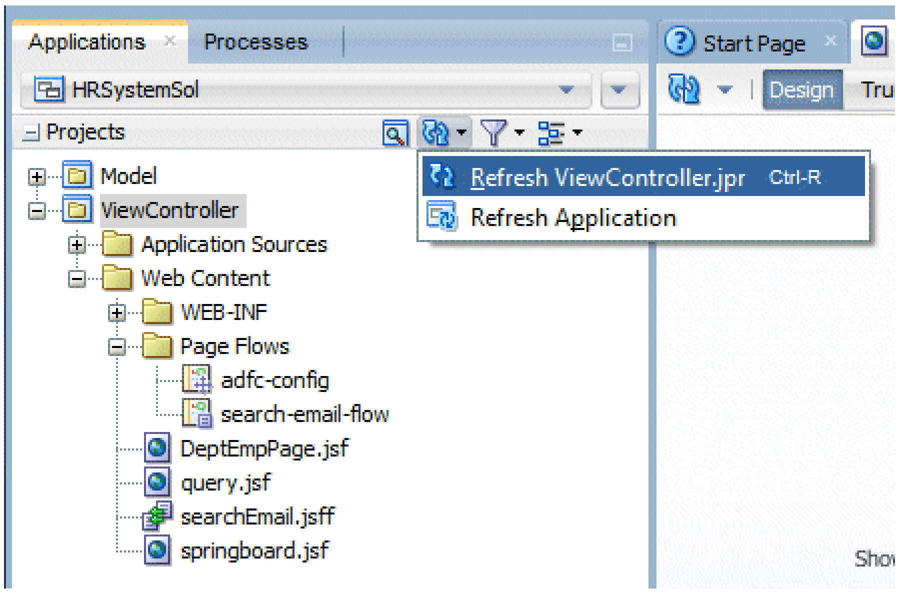
In this step you use a panel springboard as a launch pad fro your pages. Once you've created your pages and task flows, you can hook them together using a panelSpringboard component. All pages must be packaged in bounded task flows as page fragments and added as regions. Each page or task flow can have an icon associated with it and you can determine the behavior of the items, once one is selected.

1. In preparation for the next set of steps, include some icons into your application. Right click on each of the following 3 images and save the the icons to your application's **../public_html** directory.

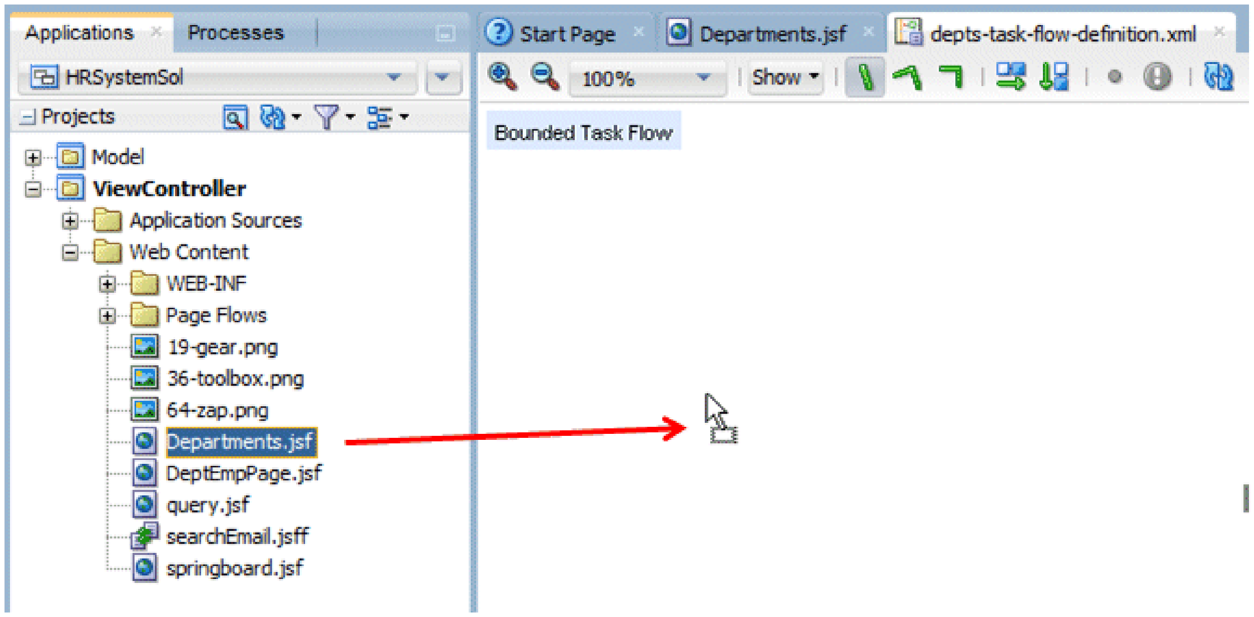

JDeveloper ▸ mywork ▸ richclient ▸ RichClient12c ▸ ViewController ▸ public_html ▸
t Viewer
Print
Burn
New folder

Name	Date modified	Type	Size
WEB-INF	5/17/2013 11:16 AM	File folder	
19-gear.png	10/15/2010 1:08 PM	PNG File	4 KB
36-toolbox.png	10/15/2010 1:09 PM	PNG File	3 KB
64-zap.png	10/15/2010 1:09 PM	PNG File	3 KB
DeptEmpPage.jsf	3/23/2011 8:23 PM	JSF File	22 KB
query.jsf	3/23/2011 8:55 PM	JSF File	16 KB
searchEmail.jsff	3/23/2011 6:30 PM	JSFF File	2 KB
springboard.jsf	5/17/2013 12:39 PM	JSF File	1 KB

2. Then back in JDeveloper, select the ViewController project and click the refresh icon. In the drop down menu, select **Refresh ViewController.jpr**.



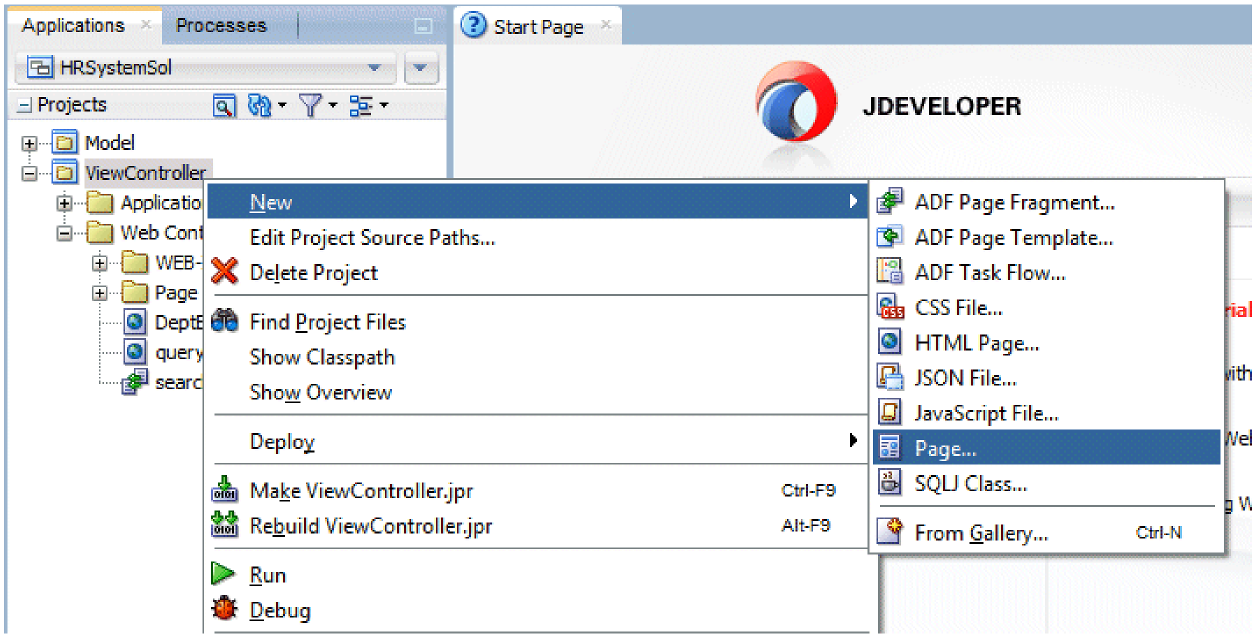
The Applications navigator will now display the included images.



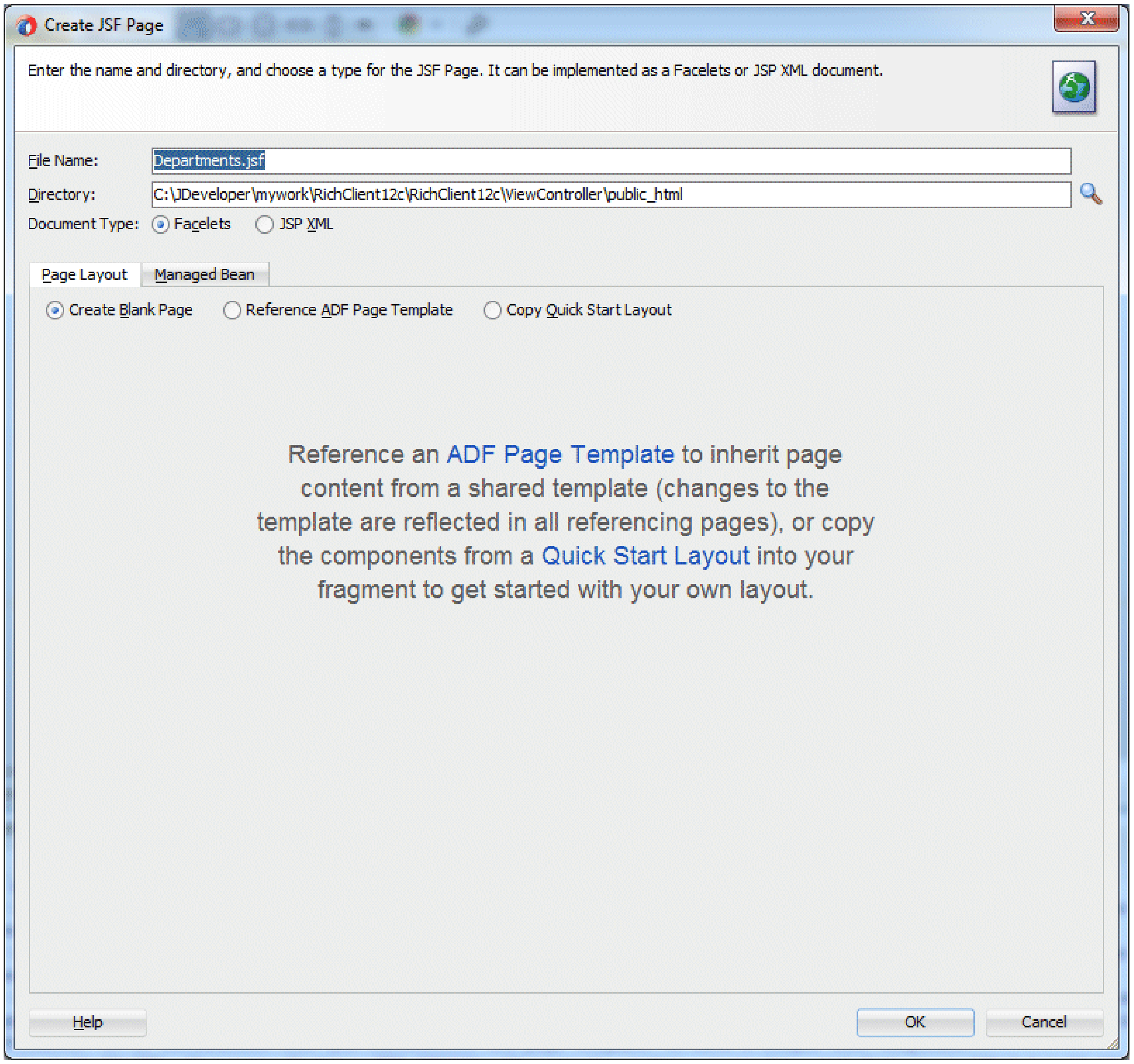
Now we are ready to create the components for the springboard.

The first thing we need to do is create a couple of pages and place them on bounded task flows as page fragments. There are a few different ways to do this, we'll show you one.

3. Right click the View Controller project and in the context menu select, **New > Page...**



4. Name it **Departments.jsf** and create it using a blank page.


Create JSF Page

Enter the name and directory, and choose a type for the JSF Page. It can be implemented as a Facelets or JSP XML document.

File Name: Departments.jsf

Directory: C:\JDeveloper\mywork\RichClient12c\RichClient12c\ViewController\public_html

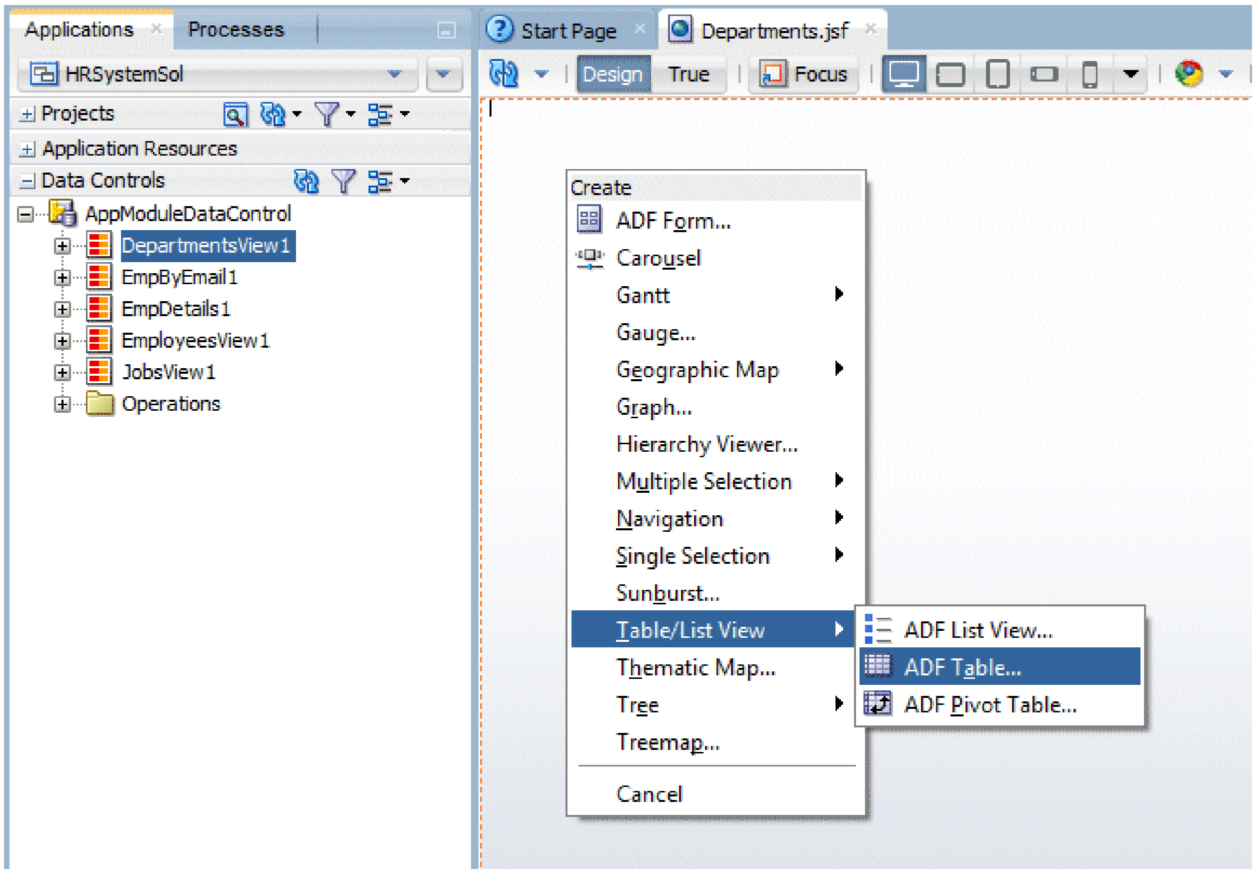
Document Type: [✓ Facelets] JSP XML

Page Layout | Managed Bean

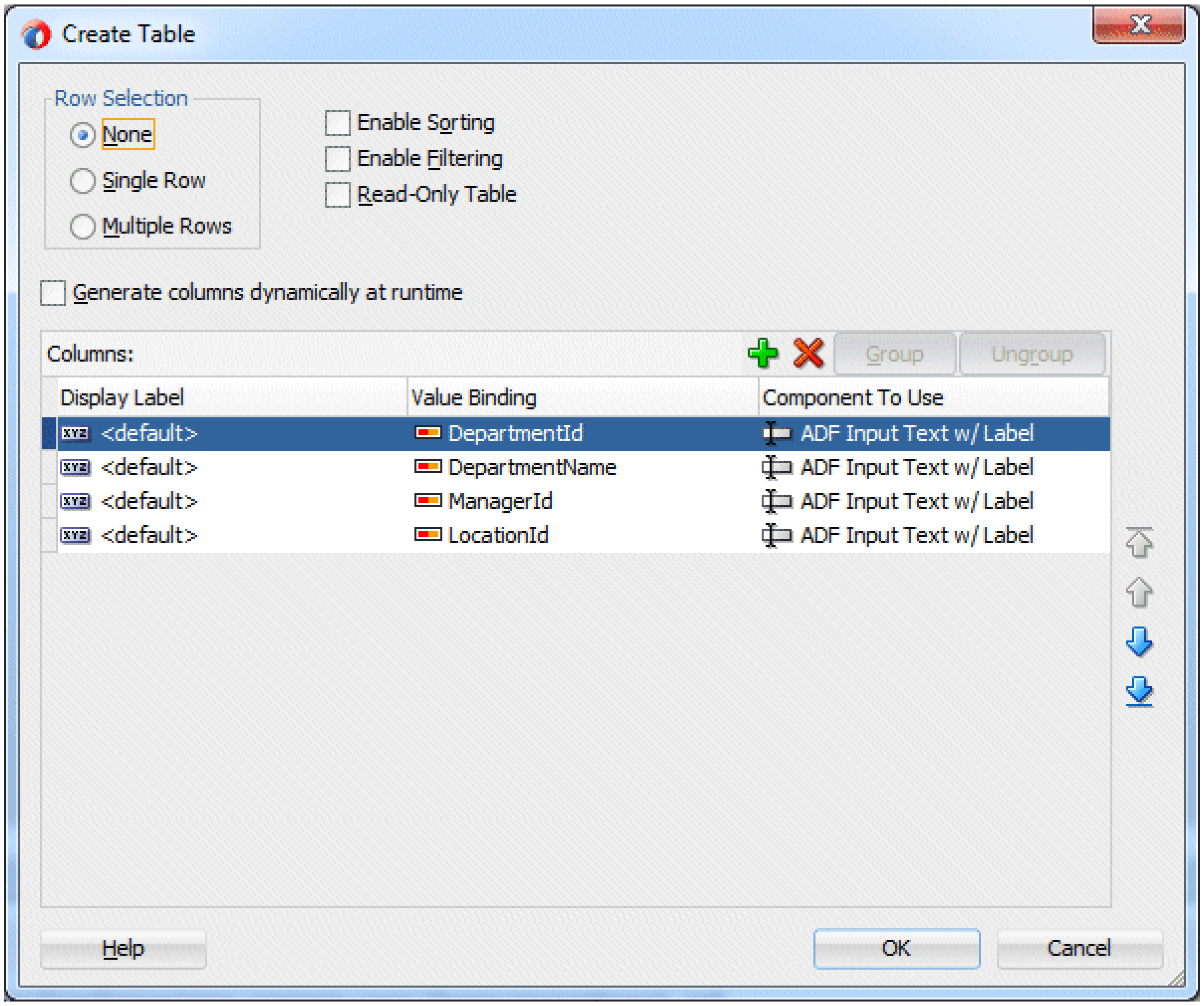
[✓ Create Blank Page] Reference ADF Page Template Copy Quick Start Layout

Reference an ADF Page Template to inherit page content from a shared template (changes to the template are reflected in all referencing pages), or copy the components from a Quick Start Layout into your fragment to get started with your own layout.

5. From the Data Controls palette, expand the AppModuleDataControl and drag and drop the **DepartmentsView1** onto the page.
   In the popup menu, select **Table/List View - ADF Table...**



6. In the Create Table dialog, click **OK** to accept all the fields.



Create Table

Row Selection
(● None) ○ Single Row ○ Multiple Rows

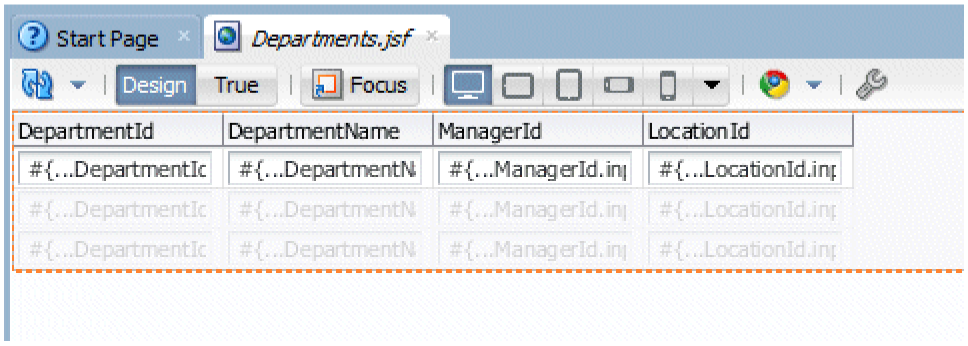
☐ Enable Sorting
☐ Enable Filtering
☐ Read-Only Table

☐ Generate columns dynamically at runtime

Columns:

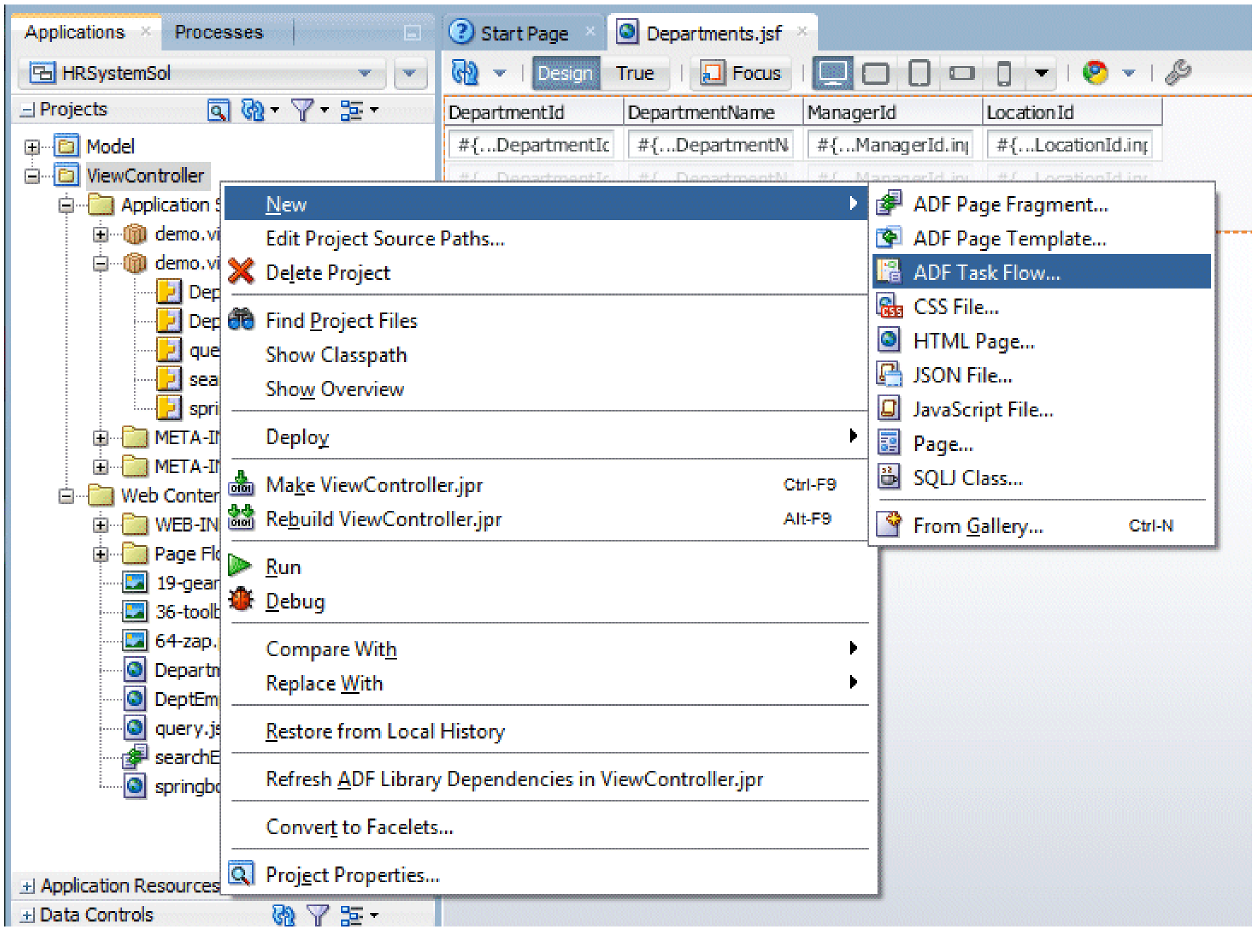
Display Label	Value Binding	Component To Use
<default>	DepartmentId	ADF Input Text w/ Label
<default>	DepartmentName	ADF Input Text w/ Label
<default>	ManagerId	ADF Input Text w/ Label
<default>	LocationId	ADF Input Text w/ Label

7. The resulting page should look like the image below. Save your work.

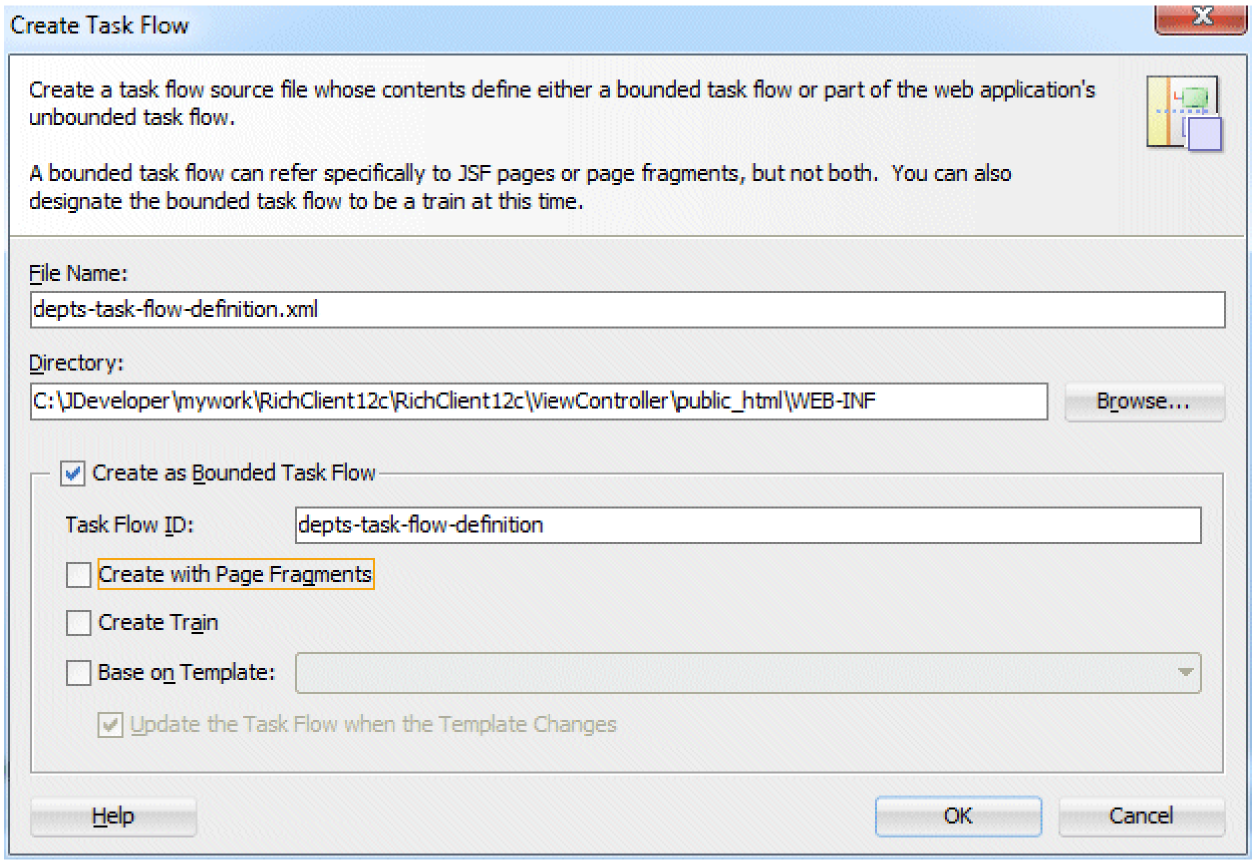


DepartmentId	DepartmentName	ManagerId	LocationId
#{...DepartmentIc	#{...DepartmentN	#{...ManagerId.in	#{...LocationId.inp
#{...DepartmentIc	#{...DepartmentN	#{...ManagerId.in	#{...LocationId.inp
#{...DepartmentIc	#{...DepartmentN	#{...ManagerId.in	#{...LocationId.inp

8. Select the ViewController project, and from the context menu select **New - ADF Task Flow...**



9. Name it **depts-task-flow-definition.xml** and deselect the **Create with Page Fragments** checkbox. Click **OK** to create the task flow.


Create Task Flow

Create a task flow source file whose contents define either a bounded task flow or part of the web application's unbounded task flow.

A bounded task flow can refer specifically to JSF pages or page fragments, but not both. You can also designate the bounded task flow to be a train at this time.

File Name: depts-task-flow-definition.xml

Directory: C:\JDeveloper\mywork\RichClient12c\RichClient12c\ViewController\public_html\WEB-INF

[✓] Create as Bounded Task Flow

Task Flow ID: depts-task-flow-definition

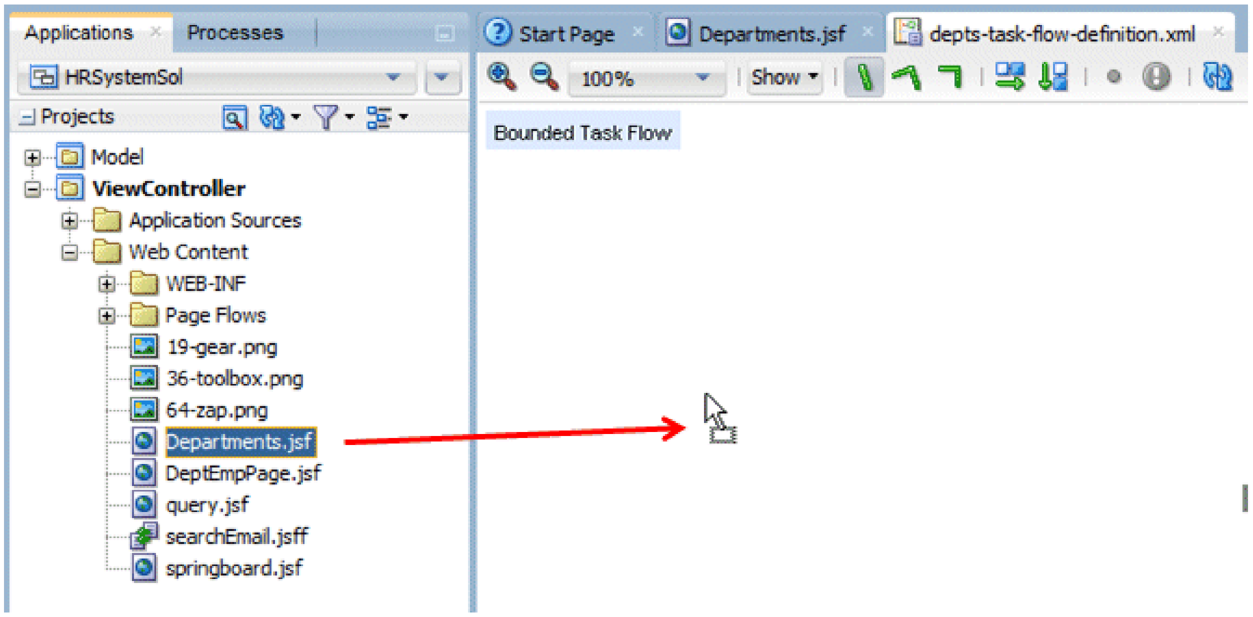
[ ] Create with Page Fragments

[ ] Create Train

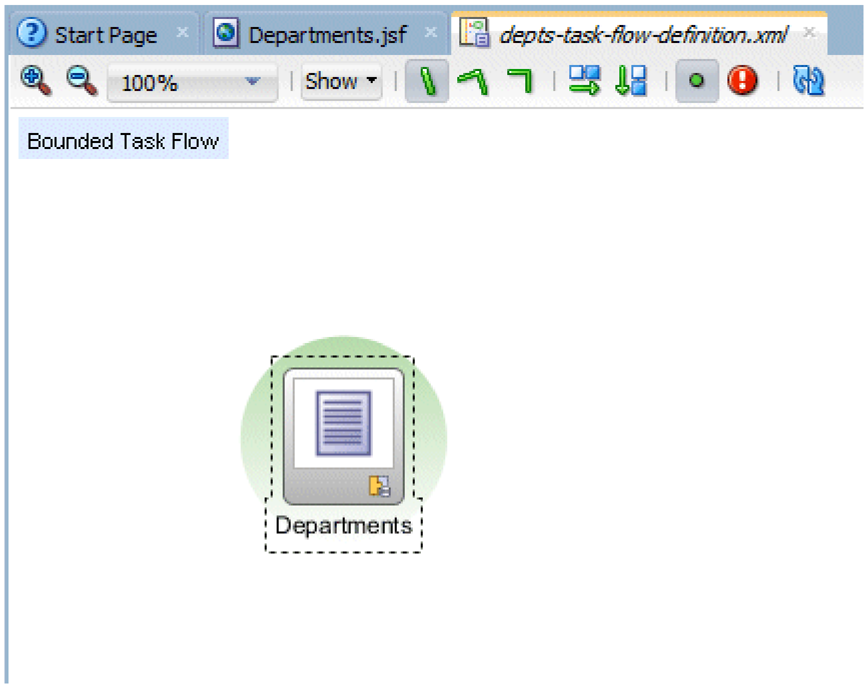
[ ] Base on Template:

[✓] Update the Task Flow when the Template Changes

10. From the Applications navigator, drag the **Departments.jsf** page onto the task flow and drop it.

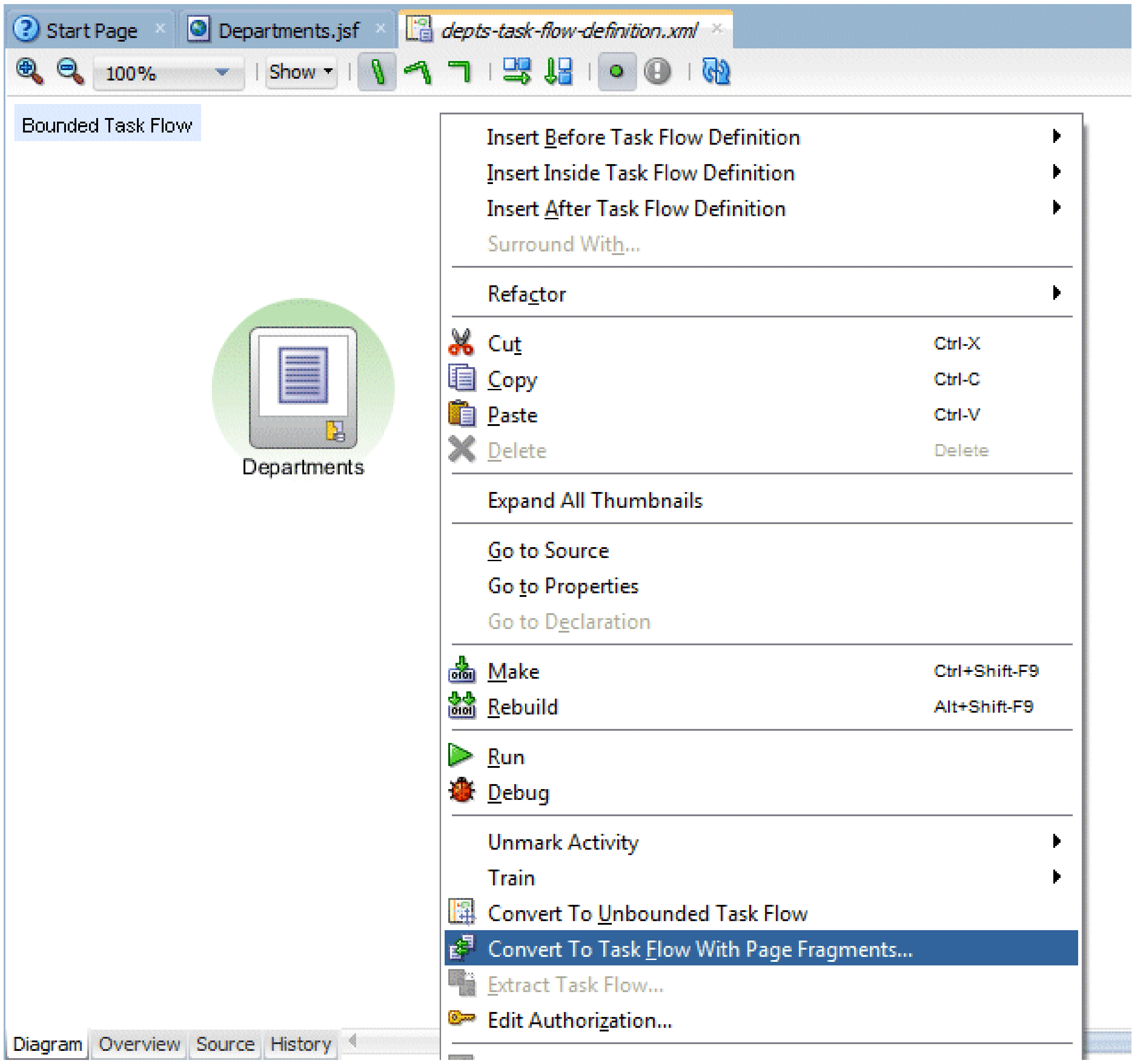


The resulting page is now part of the bounded task flow. .



11. Next, right click the task flow and select **Convert To Task Flow With Page Fragments...**

    Later on we'll create a page containing the panel springboard component. We can then add this task flow as a region on the springboard.

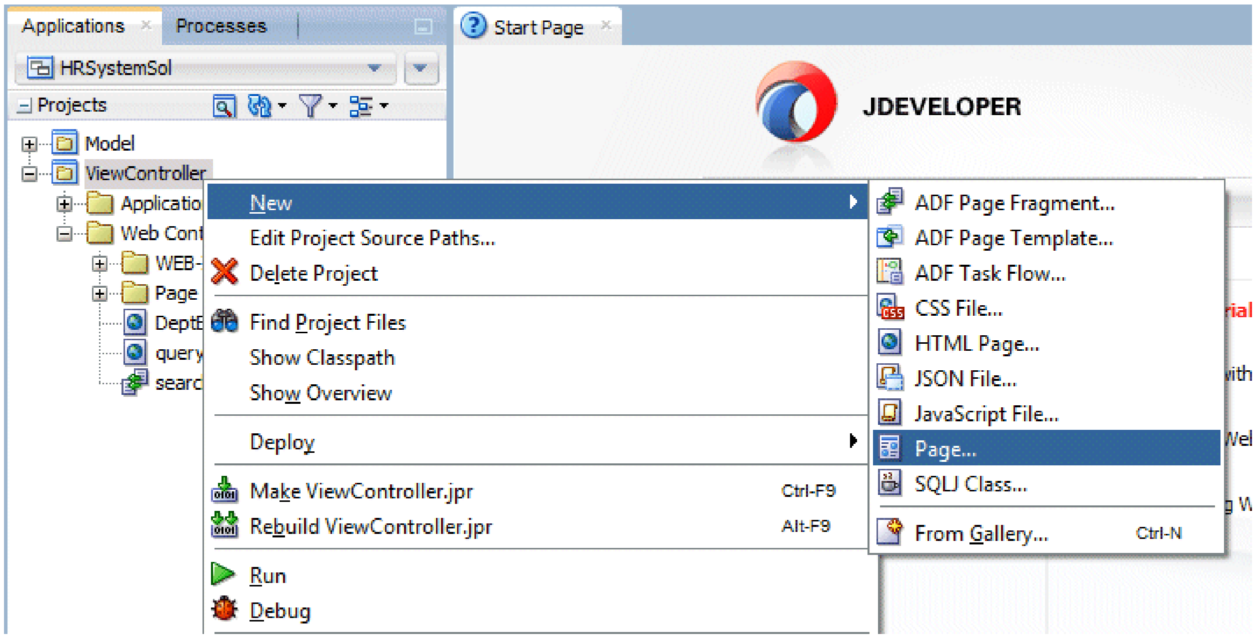


In the popup, click **OK** to complete the conversion process.

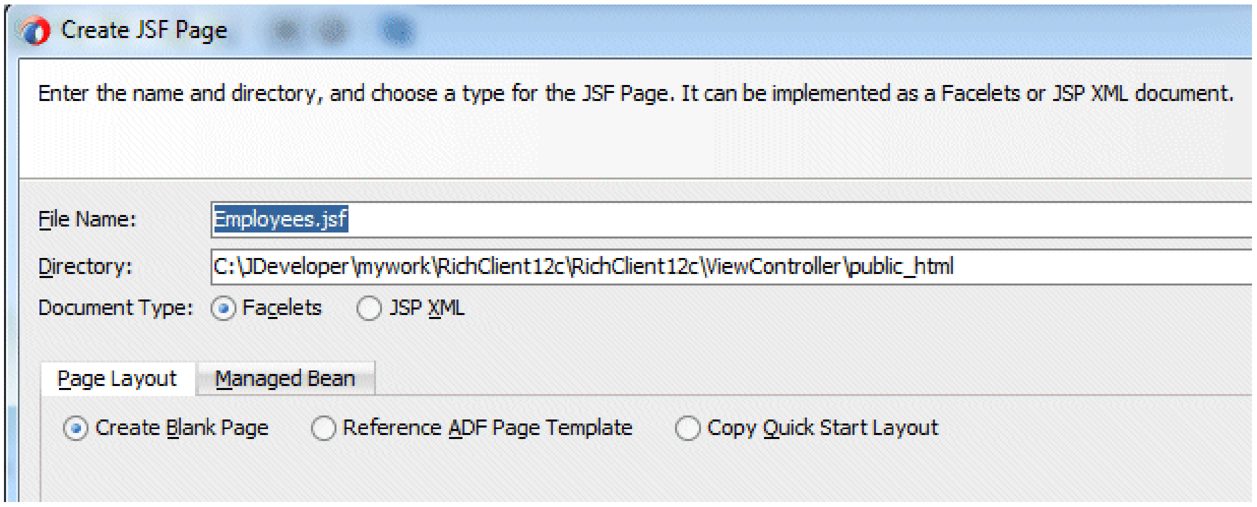
Save all your work.

Now we'll do the same process for an employees page and task flow. Then we'll have two different components to add to the springboard.

12. Right click the View Controller project and in the context menu select, **New - Page...**



13. Name it **Employees.jsf** and create it using a blank page.


Create JSF Page

Enter the name and directory, and choose a type for the JSF Page. It can be implemented as a Facelets or JSP XML document.

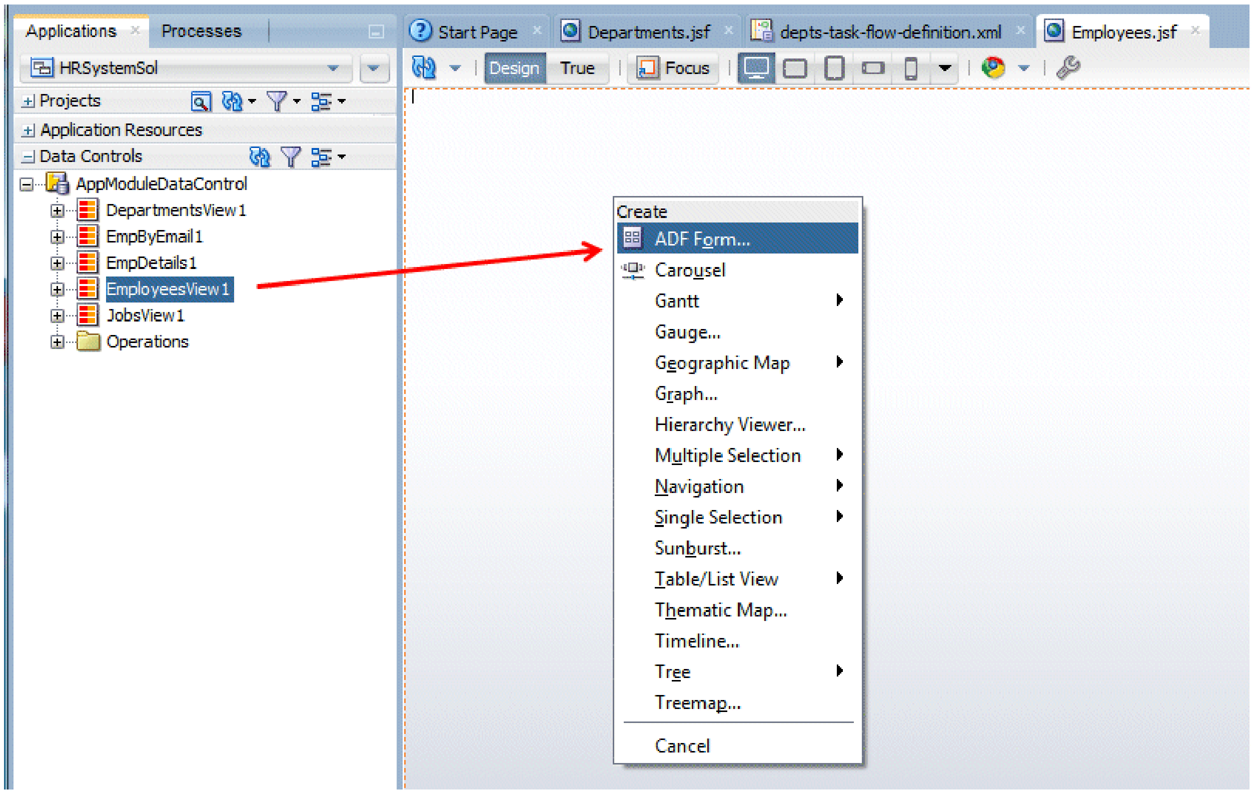
File Name: Employees.jsf

Directory: C:\JDeveloper\mywork\RichClient12c\RichClient12c\ViewController\public_html

Document Type: [✓ Facelets] JSP XML

[✓ Create Blank Page] Reference ADF Page Template Copy Quick Start Layout

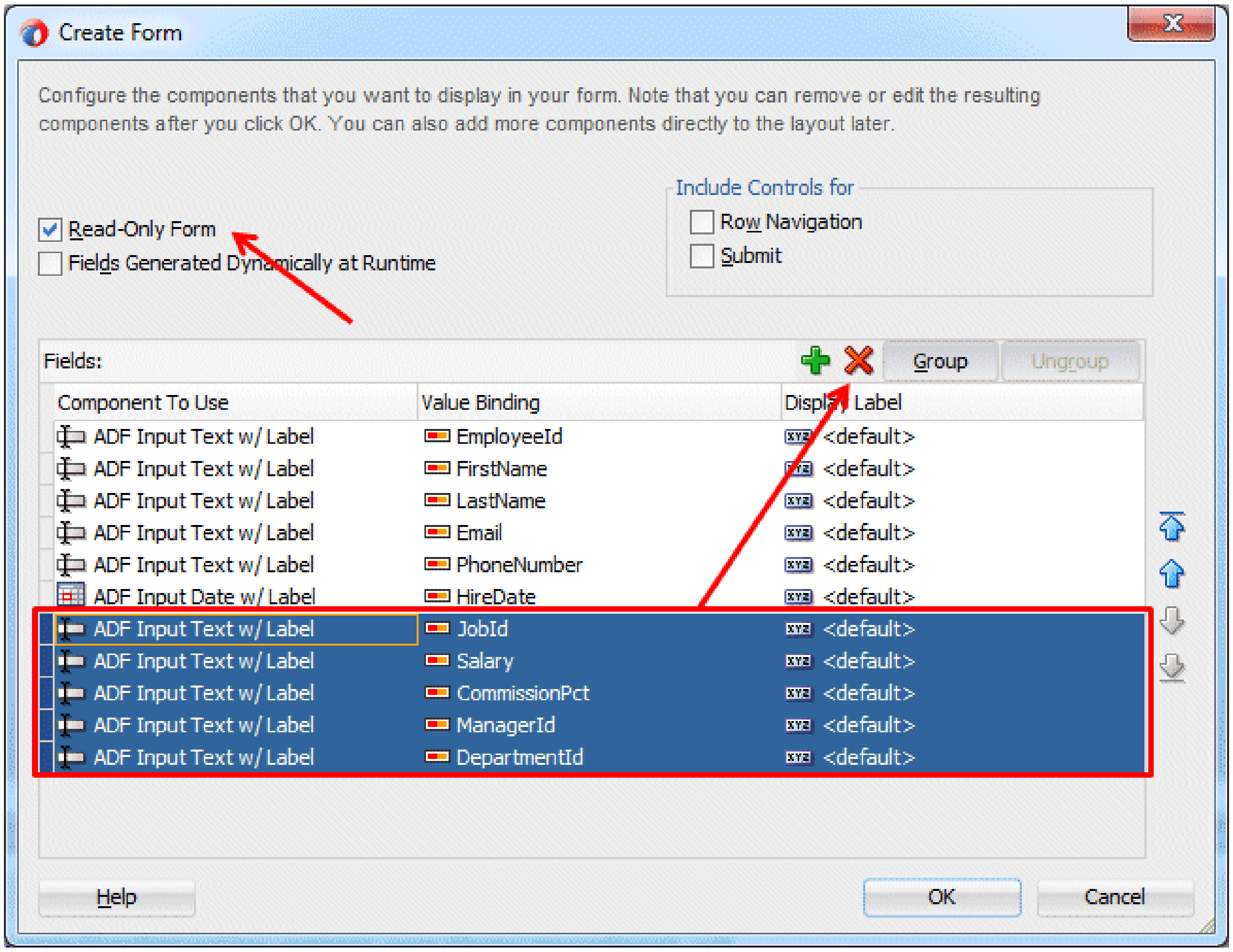
14. From the Data Controls palette, expand the AppModuleDataControl and drag and drop the **EmployeeView1** onto the page.
In the popup menu, select **ADF Form ...**



15. In the Create Form dialog, select the Read-Only Form checkbox and then delete the following fields.

    JobId, Salary, CommissionPct, ManagerId, and **DepartmentId**

    Then click **OK**.


Create Form

Configure the components that you want to display in your form. Note that you can remove or edit the resulting components after you click OK. You can also add more components directly to the layout later.

[✓] Read-Only Form
[ ] Fields Generated Dynamically at Runtime

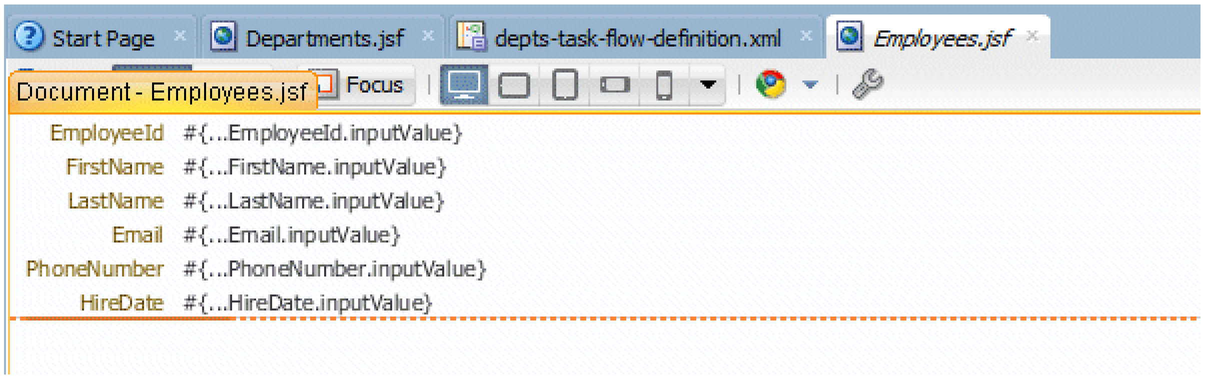
Include Controls for
[ ] Row Navigation
[ ] Submit

Fields:

Component To Use	Value Binding	Display Label
ADF Input Text w/ Label	EmployeeId	<default>
ADF Input Text w/ Label	FirstName	<default>
ADF Input Text w/ Label	LastName	<default>
ADF Input Text w/ Label	Email	<default>
ADF Input Text w/ Label	PhoneNumber	<default>
ADF Input Date w/ Label	HireDate	<default>
ADF Input Text w/ Label	JobId	<default>
ADF Input Text w/ Label	Salary	<default>
ADF Input Text w/ Label	CommissionPct	<default>
ADF Input Text w/ Label	ManagerId	<default>
ADF Input Text w/ Label	DepartmentId	<default>

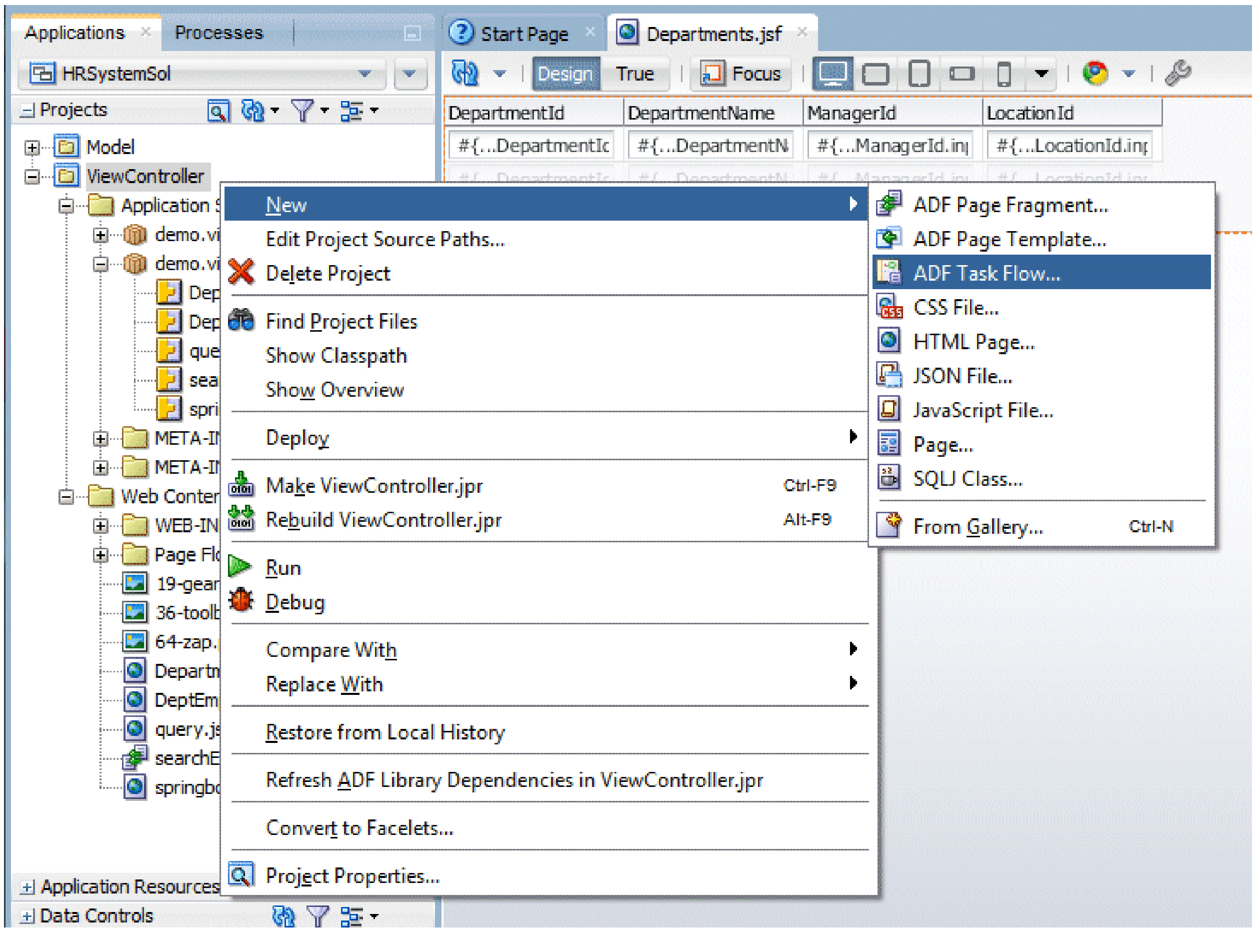
16. The resulting page should look like the image below.

    Save your work.

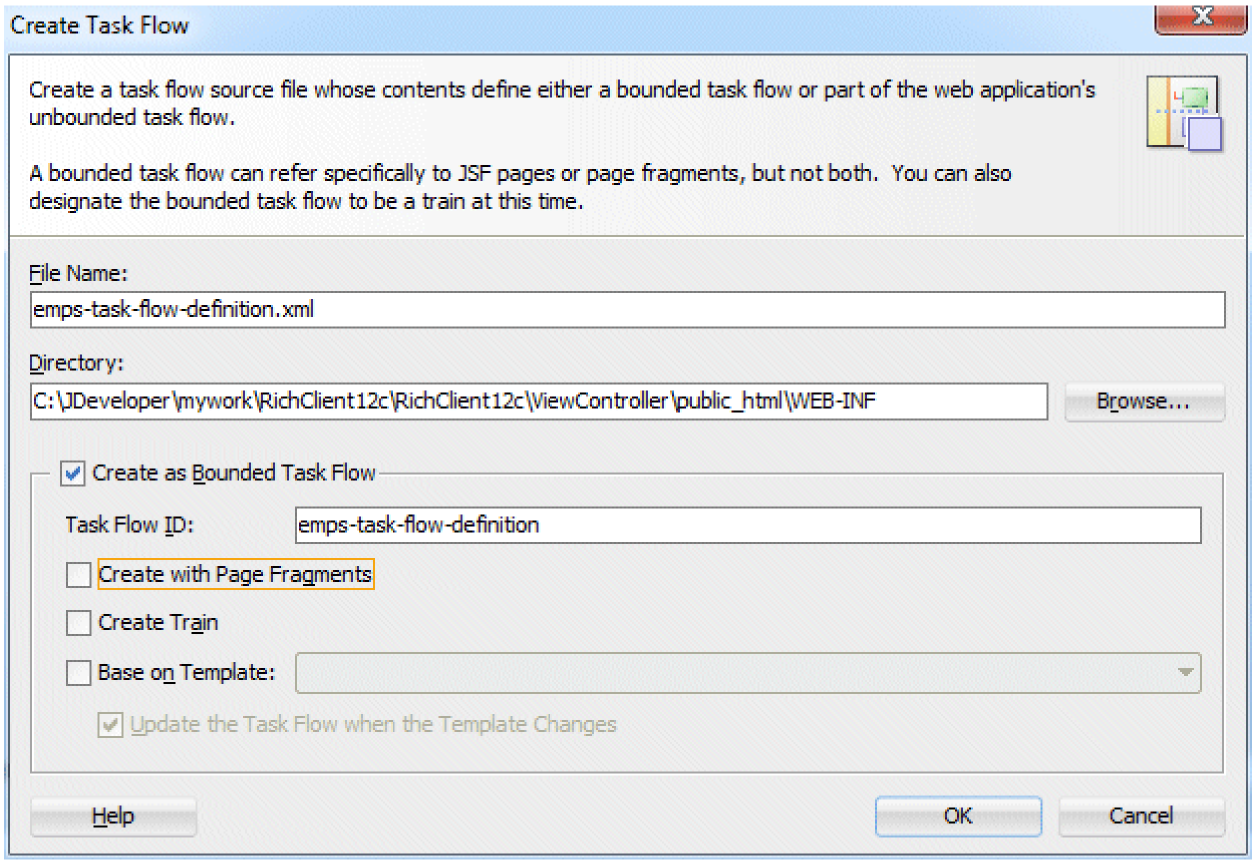


EmployeeId	#{...EmployeeId.inputValue}
FirstName	#{...FirstName.inputValue}
LastName	#{...LastName.inputValue}
Email	#{...Email.inputValue}
PhoneNumber	#{...PhoneNumber.inputValue}
HireDate	#{...HireDate.inputValue}

17. Select the ViewController project, and from the context menu select **New - ADF Task Flow...**



18. Name it **emps-task-flow-definition.xml** and deselect the **Create with Page Fragments** checkbox. Click **OK** to create the task flow.


Create Task Flow

Create a task flow source file whose contents define either a bounded task flow or part of the web application's unbounded task flow.

A bounded task flow can refer specifically to JSF pages or page fragments, but not both. You can also designate the bounded task flow to be a train at this time.

File Name: emps-task-flow-definition.xml

Directory: C:\JDeveloper\mywork\RichClient12c\RichClient12c\ViewController\public_html\WEB-INF

[✓] Create as Bounded Task Flow

Task Flow ID: emps-task-flow-definition

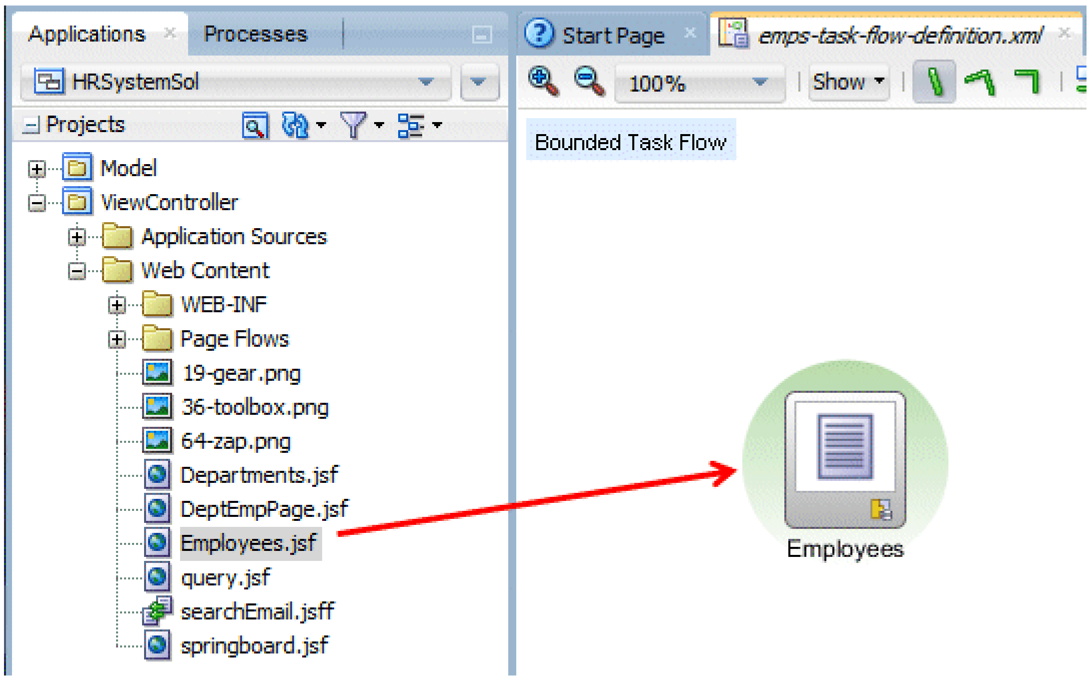
[ ] Create with Page Fragments

[ ] Create Train

[ ] Base on Template:

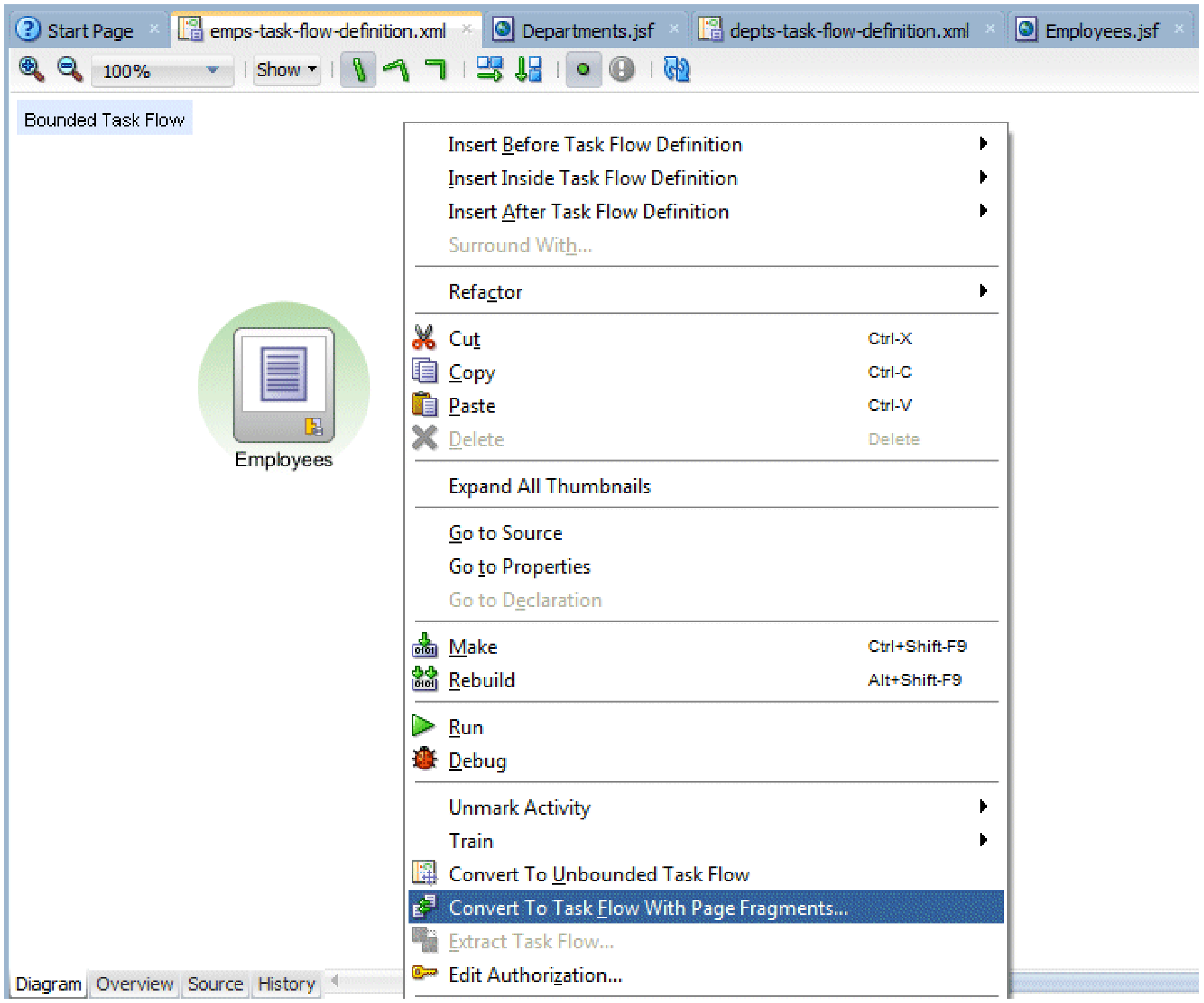
[✓] Update the Task Flow when the Template Changes

19. From the Applications navigator, drag the **Employees.jsf** page onto the task flow and drop it.

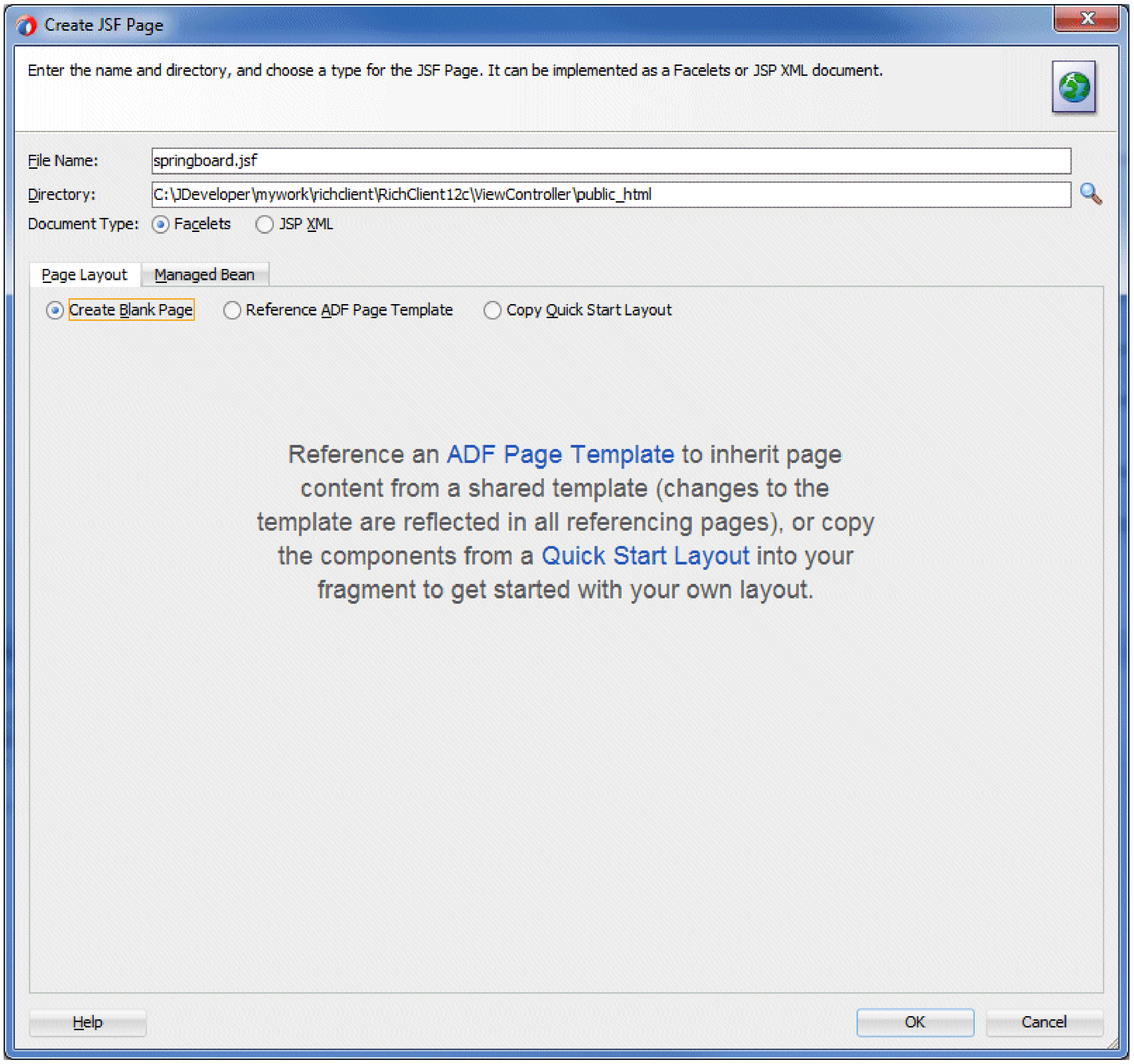


20. Next, right-click inside the task flow and select **Convert To Task Flow With Page Fragments…**

    Remember, later on we'll create a page containing the panel springboard component. We can then add this task flow as a region on the springboard.



21. In the popup, click OK to finish the process.
22. Save your work.
23. Create a page to contain the springboard. Create it in the ViewController project and name it **springboard.jsf** and select the **Create Blank Page** radio button. Then click **OK**.



Create JSF Page

Enter the name and directory, and choose a type for the JSF Page. It can be implemented as a Facelets or JSP XML document.

File Name: springboard.jsf

Directory: C:\JDeveloper\mywork\richclient\RichClient12c\ViewController\public_html

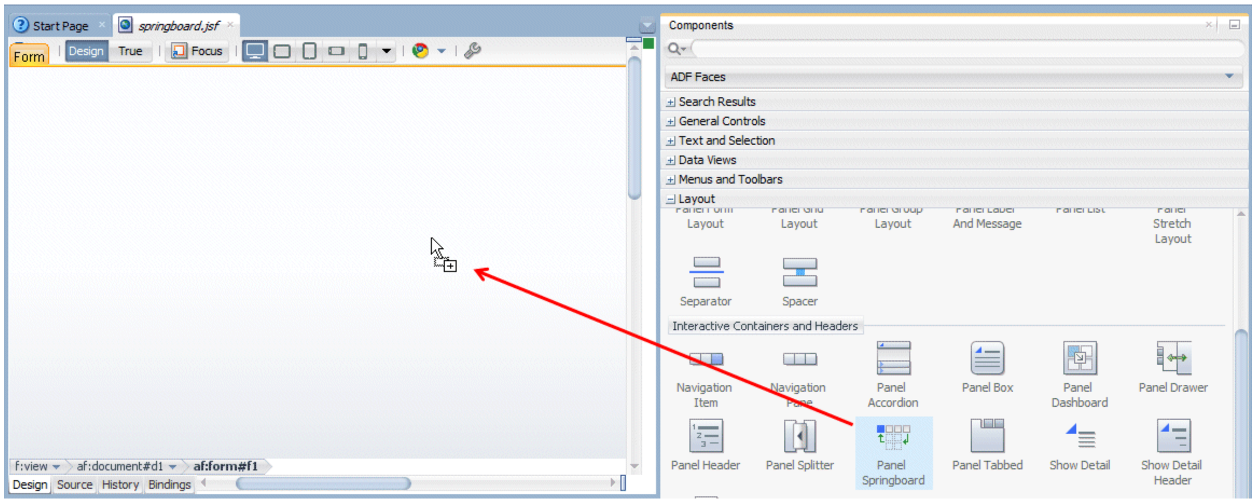
Document Type: [✓ Facelets] JSP XML

Page Layout | Managed Bean

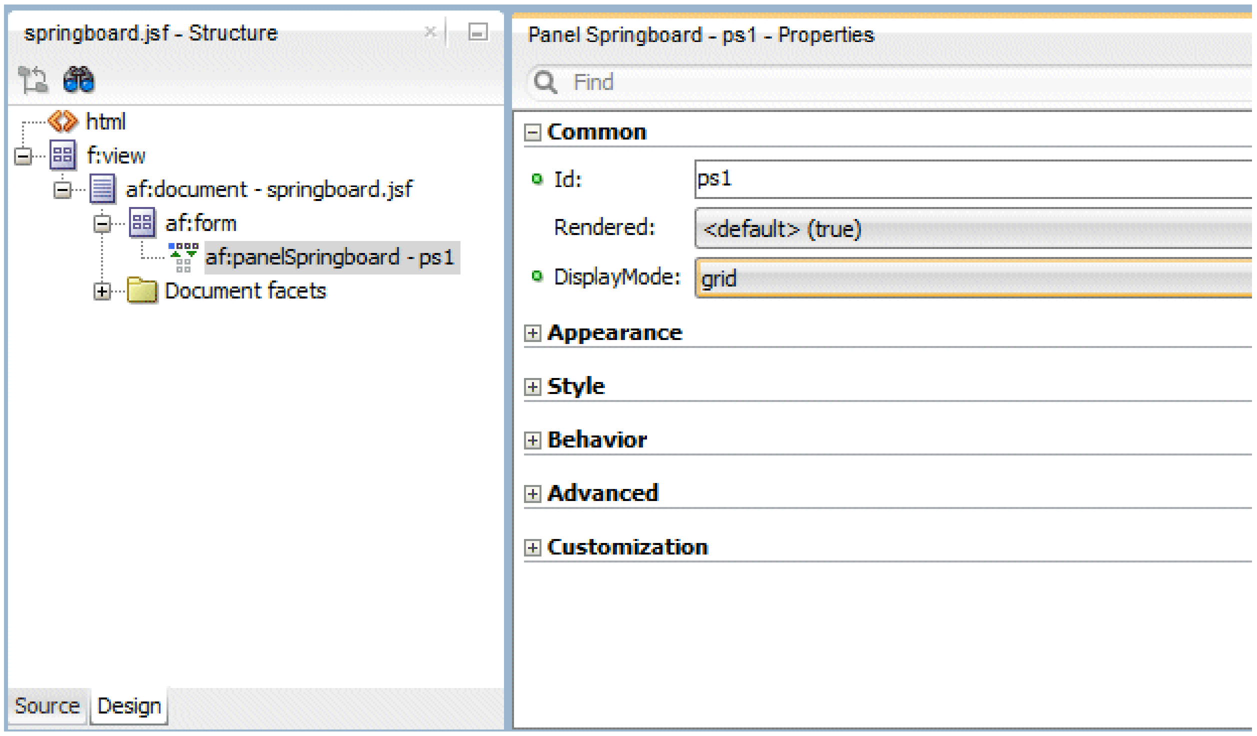
[✓ Create Blank Page] Reference ADF Page Template Copy Quick Start Layout

Reference an ADF Page Template to inherit page content from a shared template (changes to the template are reflected in all referencing pages), or copy the components from a Quick Start Layout into your fragment to get started with your own layout.

24. From the Components window, expand the Layout node and scroll to the Interactive Containers and Headers section. Then, select and drag and drop a Panel Springboard onto the page

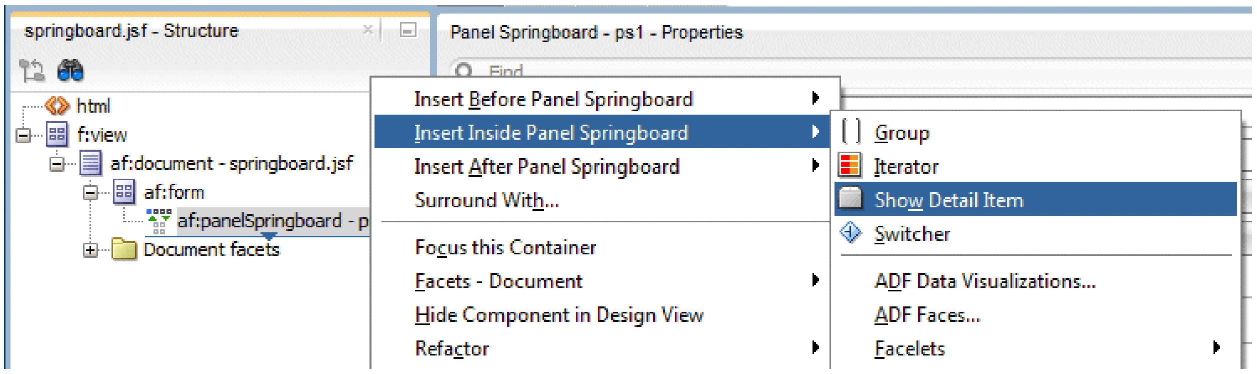


25. Select the panelSpringboard component and in the Properties window, set the Display Mode to **grid**.

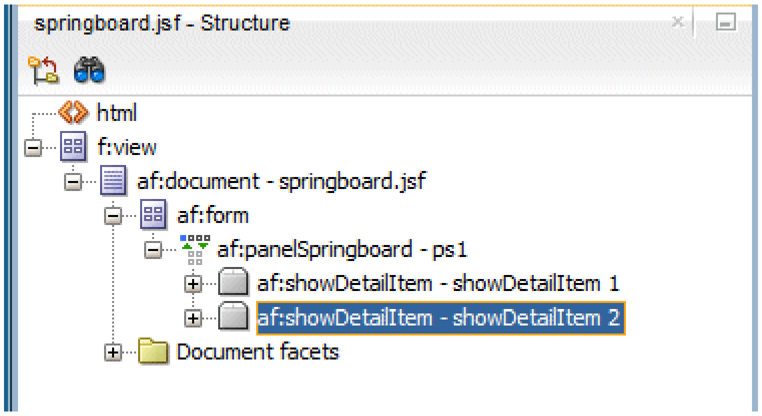

springboard.jsf - Structure
html
f:view
af:document - springboard.jsf
af:form
af:panelSpringboard - ps1
Document facets
Source
Design
Panel Springboard - ps1 - Properties
Find
Common
Appearance
Style
Behavior
Advanced
Customization

Id: ps1
Rendered: <default> (true)
DisplayMode: grid

26. Inside the panelSpringboard component, right click and select **Insert Inside Panel SpringBoard - Show Detail item**


springboard.jsf - Structure
html
f:view
af:document - springboard.jsf
af:form
af:panelSpringboard - p
Document facets
Panel Springboard - ps1 - Properties
Find
Insert Before Panel Springboard
Insert Inside Panel Springboard
Insert After Panel Springboard
Surround With...
Focus this Container
Facets - Document
Hide Component in Design View
Refactor
Group
Iterator
Show Detail Item
Switcher
ADF Data Visualizations...
ADF Faces...
Facelets

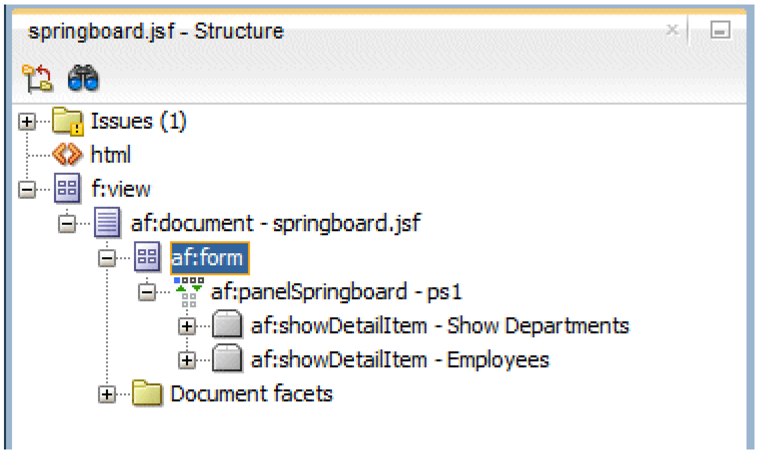
Add another detail item to the springboard


springboard.jsf - Structure
html
f:view
af:document - springboard.jsf
af:form
af:panelSpringboard - ps1
af:showDetailItem - showDetailItem 1
af:showDetailItem - showDetailItem 2
Document facets

27. For each of the showDetailItems, use the Properties window to set the **Text** values.

showDetailItem 1 = **Show Departments**
showDetailItem 2 = **Employees**

Save your work.

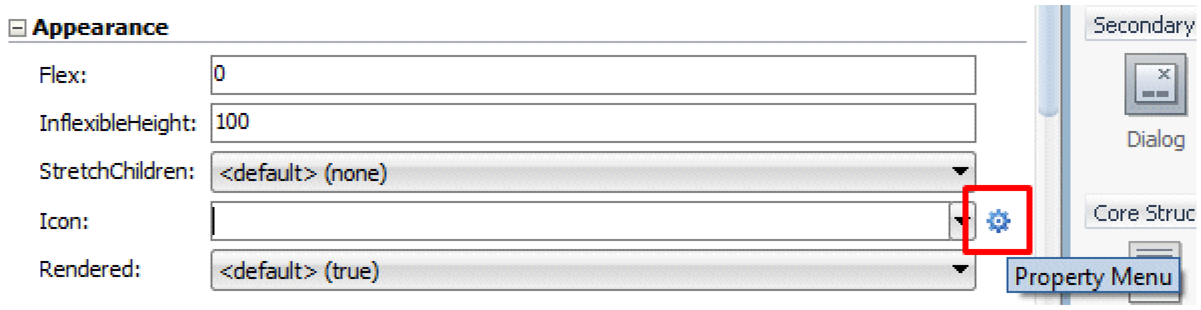

springboard.jsf - Structure
Issues (1)
html
f:view
af:document - springboard.jsf
af:form
af:panelSpringboard - ps1
af:showDetailItem - Show Departments
af:showDetailItem - Employees
Document facets

28. For each showDetailItem, select and set the **Icon** property to the image names as follows..

Show Departments = **36-toolbox.png**

Employee = **19-gear.jpg**

To set the property, move your cursor to the right side of the **Icon** property and click on the blue gear.



Appearance

Flex: 0

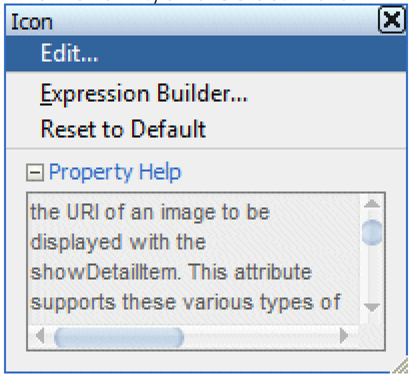
InflexibleHeight: 100

StretchChildren: <default> (none)

Icon:

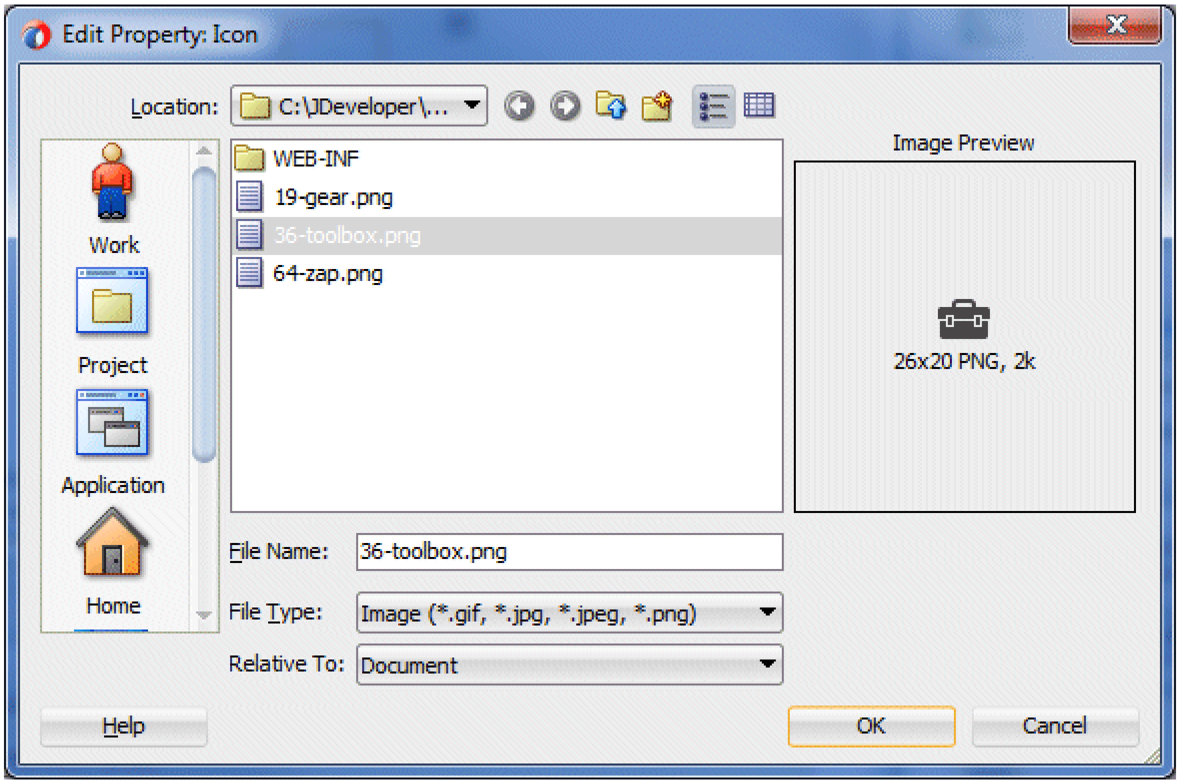
Rendered: <default> (true)

Then click it, and select **Edit**.

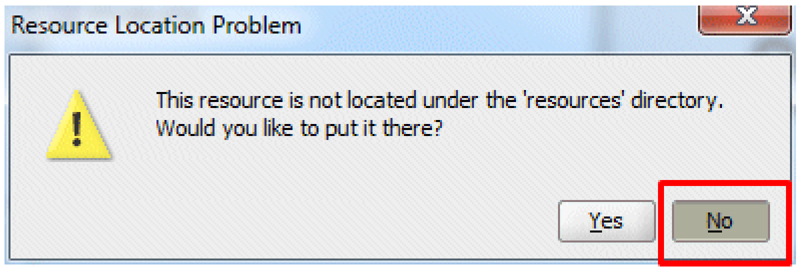
29. 

the URI of an image to be displayed with the showDetailItem. This attribute supports these various types of

Select the appropriate image and click **OK**.



When prompted to allow the image to be put in the **'resources'** directory, click **No**.

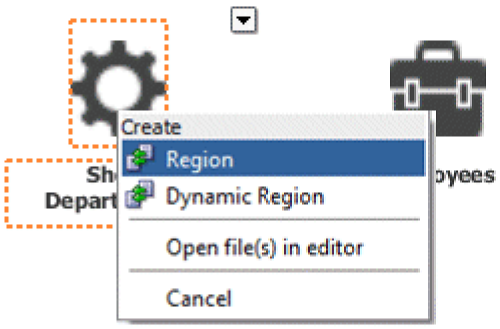

Resource Location Problem

This resource is not located under the 'resources' directory.
Would you like to put it there?

The next step is to determine and associate what page or task flow the springboard item will invoke. You can do this by dragging and and dropping a page or task flow onto the springboard detail item.

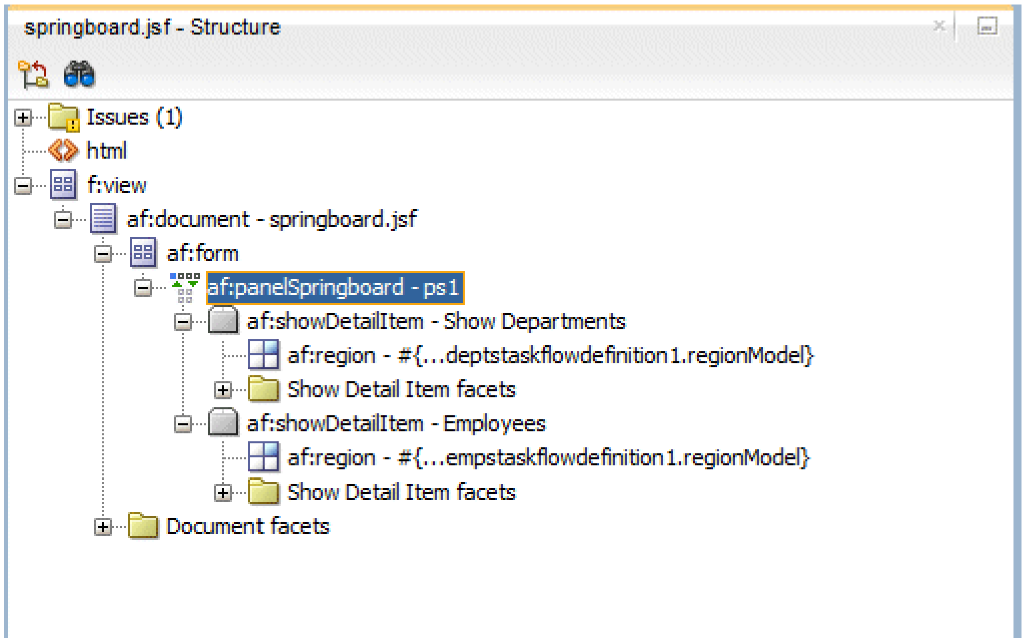
30. Drag **depts-task-flow-definition** and drop it on the **Show Departments** detail item.

When you release the mouse button, you're prompted for how you want the item created. In our case, create the item as a **Region**.



31. Drag the **emps-task-flow-definition** to the Employees detailItem and create it as a Region.

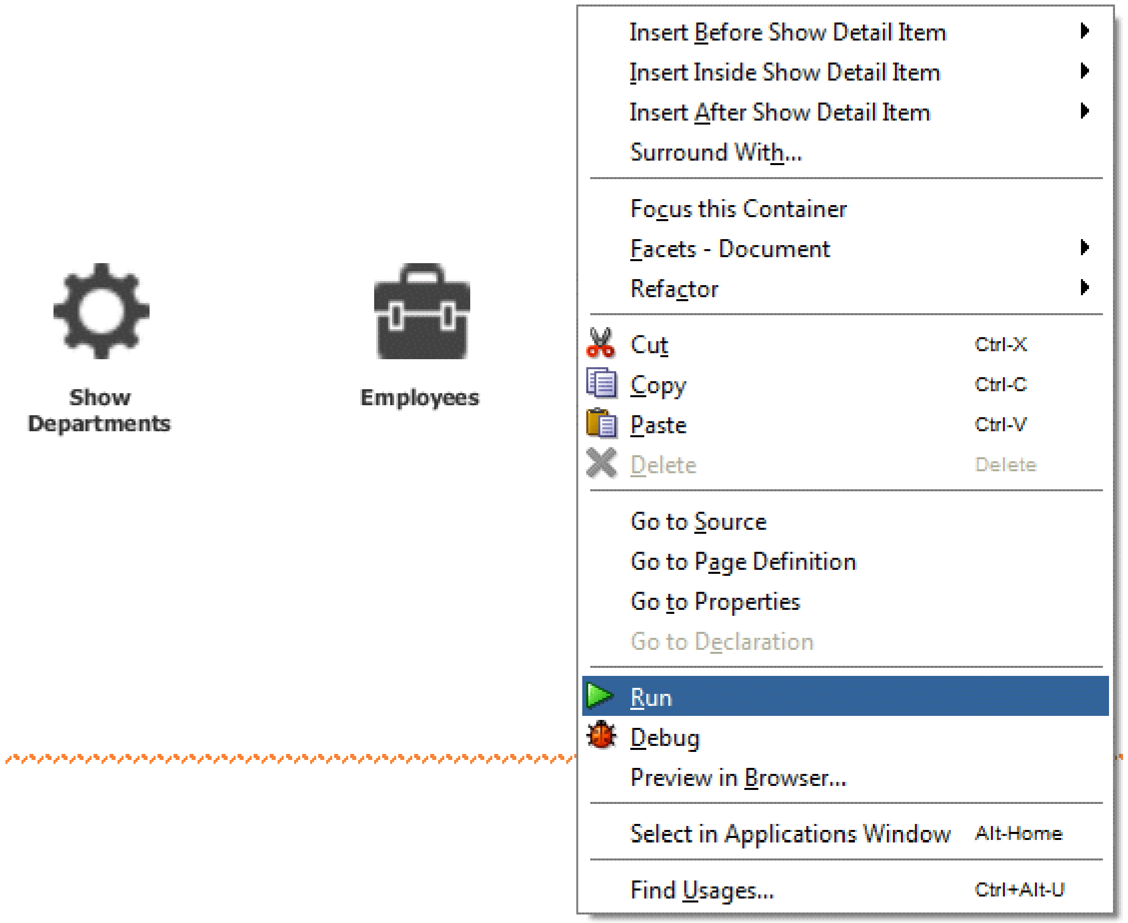
32. Save your work.



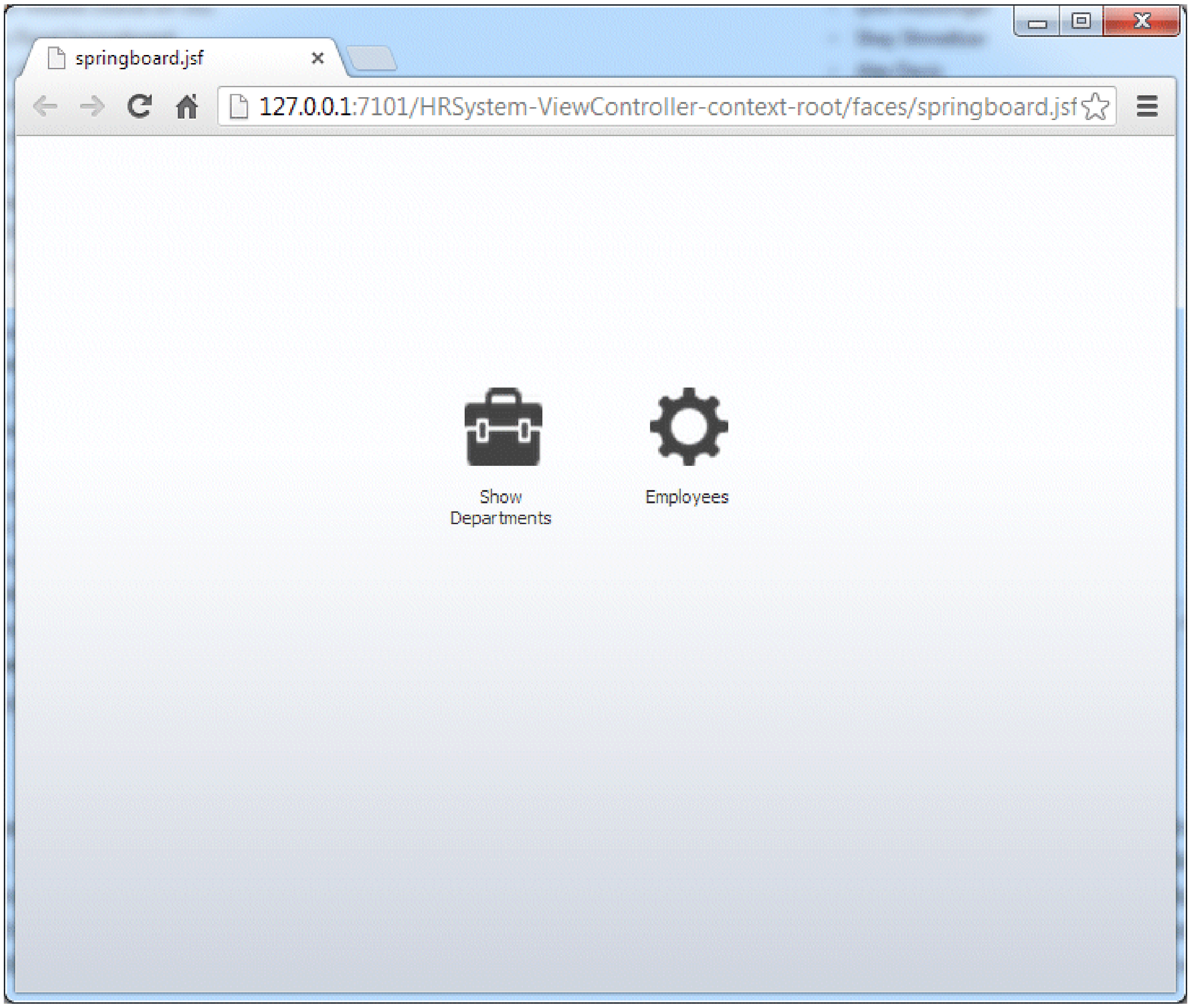
springboard.jsf - Structure

```
Issues (1)
html
f:view
  af:document - springboard.jsf
    af:form
      af:panelSpringboard - ps1
        af:showDetailItem - Show Departments
          af:region - #{...deptstaskflowdefinition1.regionModel}
          Show Detail Item facets
        af:showDetailItem - Employees
          af:region - #{...empstaskflowdefinition1.regionModel}
          Show Detail Item facets
    Document facets
```

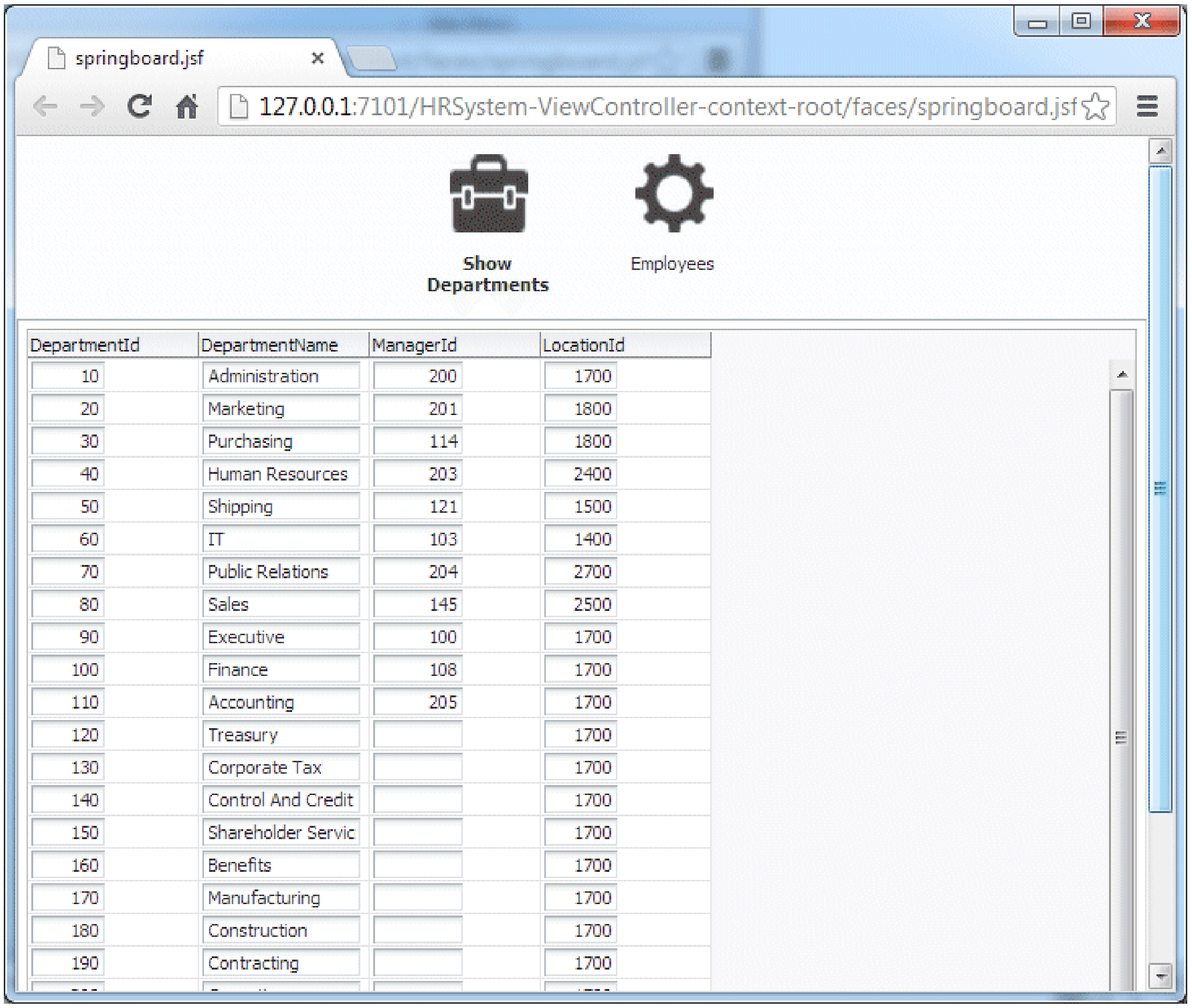
33. Finally, right click in the **springboard.jsf** page and select **Run**.



34. Click on each of the icons to see the see how the Springboard allows you to launch individual task flows.

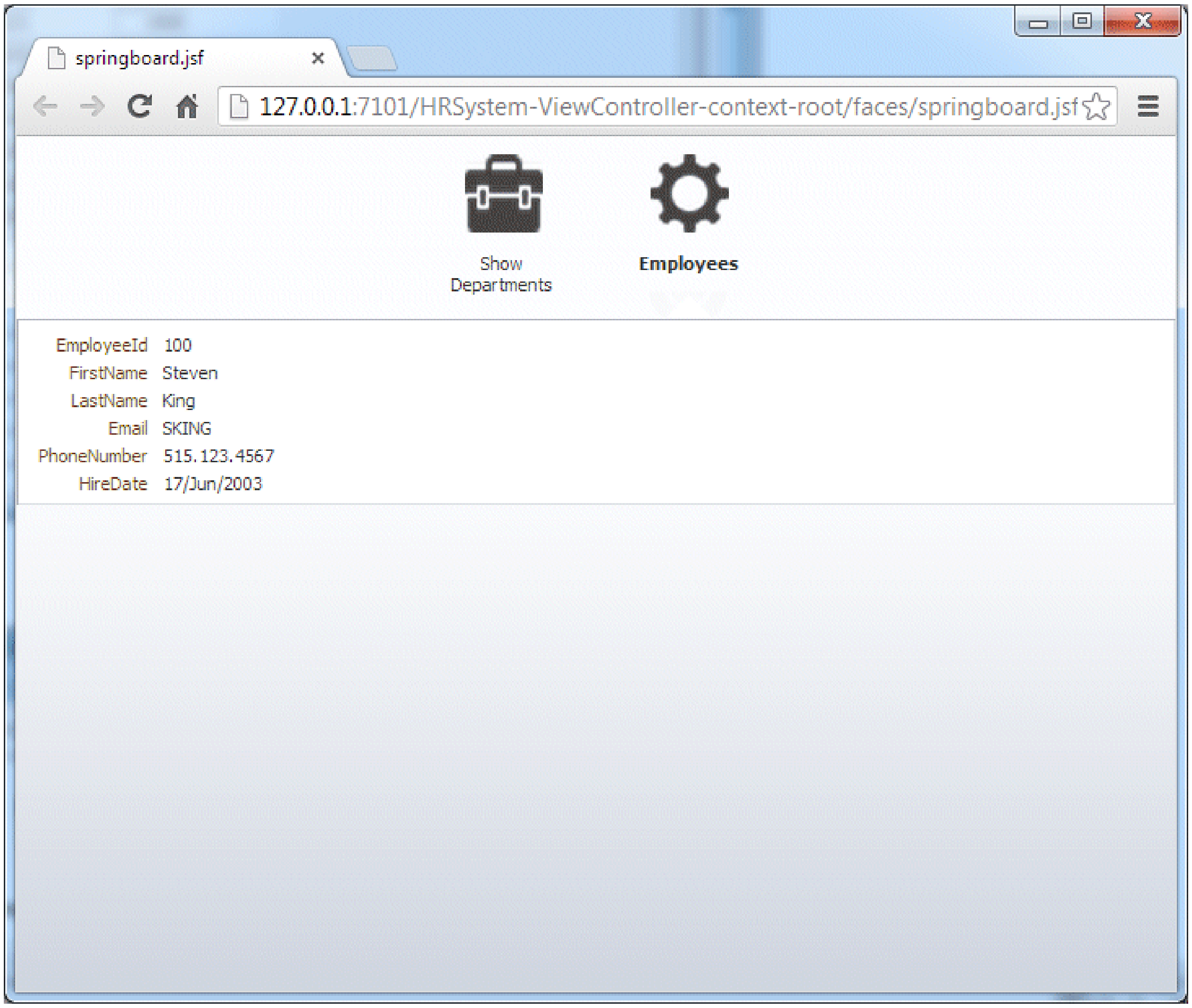

springboard.jsf
127.0.0.1:7101/HRSystem-ViewController-context-root/faces/springboard.jsf
Show Departments
Employees

When one item is selected the icons move to the top of the browser, and the page details appear below.


springboard.jsf
127.0.0.1:7101/HRSystem-ViewController-context-root/faces/springboard.jsf
Show Departments
Employees

DepartmentId	DepartmentName	ManagerId	LocationId
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1800
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury		1700
130	Corporate Tax		1700
140	Control And Credit		1700
150	Shareholder Servic		1700
160	Benefits		1700
170	Manufacturing		1700
180	Construction		1700
190	Contracting		1700

Click another icon and the page switches.


springboard.jsf
127.0.0.1:7101/HRSystem-ViewController-context-root/faces/springboard.jsf
Show Departments
Employees

EmployeeId	100
FirstName	Steven
LastName	King
Email	SKING
PhoneNumber	515.123.4567
HireDate	17/Jun/2003

35. Close your browser and exit JDeveloper.

Summary

In this tutorial you built a small Web application that interacts with a database. You learned how to:

- Build the business services that supply the data to the application
- Create a data-bound JSF page
- Enhance the page by adding more complex operations
- Add a second JSF page to the application and create a task flow to define the navigation between the pages
- Create a reusable page fragment containing a business service based on parameters
- Add a springboard to launch your application components

Courtesy: https://docs.oracle.com/cd/E37547_01/tutorials/tut_rich_app/tut_rich_app_3.html

Modified: 2021.10.04.7.45.AM
Dököll Solutions, Inc.