

*The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.*



## **HOL9967 - Developing Applications for Mobile iOS and Android Devices with Oracle ADF Mobile *Hands-On Lab***

### **Table of Contents**

|  |           |
|--|-----------|
| <b>Introduction .....</b>                            | <b>2</b>  |
| <b>Create a Project .....</b>                        | <b>3</b>  |
| <b>Setup Data Control .....</b>                      | <b>5</b>  |
| <b>Create ADF Mobile Feature and Task Flow .....</b> | <b>17</b> |
| <b>Setup the List Page .....</b>                     | <b>22</b> |
| <b>Setup Details Page .....</b>                      | <b>30</b> |
| <b>Set Bindings for Details Page .....</b>           | <b>42</b> |
| <b>Deploy to the iOS Simulator .....</b>             | <b>48</b> |
| <b>Extra Credit .....</b>                            | <b>53</b> |
| <b>Finished Early? .....</b>                         | <b>67</b> |

## Introduction

This lab shows how to build a simple List – Details application using ADF Mobile. In the steps that follow, you'll create an application and deploy it to the Apple iOS Simulator. If you run out of time, there's a completed lab in a folder on the desktop.

The “Extra Credit” section shows how to deploy this same application to the Android Emulator as well.

Finally, a more complex variation of this application is available in a sample application (HR - Human Resources Demo) described in the “Finished Early?” section at the end of this document. That sample works on tablet form factors and uses SQLite DB – worth a look if you have time.

***Note: this assumes that JDeveloper and the Mobile SDKs are installed and configured properly. Also, you will need to download the sample POJO classes separately from OTN. Please see ADF Mobile landing Page on OTN for details.***

### What's on the Mac in front of you?

JDeveloper 11g Release 3 beta is installed on your lab system along with a pre-release version of the ADF Mobile extension.

Paths to Apple Xcode 4.4.1 are already setup inside JDeveloper. That's required to deploy to the iOS Simulator, but we've taken care of that for you.

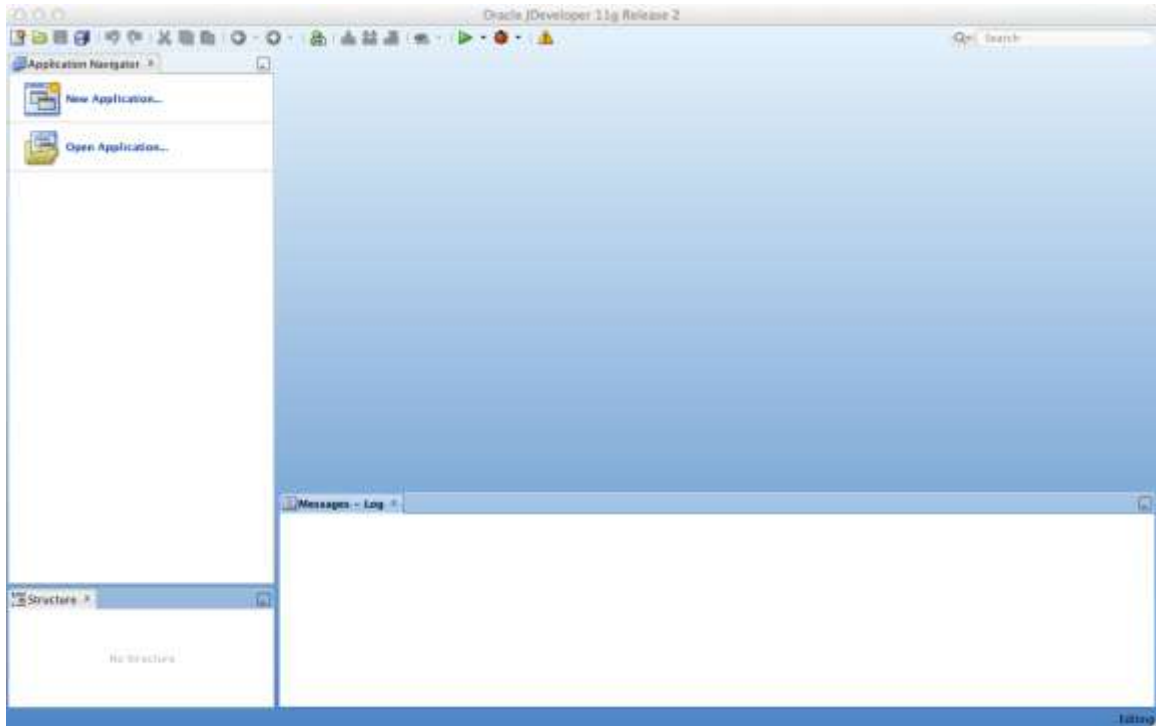
To start JDeveloper click the icon at the bottom of the screen (circled in image below).



## Create a Project

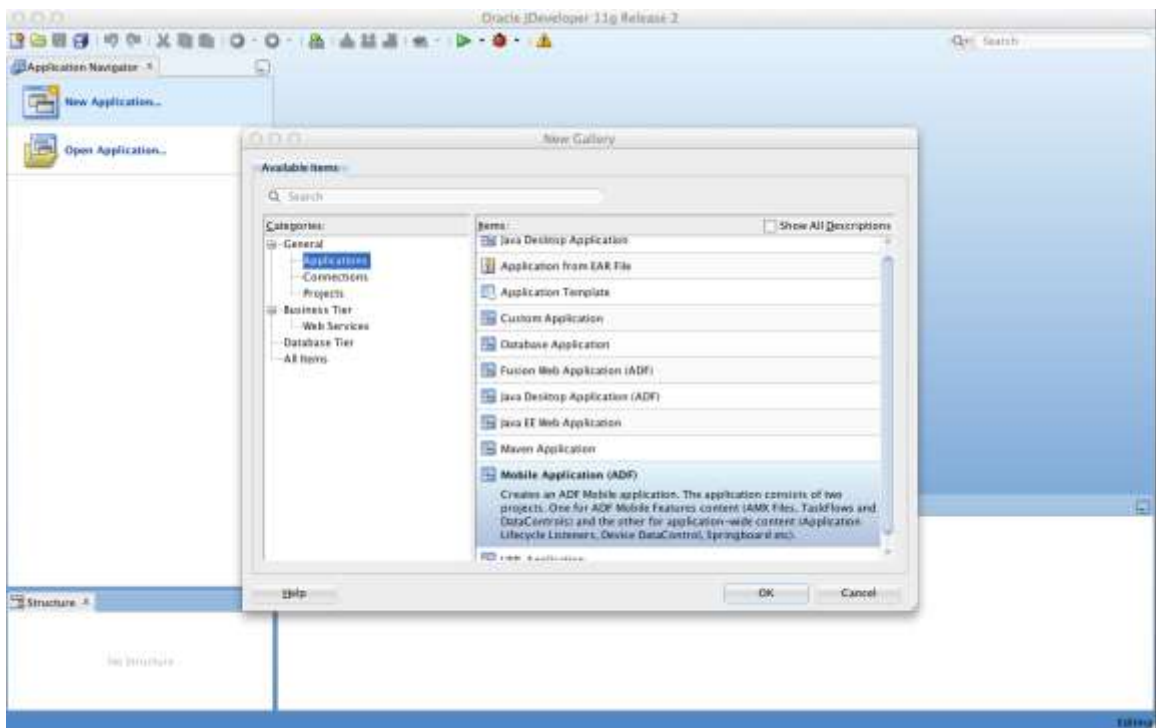
1. Open JDeveloper (click the icon shown below).





2. Choose **New Application**.

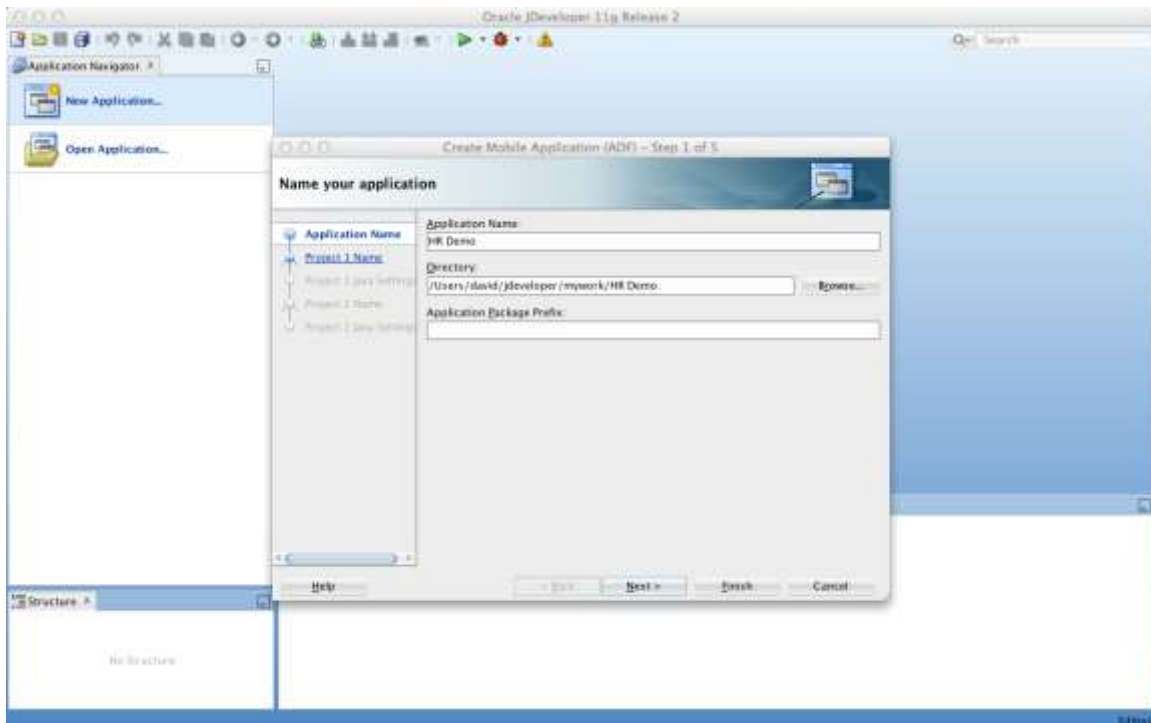
The **New Gallery** dialog appears.



3. Select **Mobile Application (ADF)** and click **OK**.

The **Create Mobile Application (ADF) – Step 1 of 5** dialog appears.

4. In the **Application Name** input box, change the application name from the default to “HRDemo”. Please note there is no space between the letters HR and Demo – the screen shot below contains a typo.



5. Click **Finish**.

(Thus, accepting all the default settings for the app. If you want to click **Next** to cycle through the dialogs, that's fine too, just don't change the defaults.)

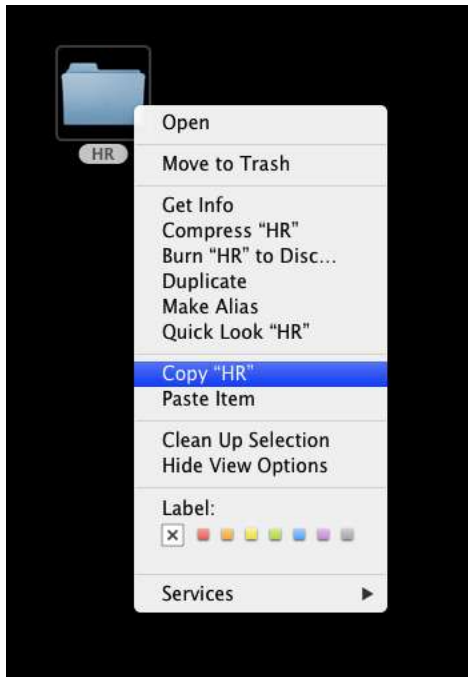
JDeveloper will create and open the application. By default the first file opened is **adfmf-features.xml**. This file is very important we'll come back to it shortly. First we'll copy in a data model for a simple HR organization.

## Setup Data Control

In preparation for this lab we've created some JavaBeans that represent a simple data model. You're going to add these to your project and use them as a source of data for lists of departments and employees.

Building these JavaBeans takes more time than we have available in this short lab, but you'll find it's actually pretty easy. And the pattern you're learning today in using the Data Control is similar to the techniques for working with SQLite and Web Services.

1. Minimize the JDeveloper application window.
2. Control-click the HR folder on the desktop and choose Copy "HR".



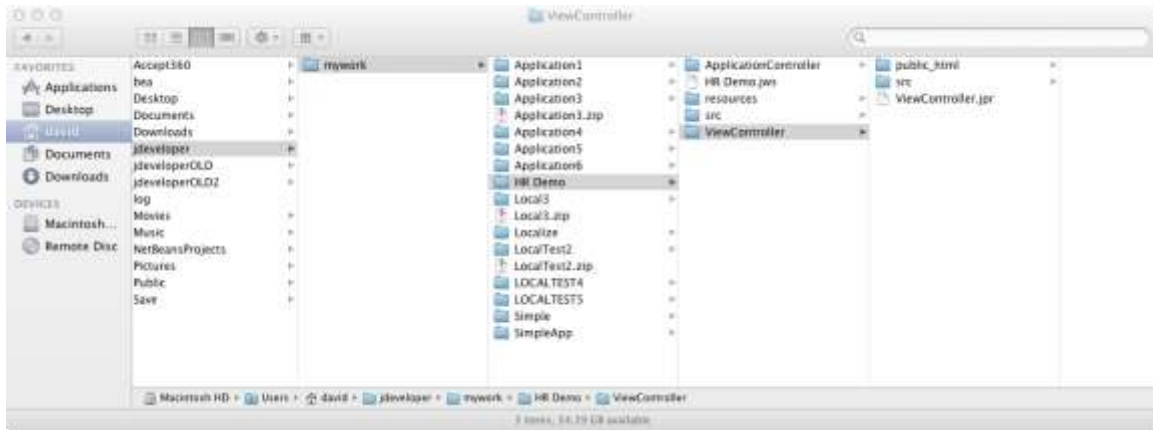
2. Open the **Finder** Application.

The Finder icon looks like this at the far left on your desktop.



Navigate to the folder `//Admin/jdeveloper/mywork/HR Demo/ViewController/src`

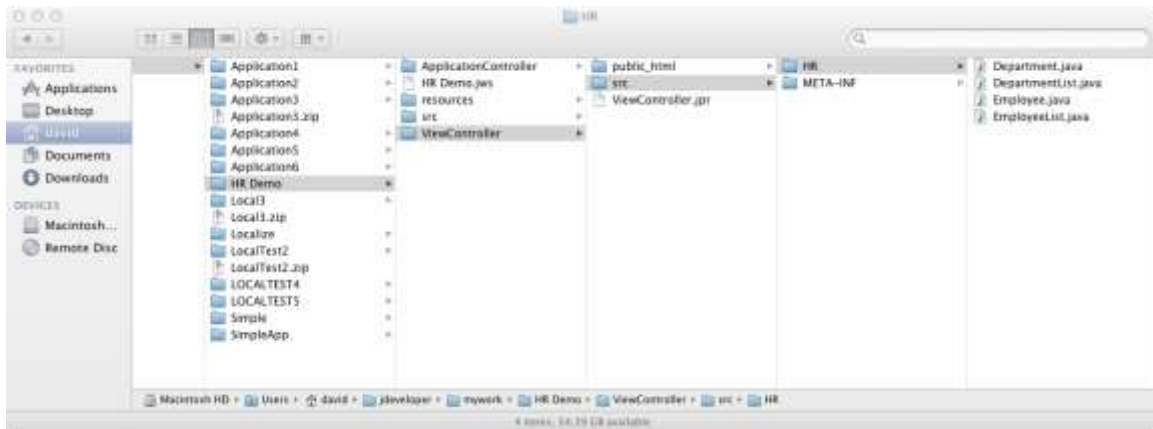
This should appear similar to the following.



(Unlike the screenshots here, the mac used for the lab will say Admin rather than David.)

4. Control-click inside the src folder and choose **Paste Item**.

The HR folder should now appear inside src, as follows:

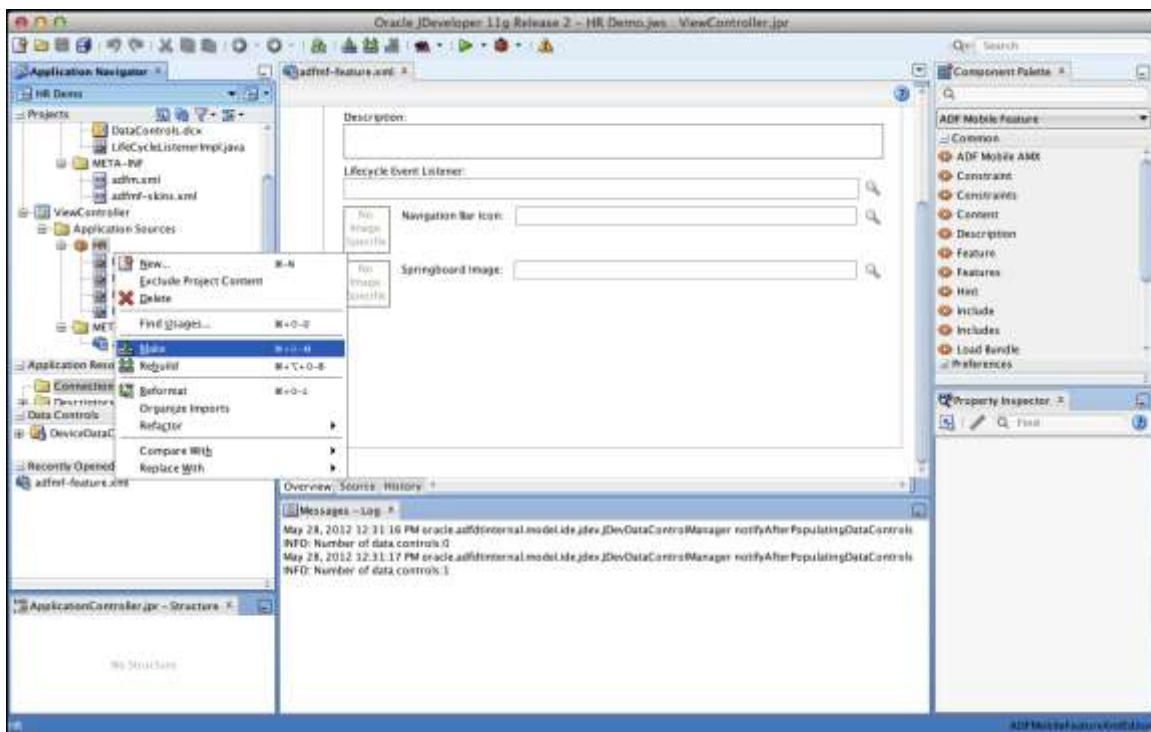


5. Return to JDeveloper (i.e. click the JDeveloper icon at the bottom of the screen).

6. Click the **Refresh** icon and observe that HR now appears under **Application Sources**.

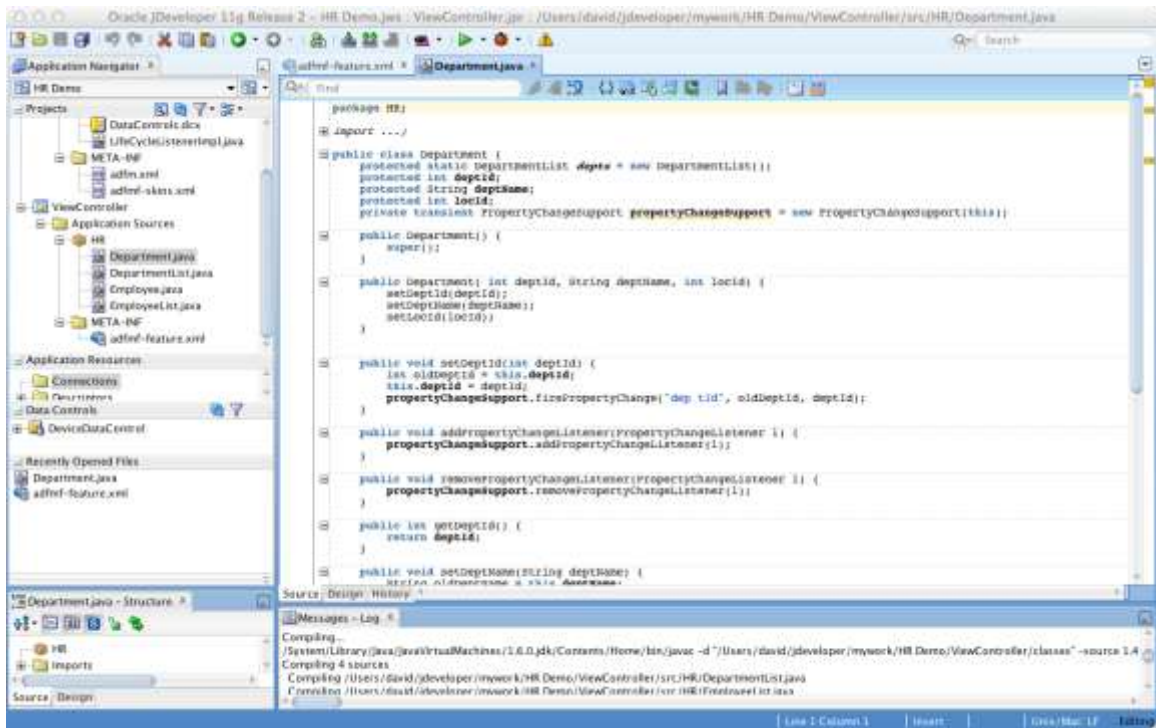


7. Control-click HR and choose **Make**.

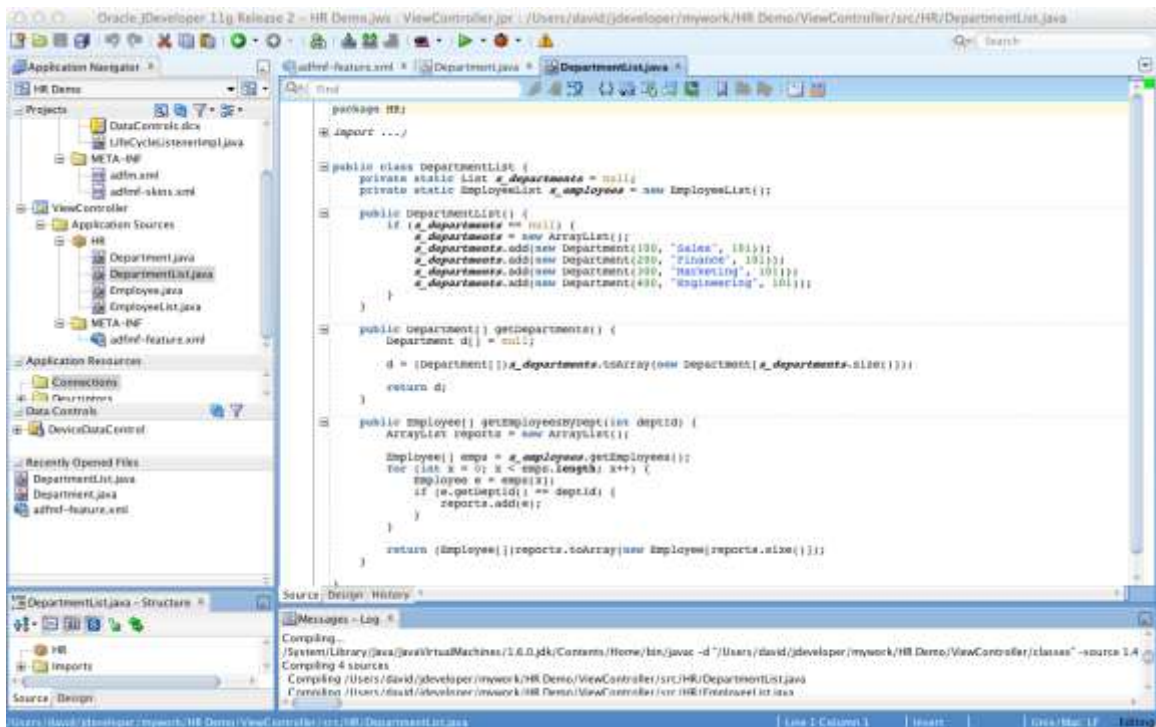


8. Double-click each of the Java files, examine their source, and skim them to get a sense of what they do.

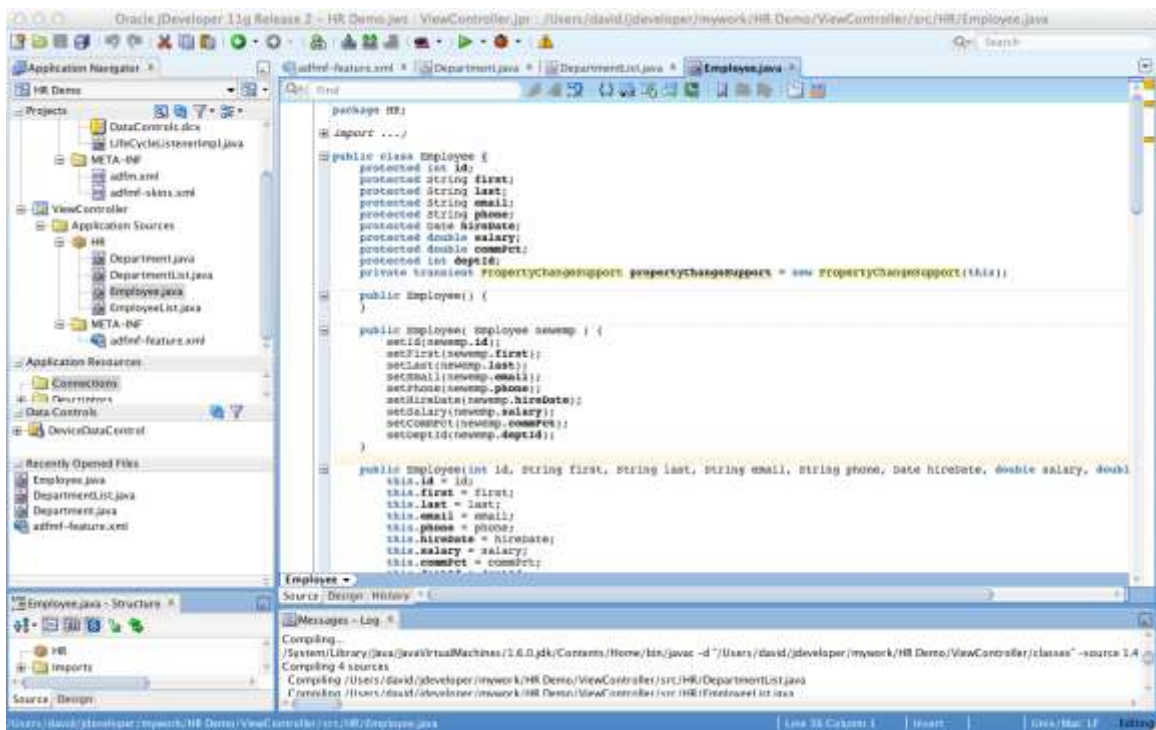
A Department has properties corresponding to name, ID, and location.



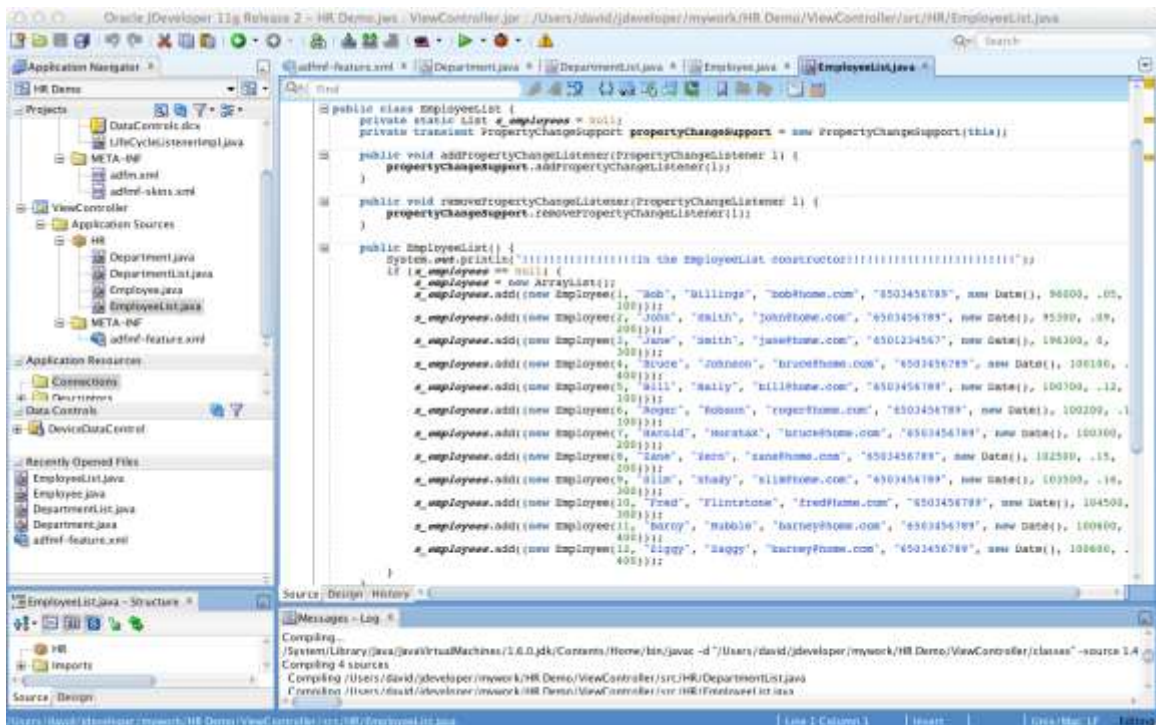
A DepartmentList is a collection of Departments. And there are four departments added to the collection.



An Employee has several properties, corresponding to ID, first and last name, email, phone, hire date, commission, and department ID.

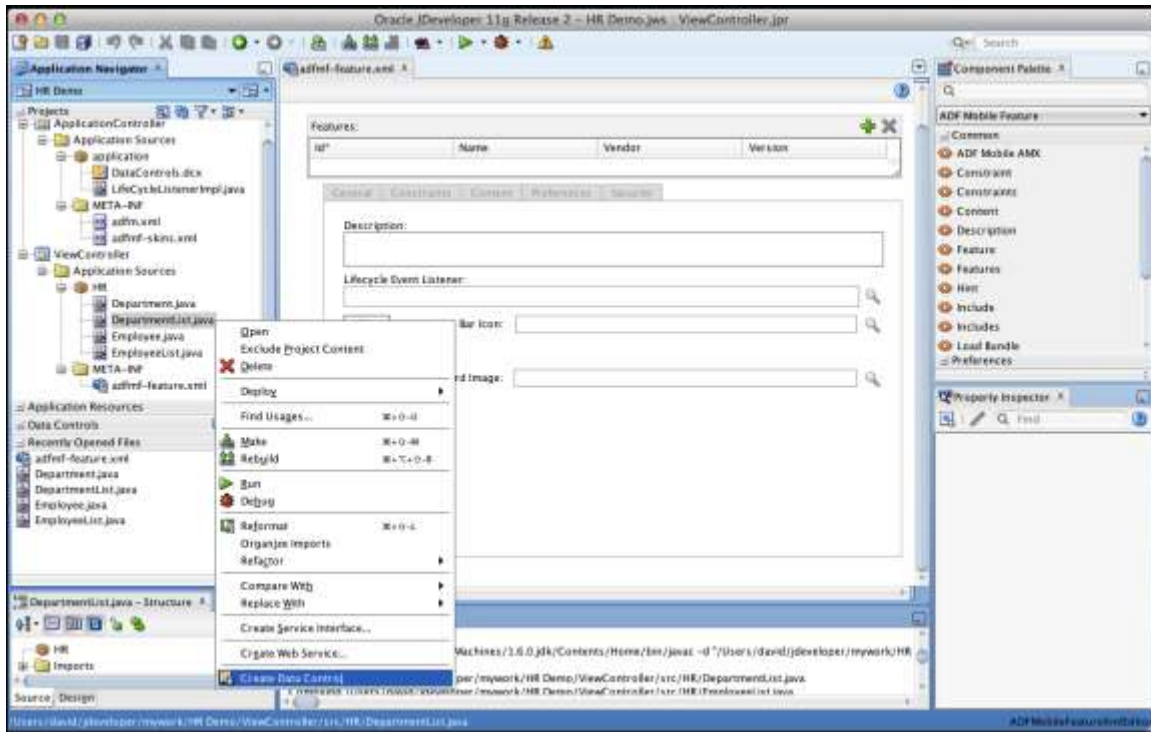


Finally, EmployeeList is a collection of Employees. Observe there are about a dozen employees that are created in this dataset.

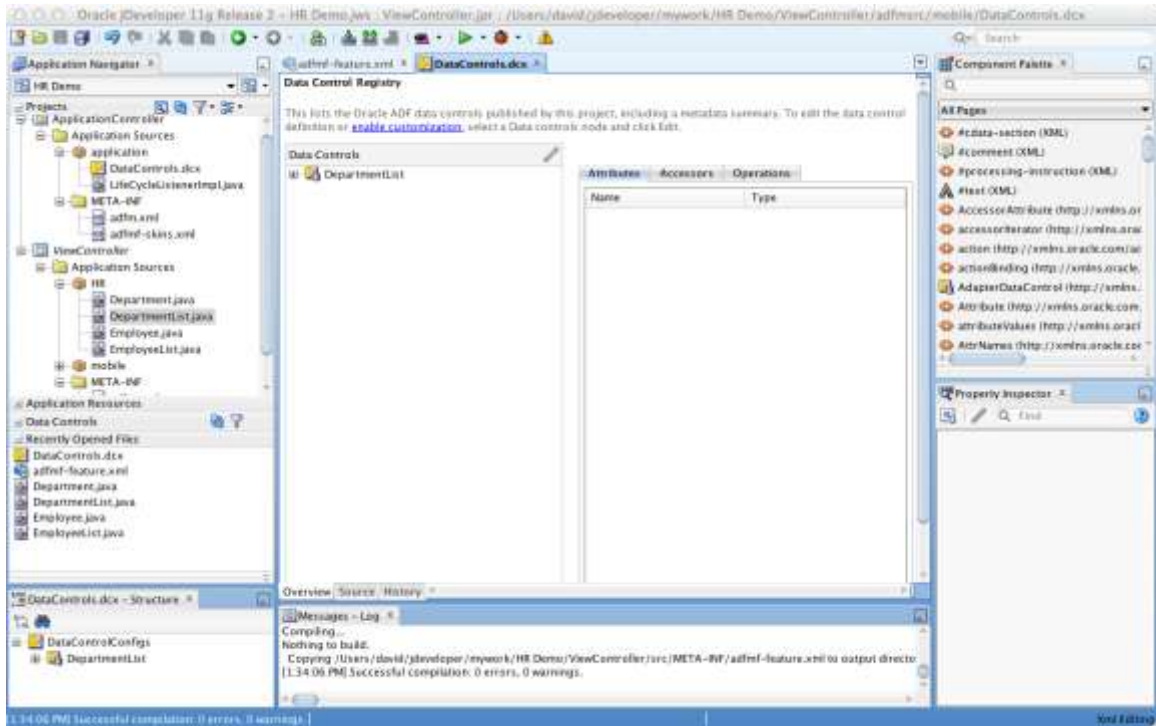


9. Close each of the Java files.

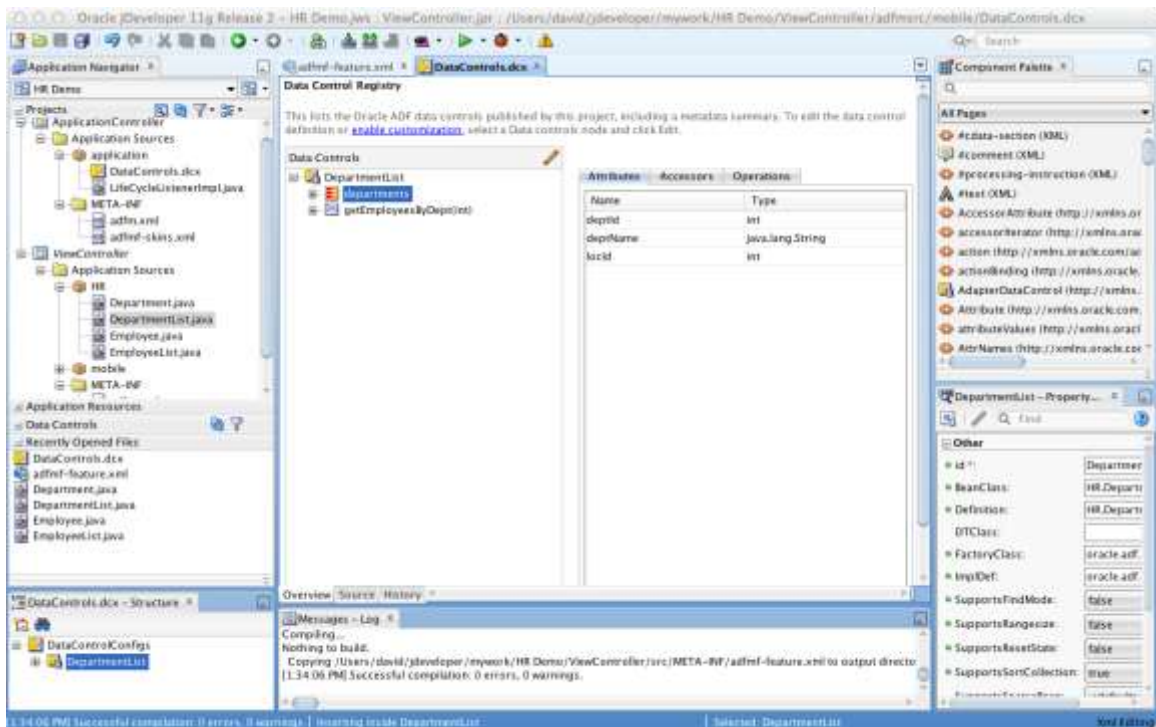
10. Control-click DepartmentList.java and choose **Create Data Control**.



Observe the **Data Control Registry** opens and you should see DepartmentList there.

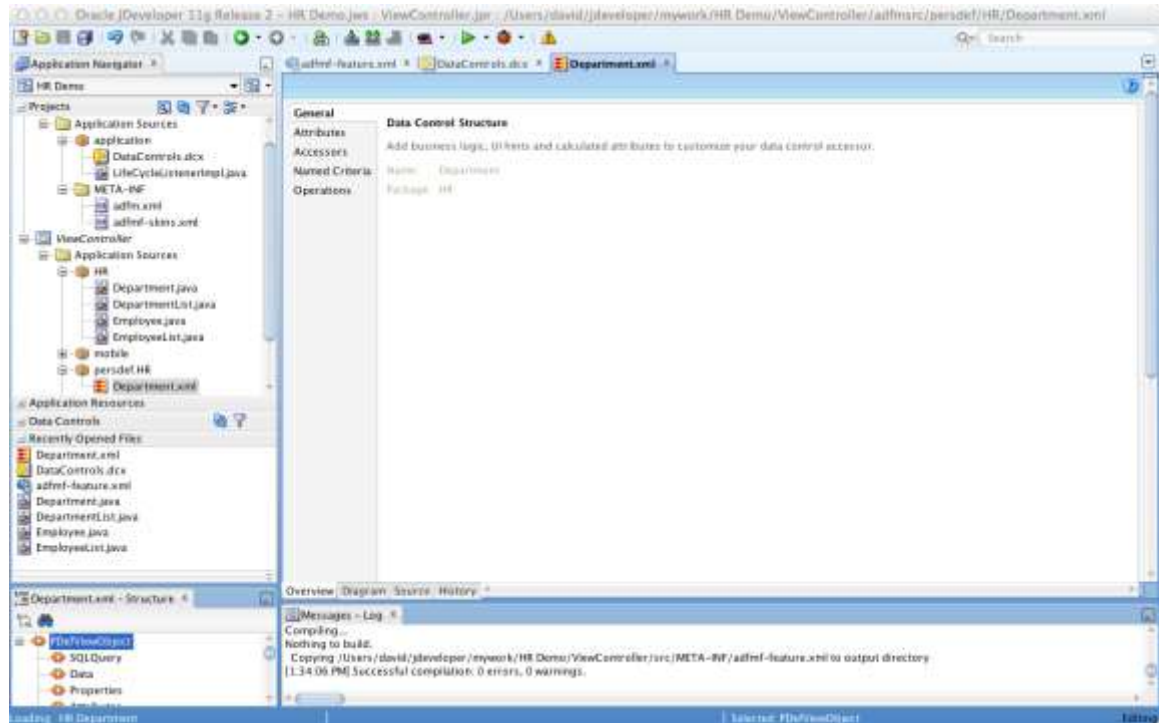


11. Expand DepartmentList and select departments.

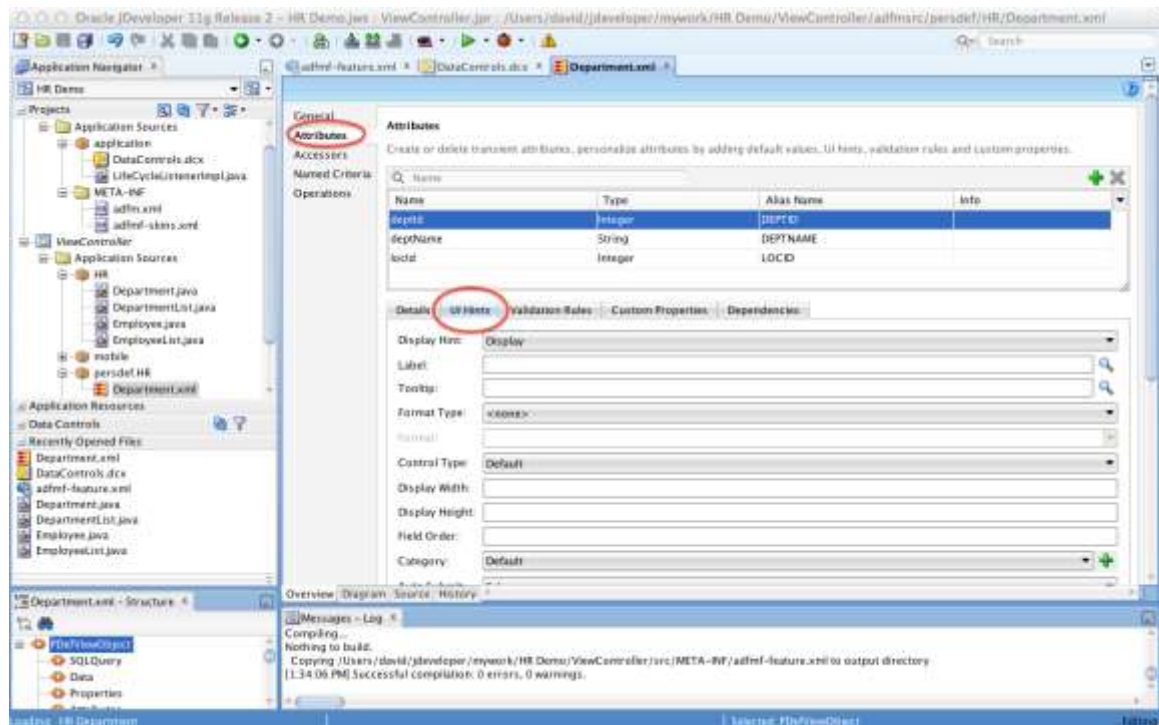


12. Click the **Edit Selected Element** icon (i.e. the pencil)

Department.xml opens.



13. Select **Attributes** and **UI Hints**.



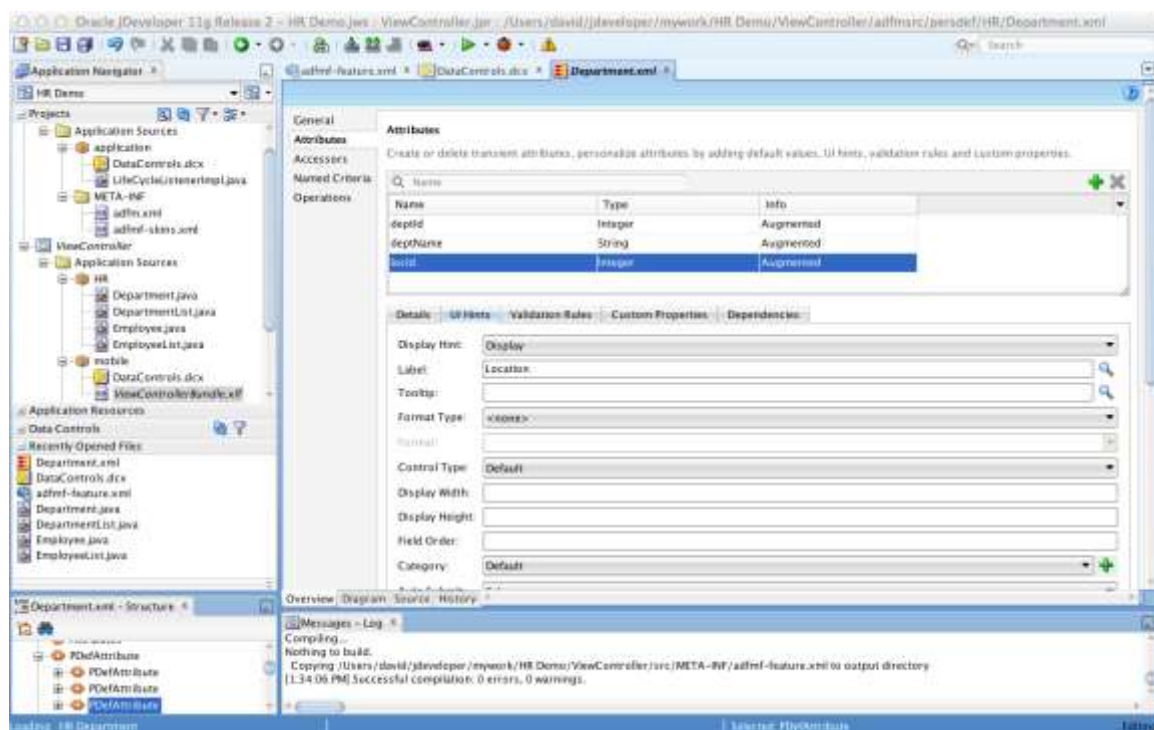
Next you're going to specify labels for each of these attributes. These will be used in the UI for your mobile application. You only have to set these once and they'll display wherever the attribute is used.

14. Assign **DeptId** a Label of **Dept ID** and type **return**.

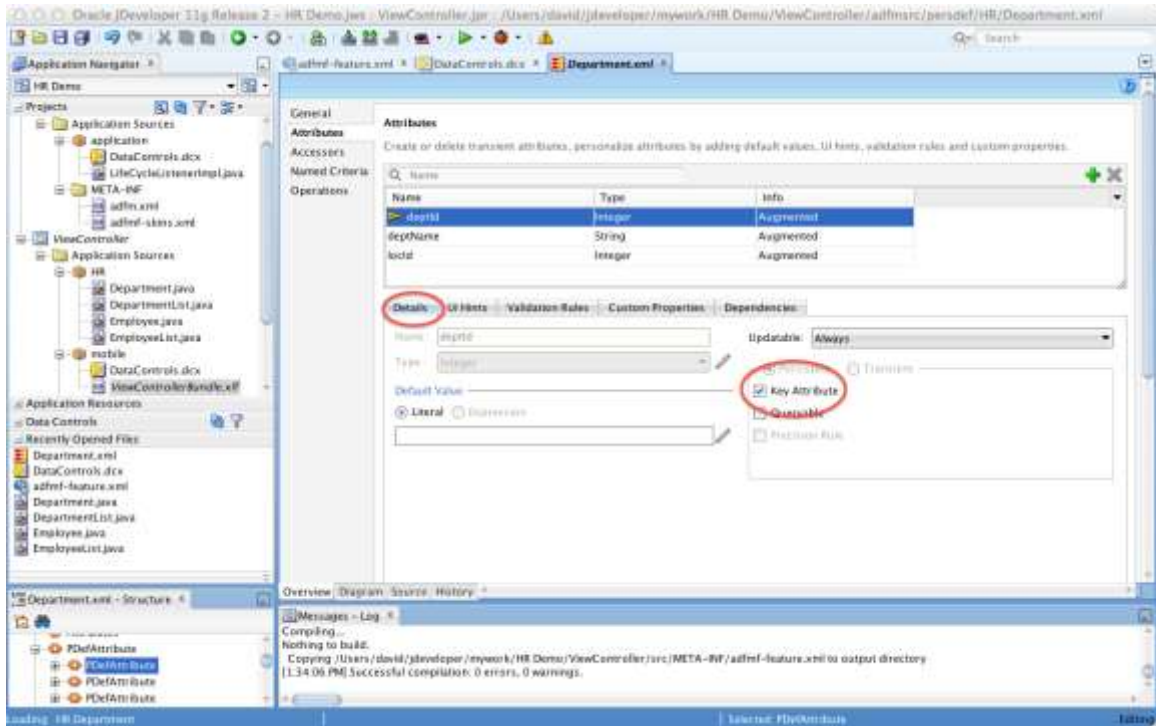
15. Assign **deptName** a Label of **Dept Name** and type **return**.

16. Assign **locId** a label of **Location** and type **return**.

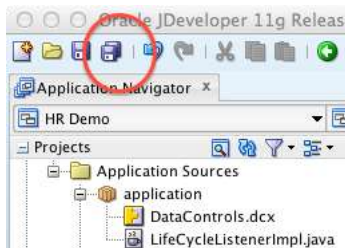
When you're done the info column should say "Augmented" for each of those attributes.



17. Select **deptId**, choose the **Details** tab and select **Key Attribute**.



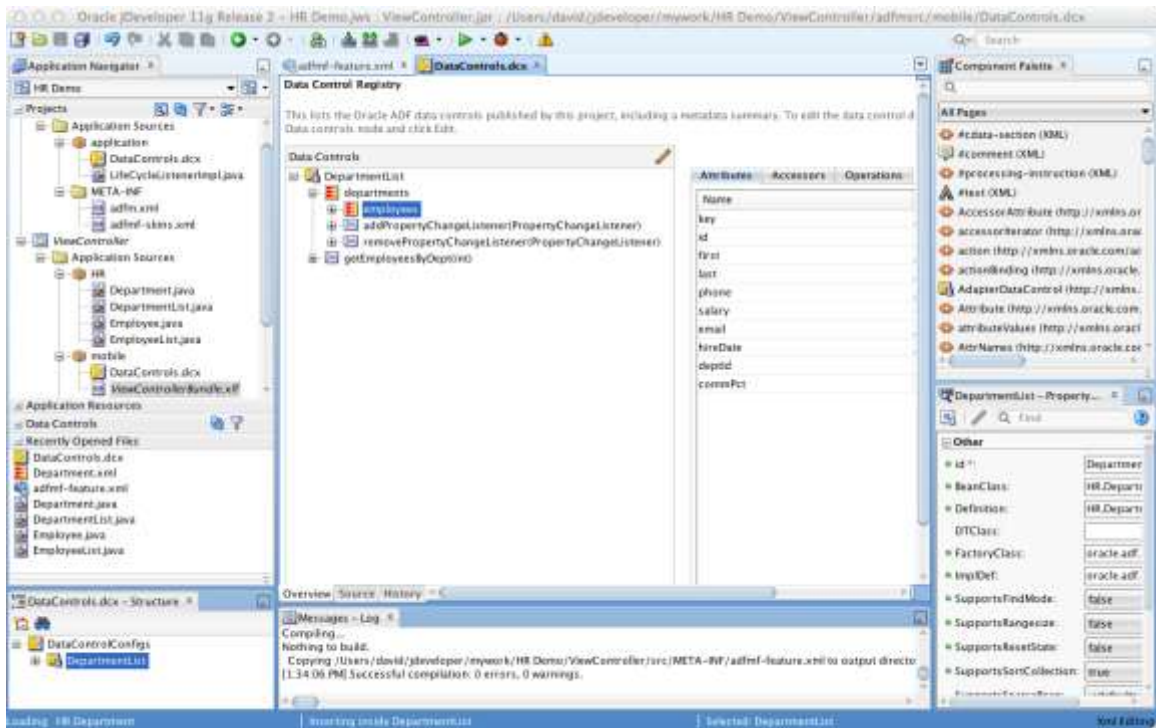
18. Save all files.



19. Close Department.xml

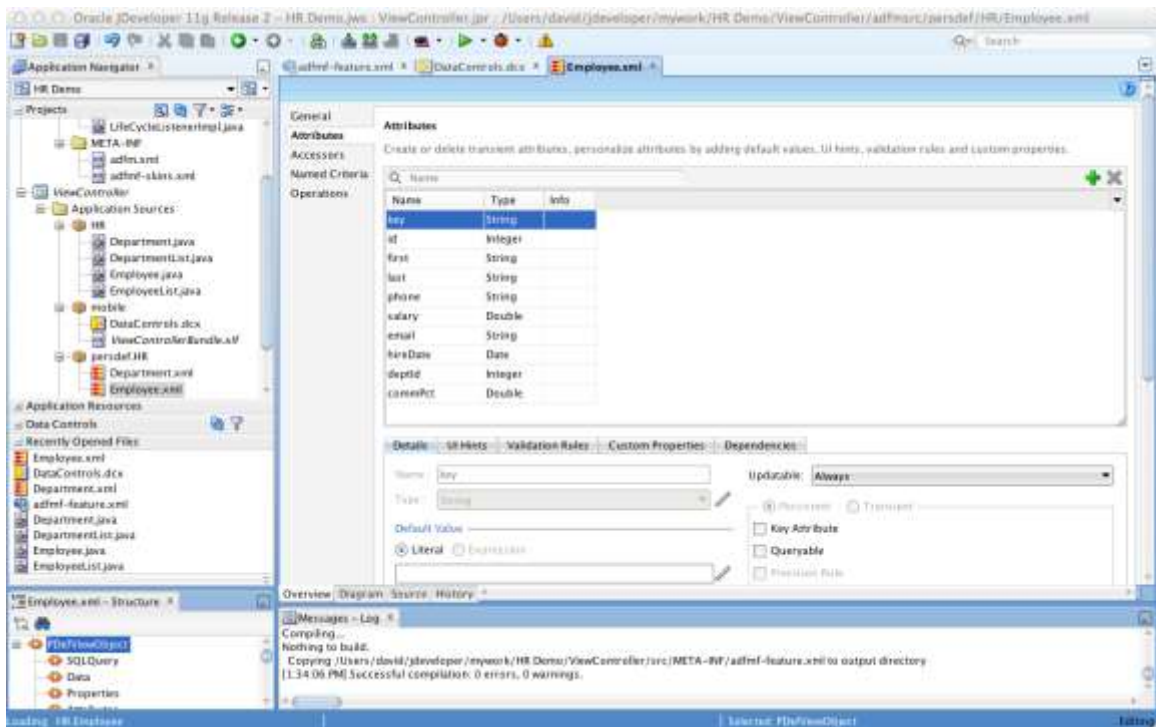
This should bring you back to the **Data Control Registry**.

20. Expand departments and select employees.



21. Click the **Edit Selected Element** icon (i.e. the pencil)

Employee.xml opens.



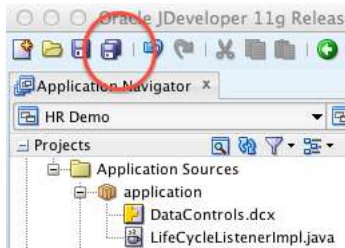
Next you're going to specify labels for a few attributes. This is similar to what you did for Department a few minutes ago.

22. Assign **id** a Label of **ID** and type **return**.

23. Assign **last** a label of **Last** and type **return**.

24. Assign **phone** a label of **Phone** and type **return**.

25. Save all files.



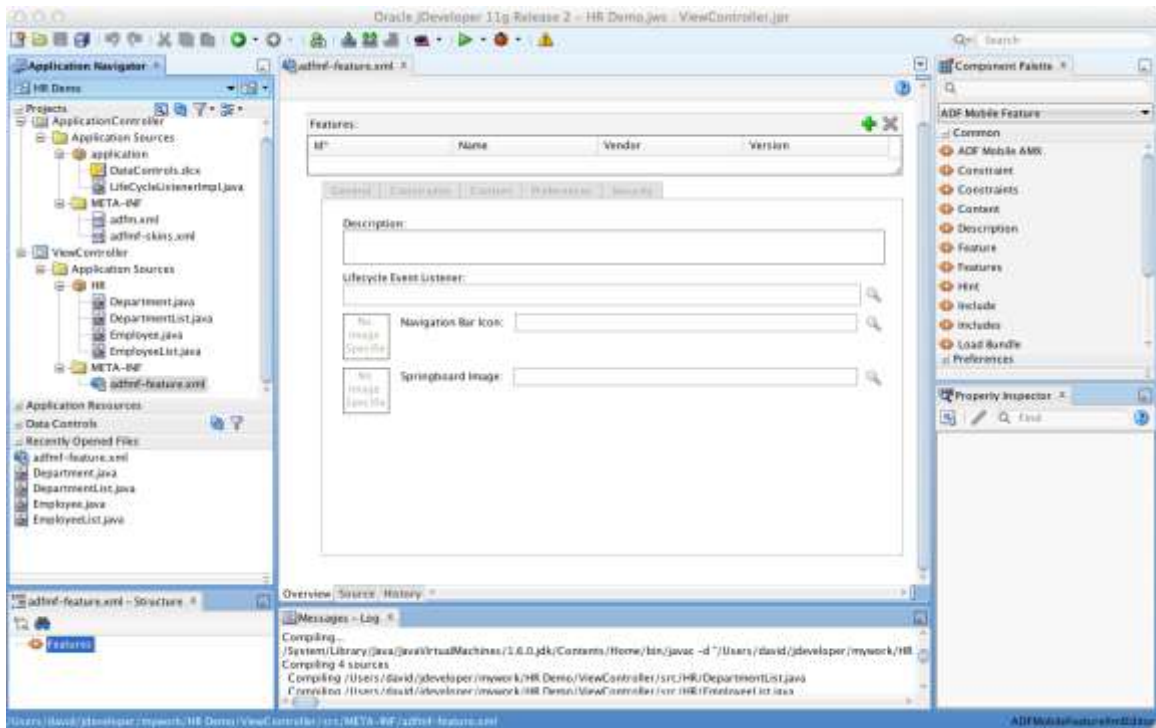
26. Close **employees.xml** and the **Data Control Registry**.

You've finished setting up the Data Control. Now it's time to build out the mobile app.

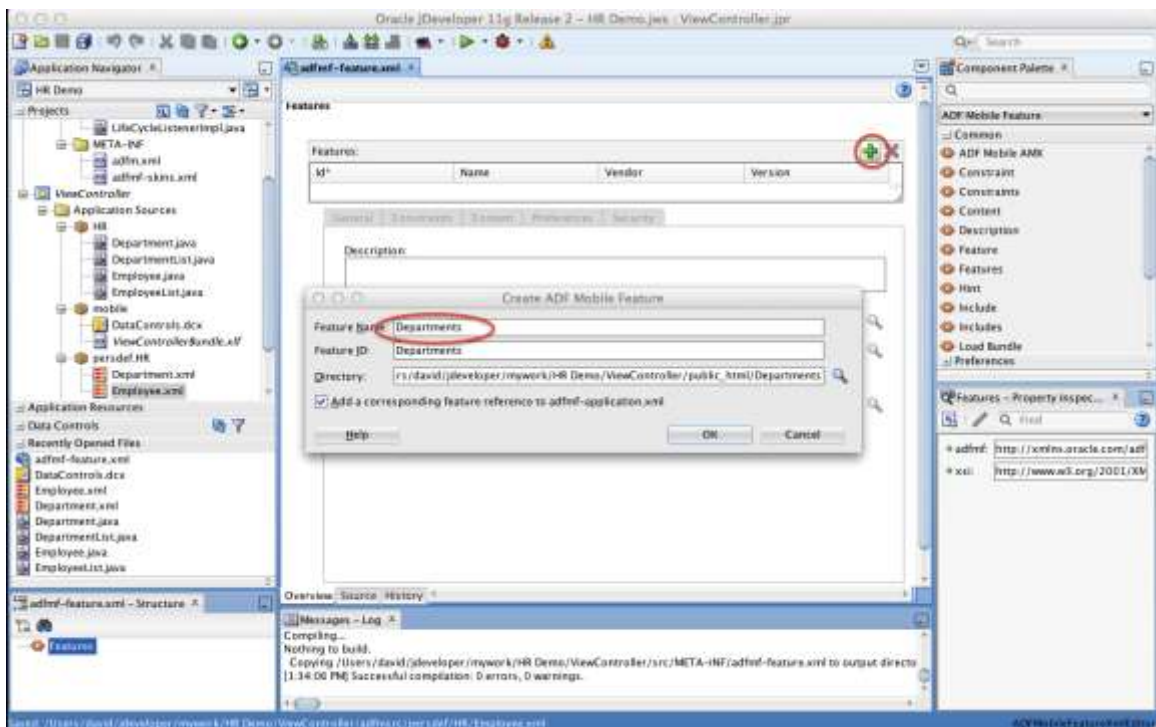
### Create ADF Mobile Feature and Task Flow

1. Return to adfmf-feature.xml.

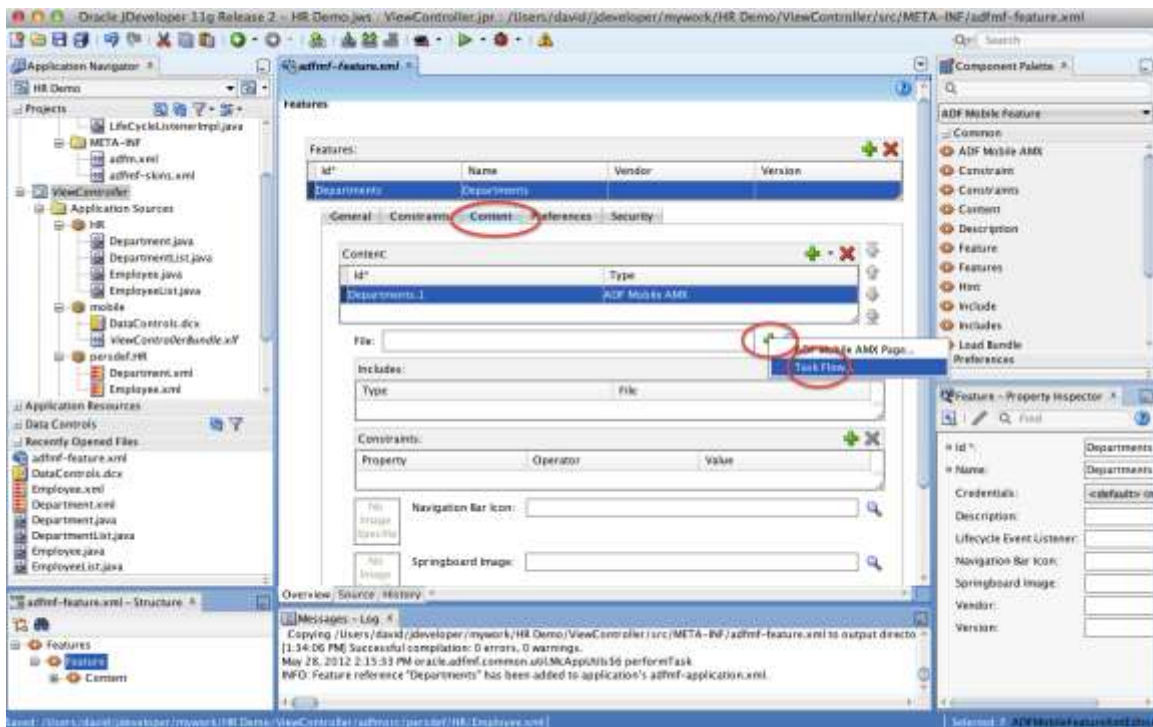
If you accidentally closed it, note it's selected in the left side of the following screenshot. Double clicking it will reopen the file.



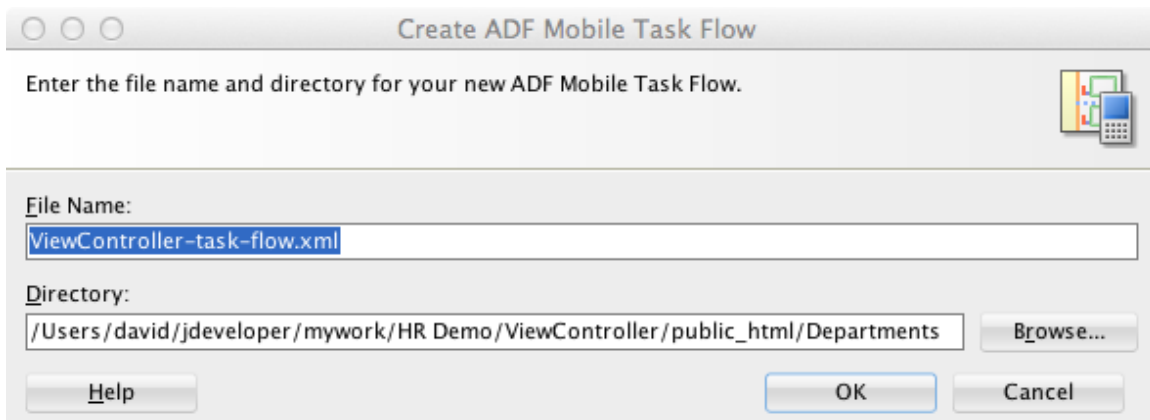
2. Click the plus icon to add a new feature and then name it “Departments” and click OK.



3. Choose the **Content** tab for the feature, then click the Plus icon to add a new **File** and choose type **Task Flow**.

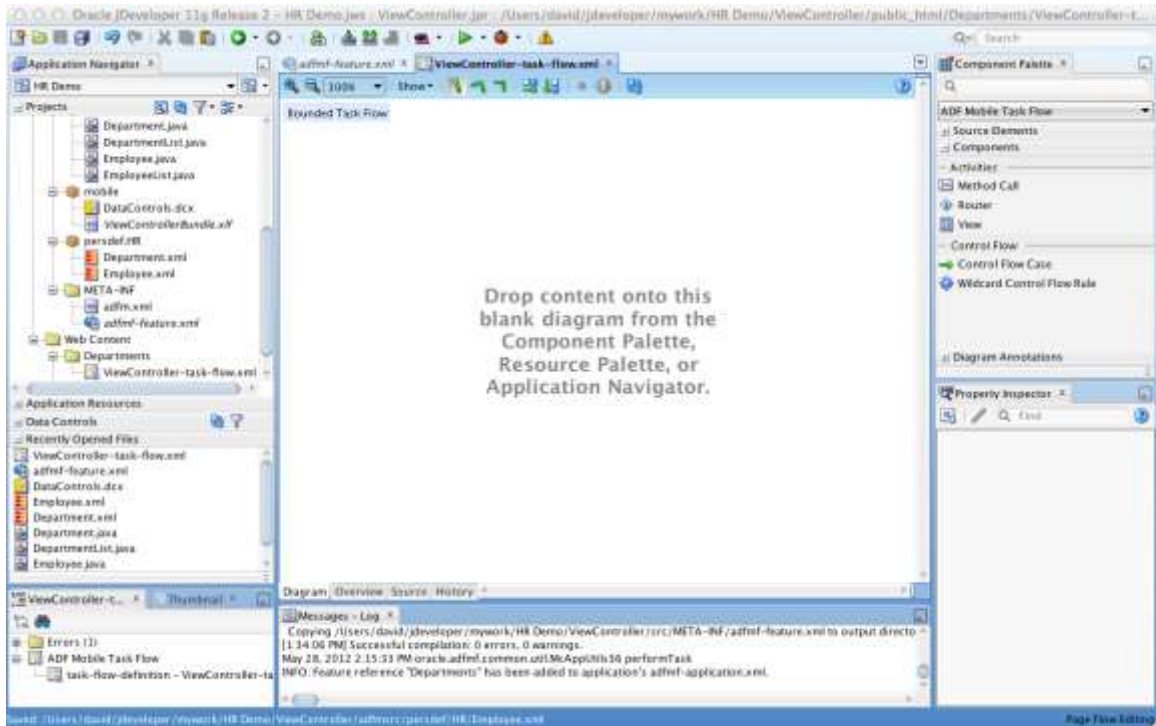


The **Create ADF Mobile Task Flow** dialog appears.

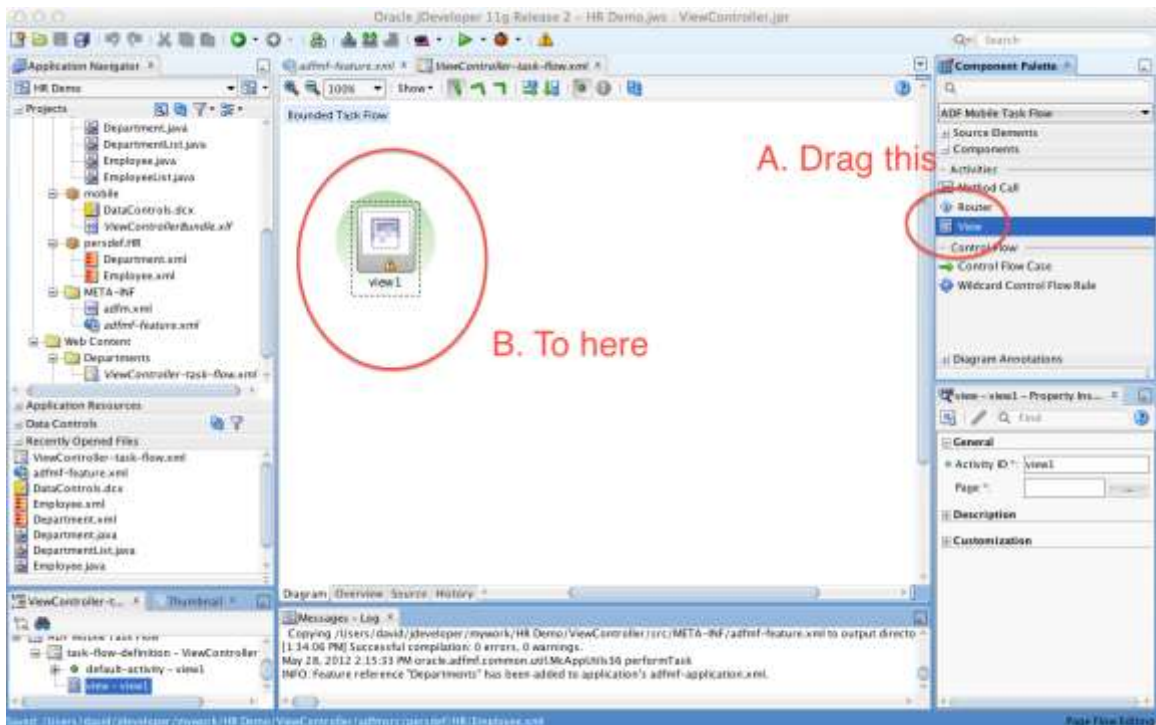


4. Click **OK** (the default name of ViewController-task-flow.xml is fine).

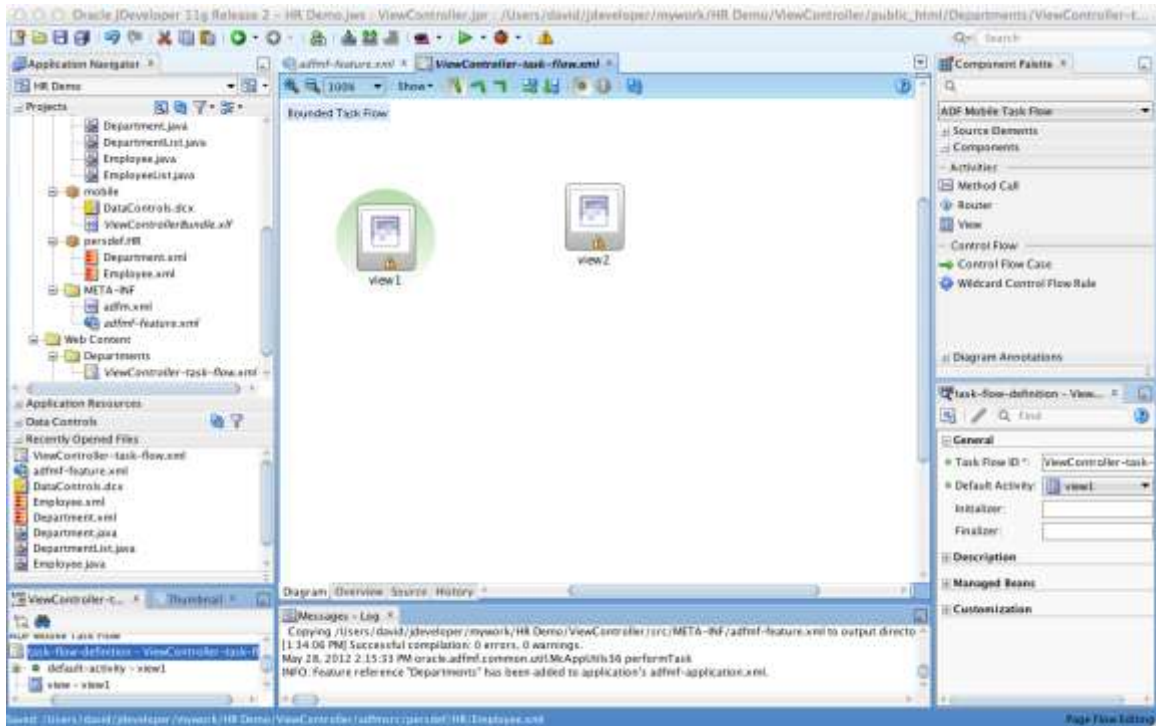
The **Task Flow** editor appears.



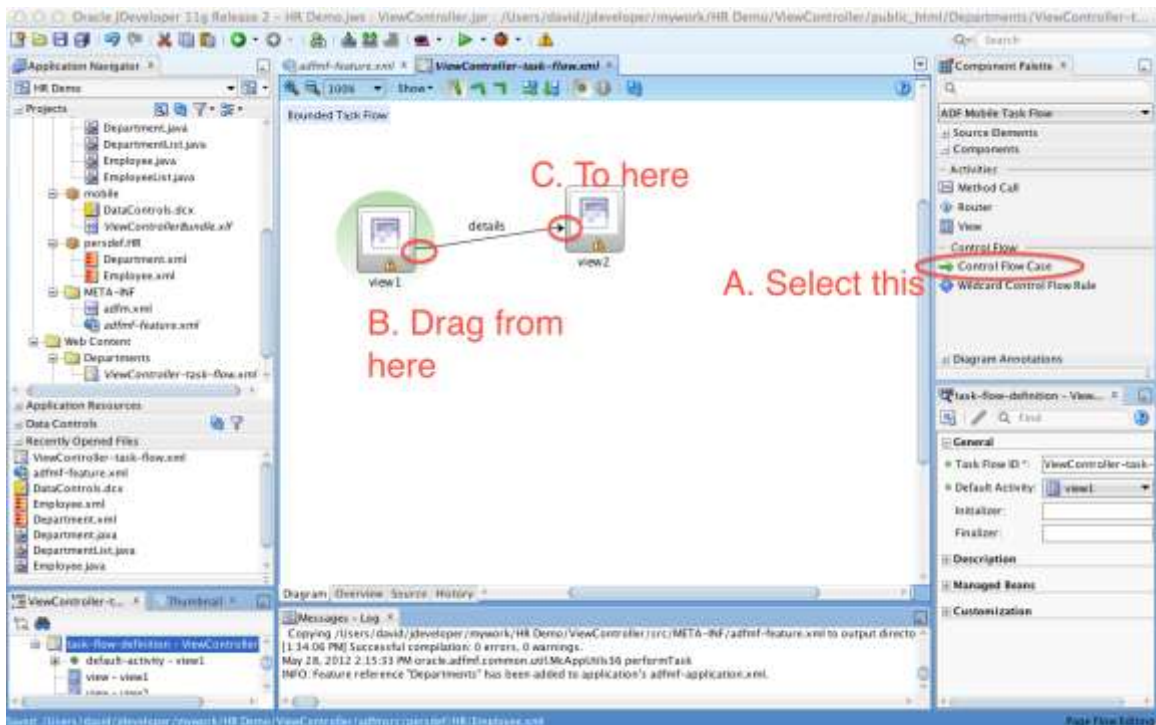
5. Drag a **View** onto the diagram from the **Component Palette**.



6. Add another **View**.

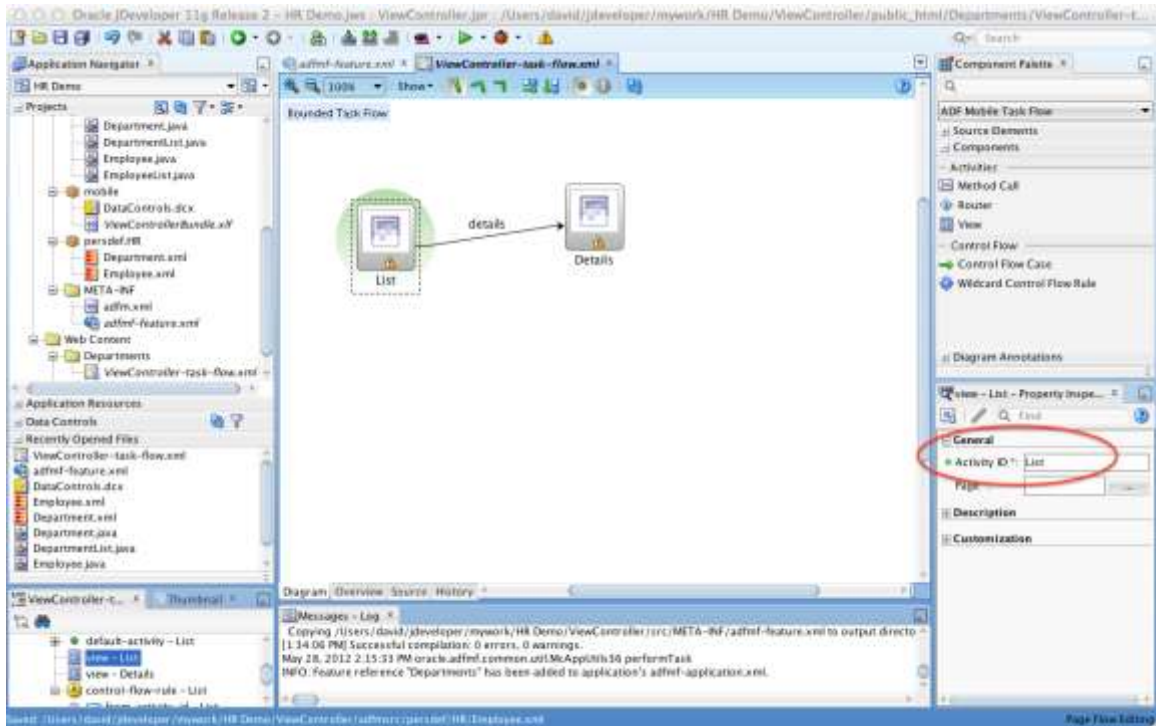


7. Select **Control Flow Case**, then drag from view1 to view2.



8. Name the Control Flow Case details.

9. Select view1 and set the **Activity ID** to List.

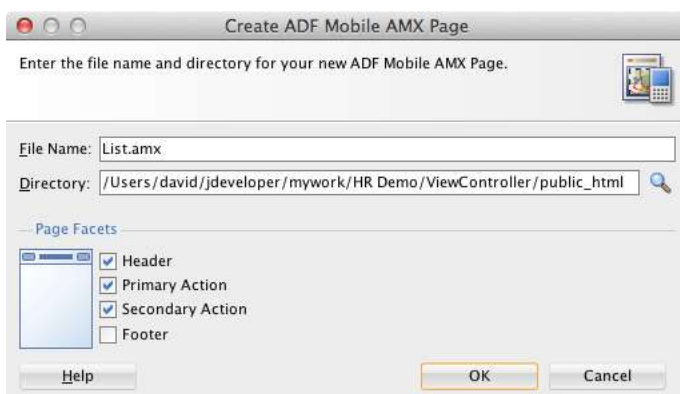


10. Similarly, select view2 and set the **Activity ID** to Details.

## Setup the List Page

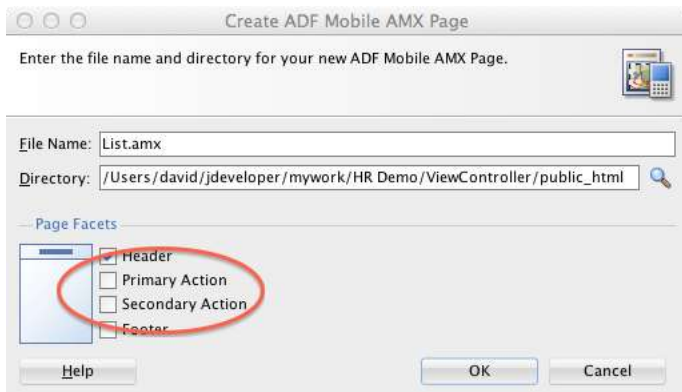
1. Double-click on List in the **Task Flow**.

The **Create ADF Mobile AMX Page** dialog displays.

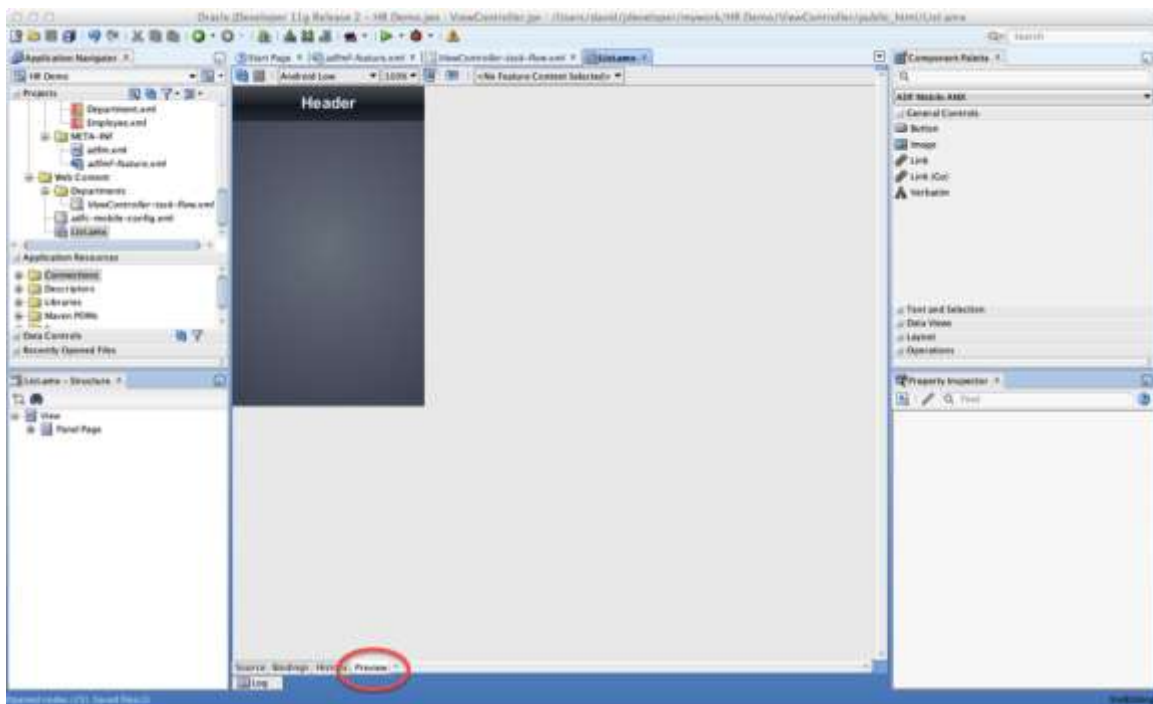


In a few steps, you'll add a list of departments to this page. In the meantime, you don't need navigation buttons on this page, so you're going to remove those.

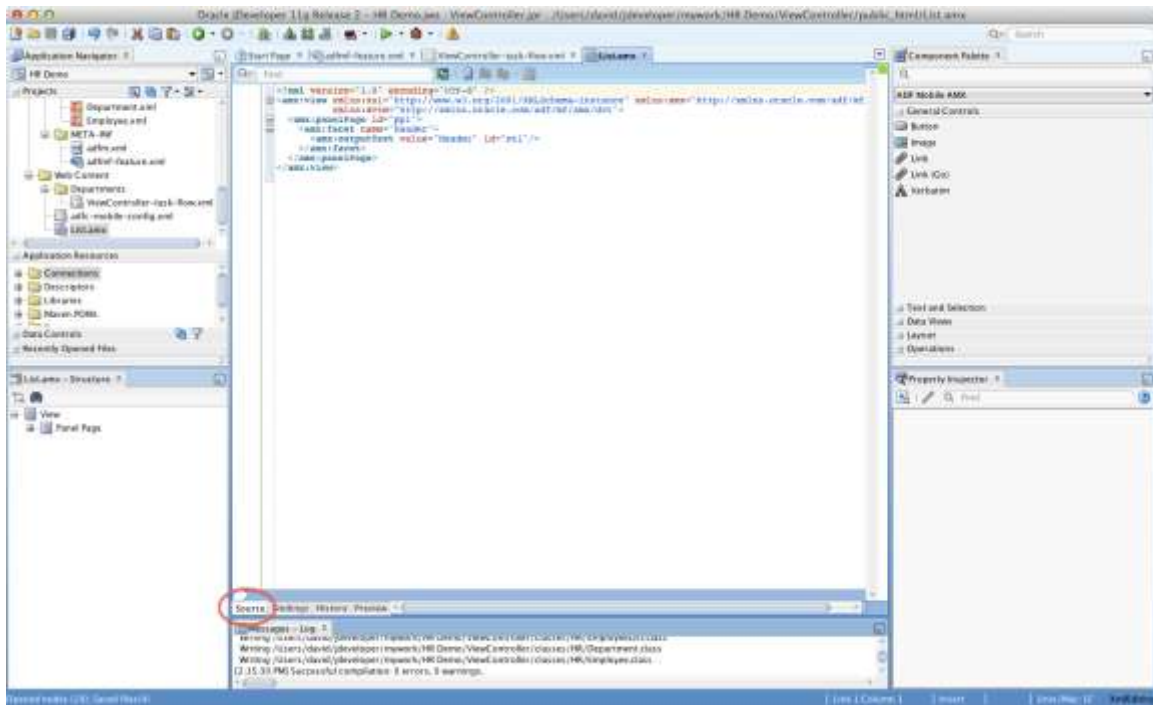
2. Deselect **Primary Action** and **Secondary Action**. Leave **Header** as-is.



3. Click **OK**.
4. List.amx opens in the **Source** view. Click the **Preview** tab and observe, it's a rather empty page. You're going to change that.

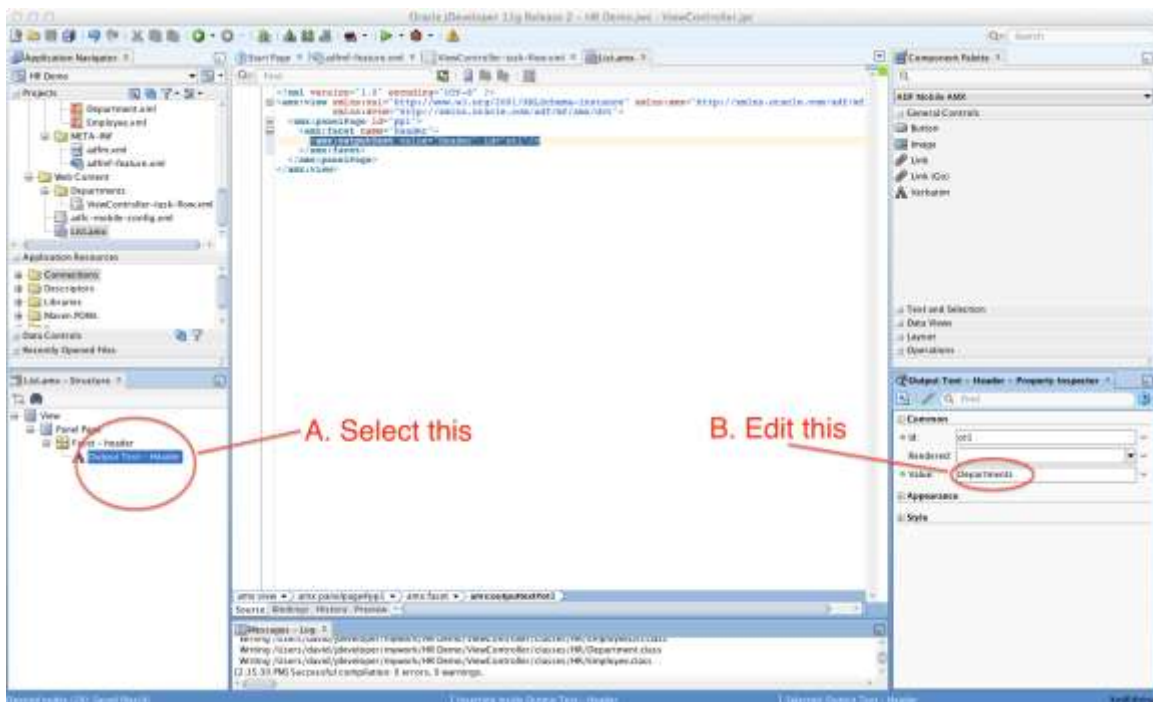


5. Click the **Source** view tab to return to the view where you'll edit the page.



6. In the **Structure window** in the bottom left expand **Panel Page - Facets - Header -> Output Text - Header**.

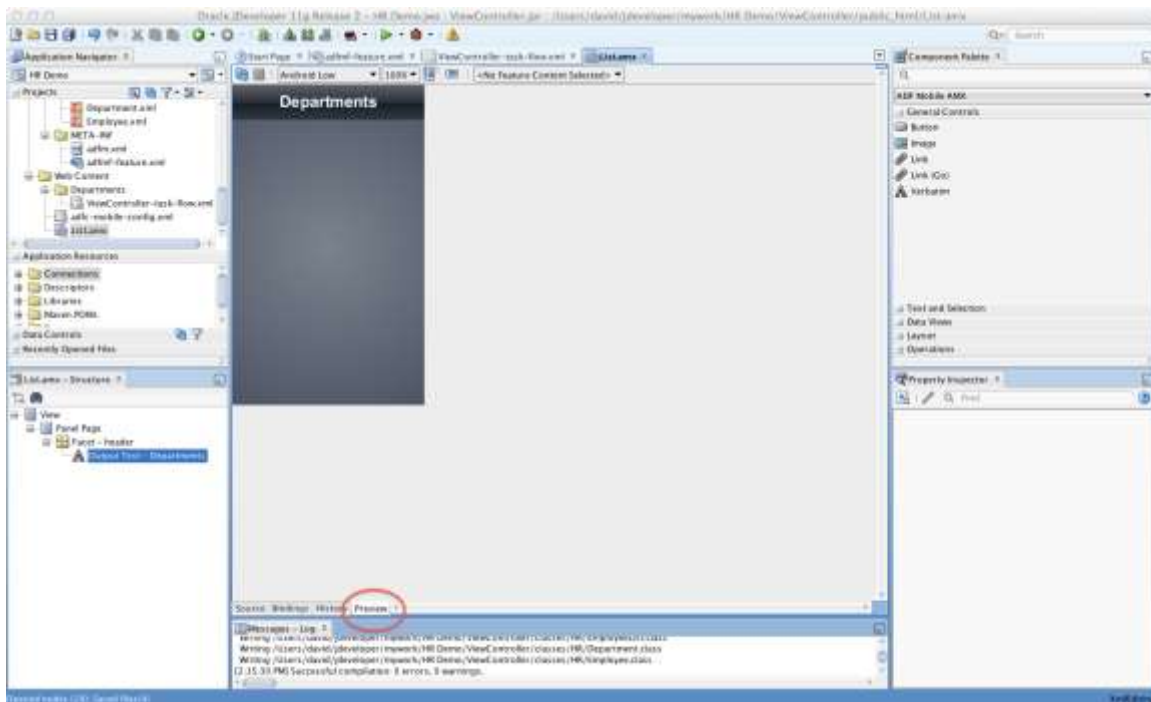
If the **Structure Window** isn't open you can select it from the **View** menu.



7. Use the **Property Inspector** to edit the **Label** for the **Output Text** from so it says “Departments” rather than “Header” (as shown in the screenshot above).

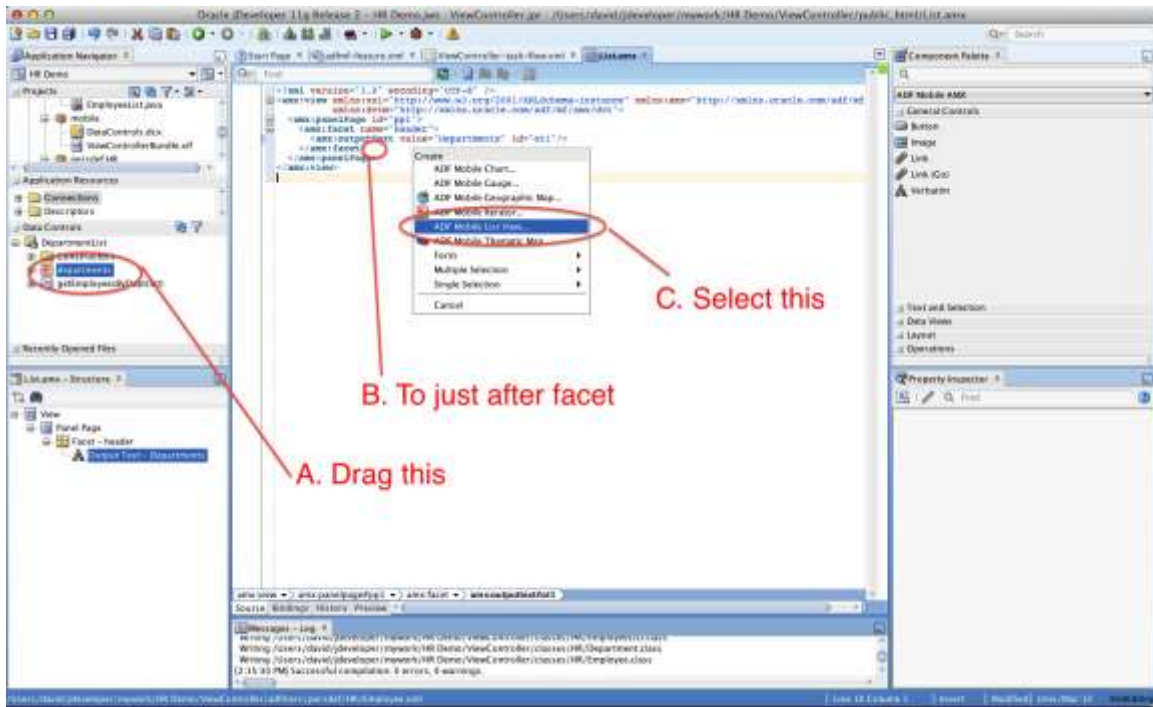
If the **Property Inspector** isn't open you can select it from the **View** menu.

8. Navigate back forth to the **Preview** tab at the bottom of the main editor window you'll see how the header will display.



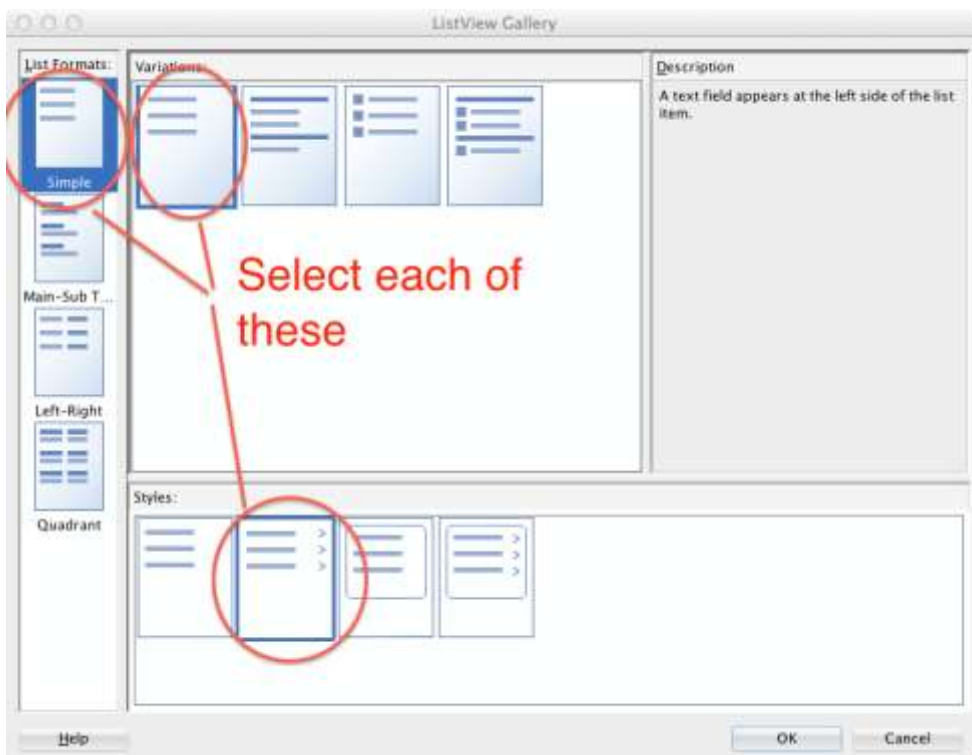
9. Expand the **Data Control** list within the Application Navigator window on the left. If nothing is showing hit the refresh button. Then expand the DepartmentList object and drag the **departments** object just after the closing `</amx:facet>` tag in the panel page and select **ADF Mobile List View**.

(A screenshot that spells out these instructions more clearly appears on the next pag.)



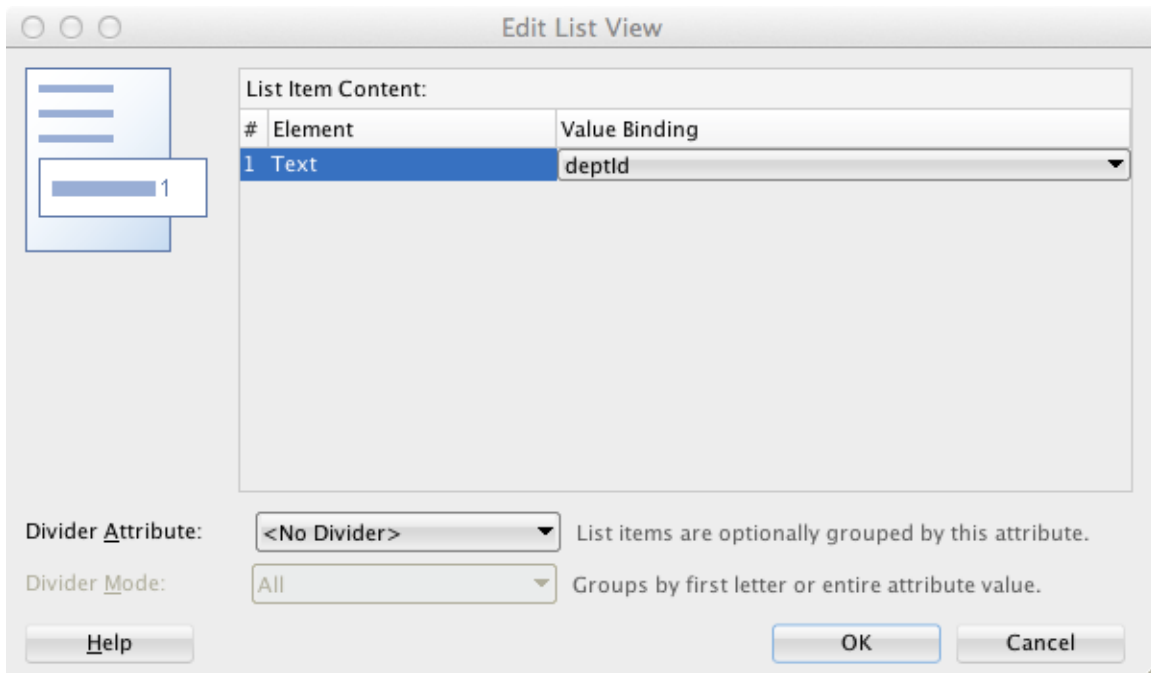
The **ListView Gallery** dialog appears.

10. Select **Simple List Format** and choose the style as indicated in the following screenshot.

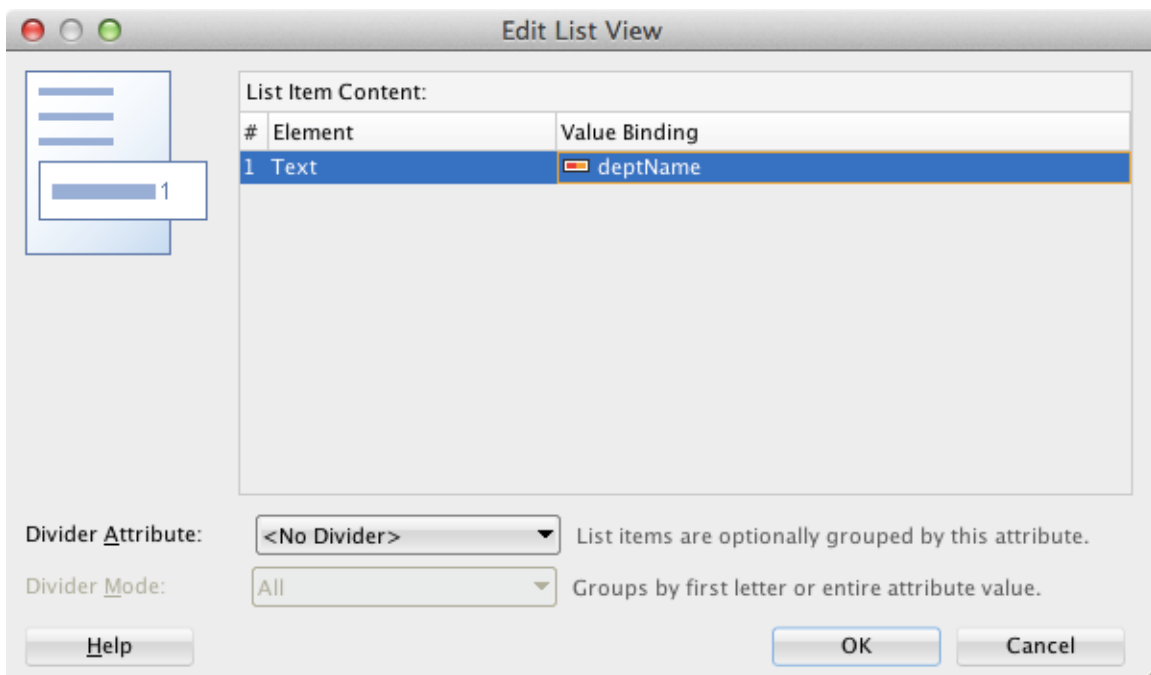


11. Click OK.

The **Edit List View** dialog appears.

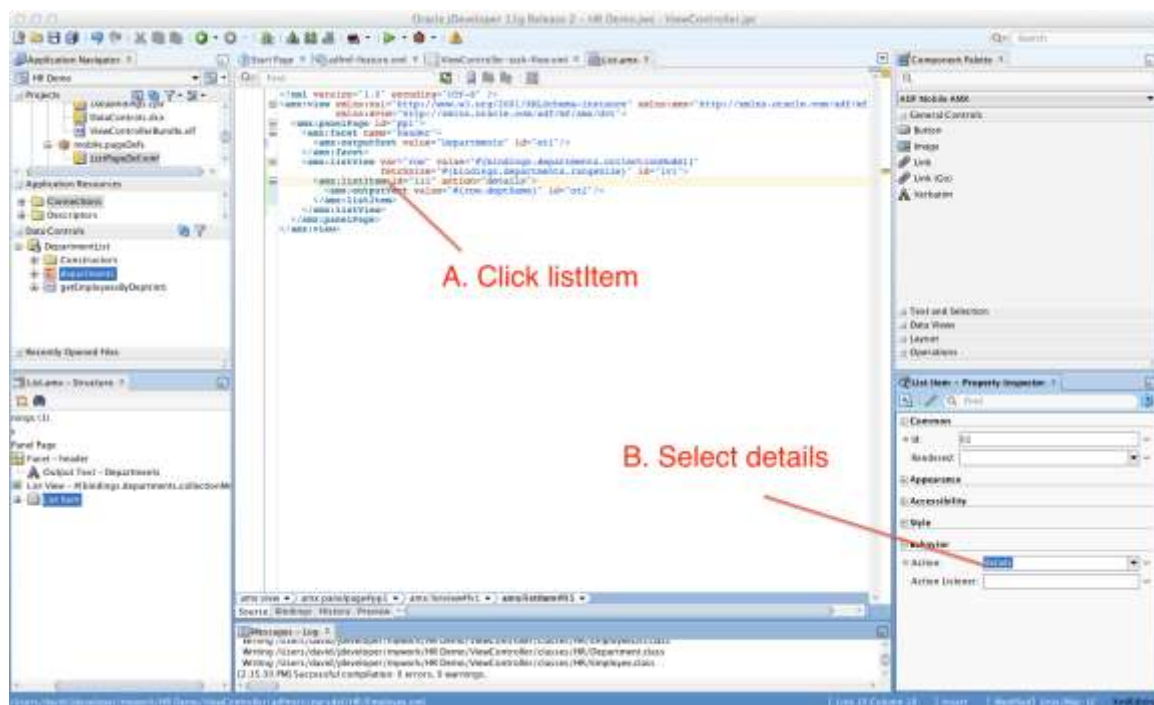


12. Change **Value Binding** to deptName.

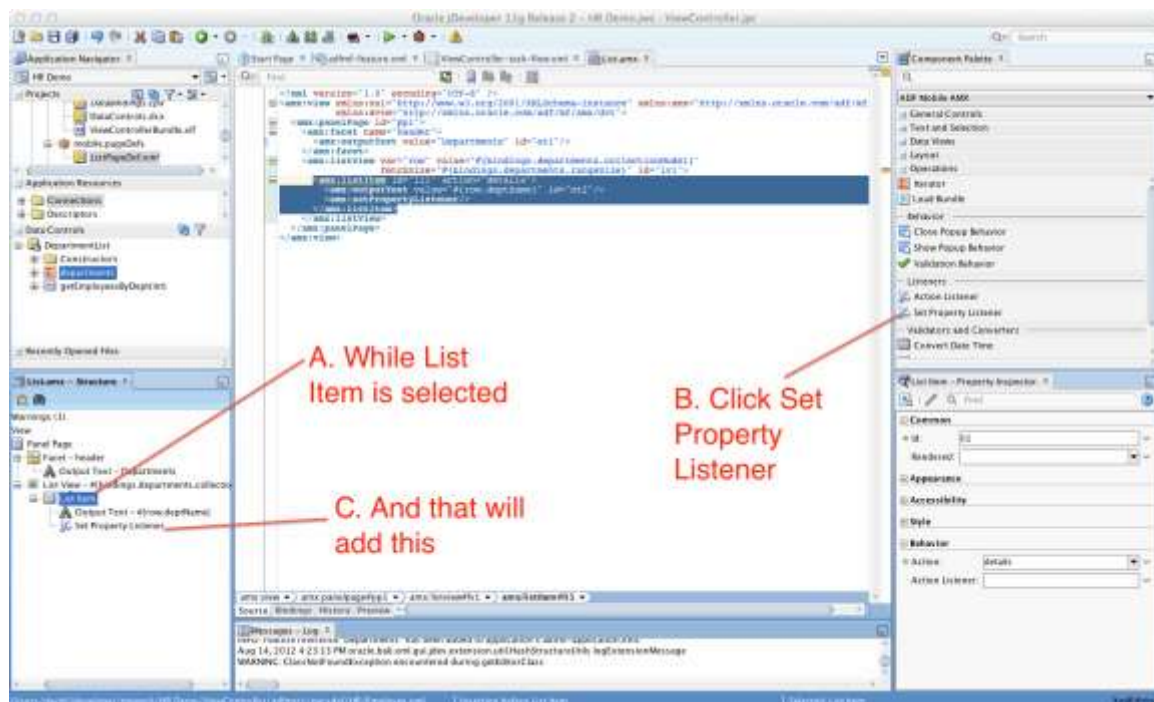


13. Click **OK**.

14. Click inside the `listItem` line. Then change the **Action** to details.



15. Select the **List Item** in the Structure window, then click **Set Property Listener** from **Operations** group of **Component Palette**.

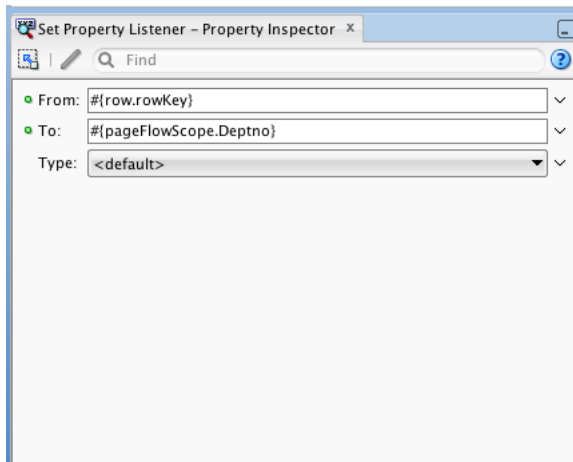


16. In the **Property Inspector**, set the Property Listener's **From** field to `#{row.rowKey}`. You could also use the Expression Builder to do this (Under ADF Mobile Objects -> Row).

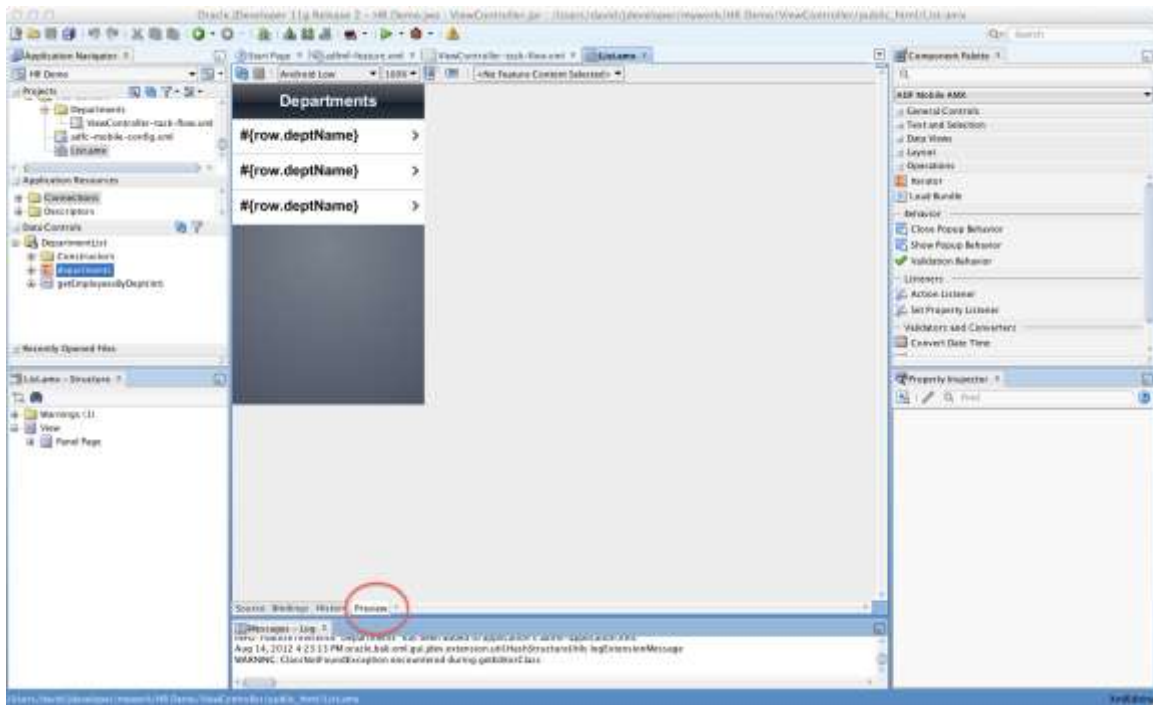
17. Set the Property Listener's **To** field to `#{pageFlowScope.Deptno}`

For the **To** field, you won't find this value in the Expression Builder. Take care typing this – you're going to use it later and of course it must match exactly.

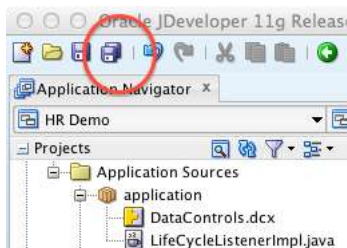
You're putting the value of the rowkey into a field in page flow scope named Deptno.



18. Navigate back forth to the **Preview** tab at the bottom of the main editor window you'll see how the list will display.



19. Save all files.



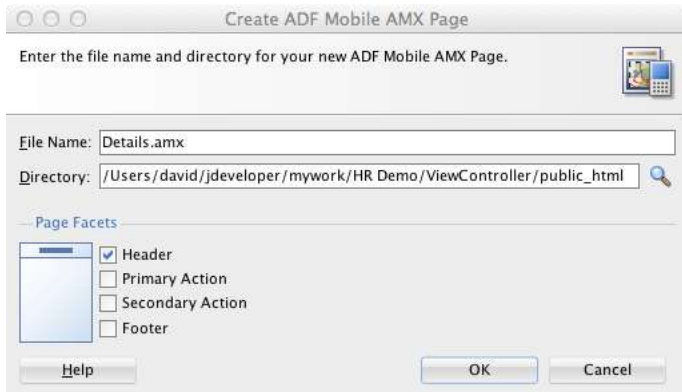
20. Close List.amx.

That should bring you back to the **Task Flow** (ViewController-task-flow.xml).

## Setup Details Page

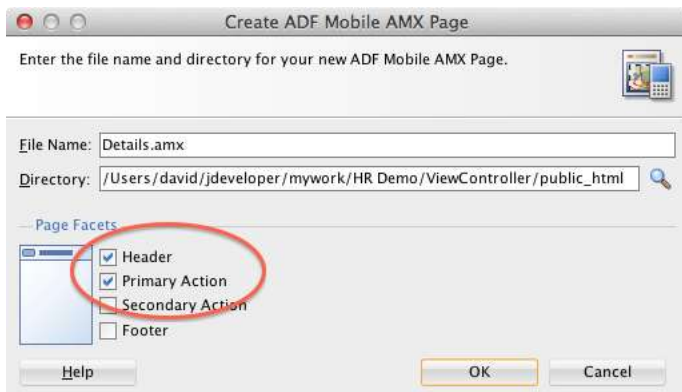
1. Double-click on Details view.

The **Create ADF Mobile AMX Page** dialog displays.



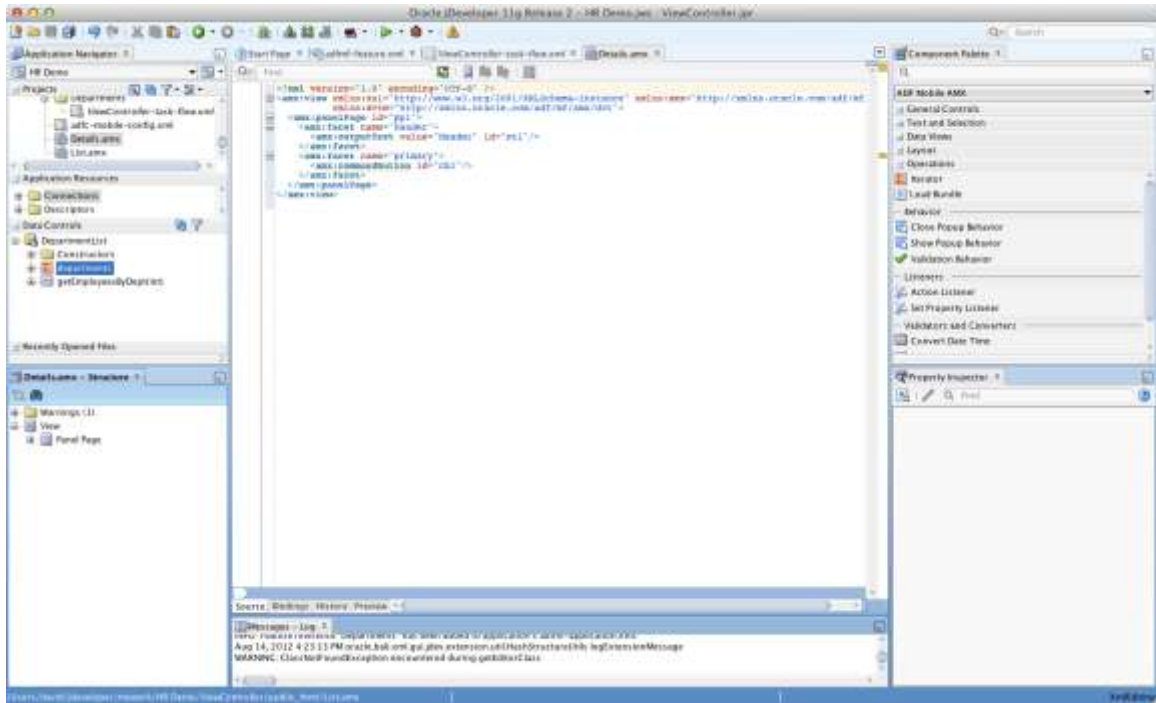
This time you'll want to add a navigation button.

2. Select **Primary Action** and leave **Header** as-is.

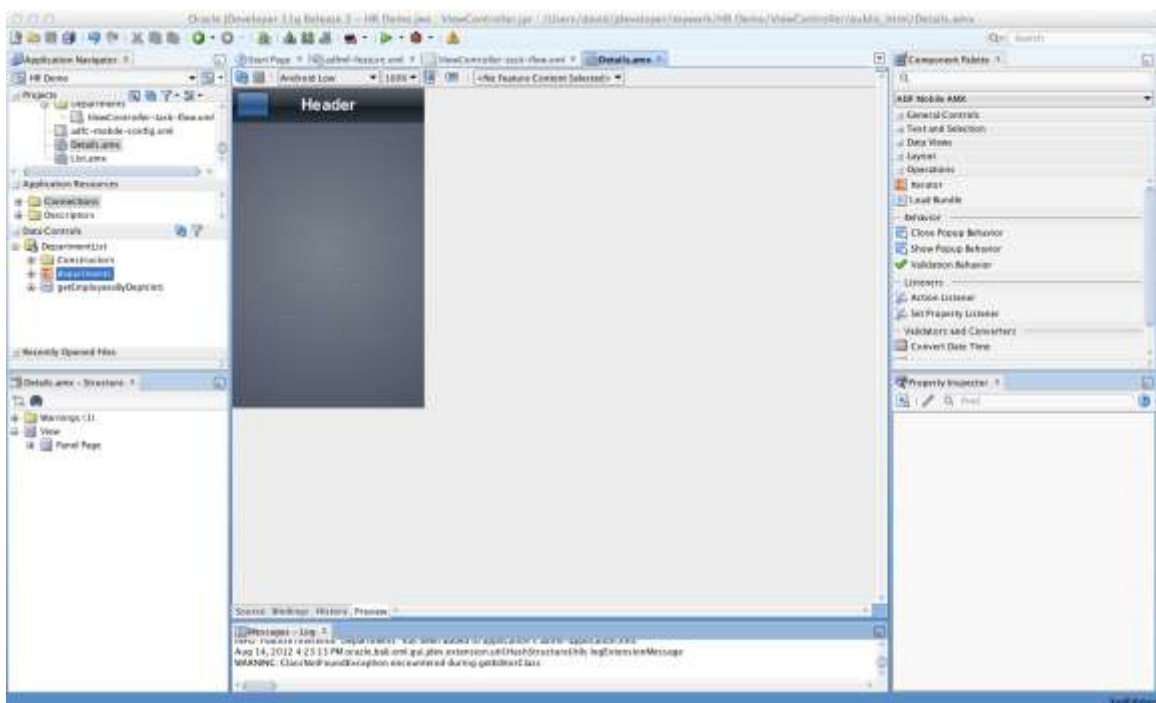


3. Click **OK**.

Details.amx opens in the **Source** view.



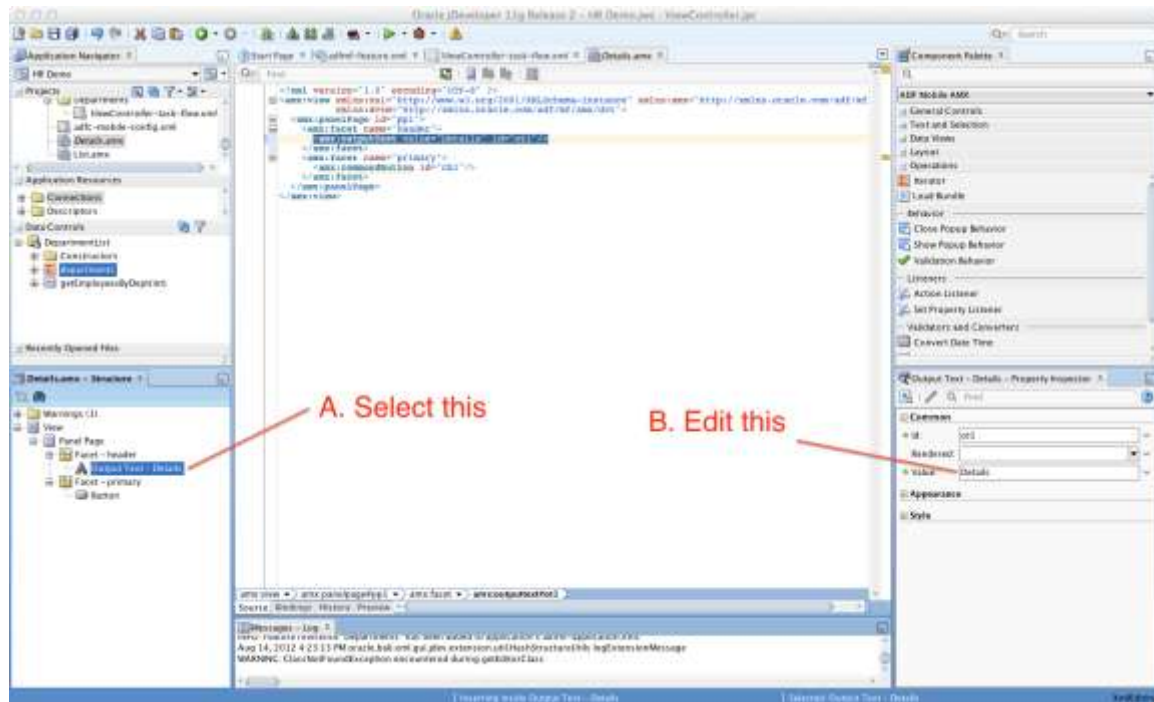
20. Navigate back forth to the **Preview** tab at the bottom of the main editor window you'll see how the list will display.



3. Click the **Source** view tab.

4. In the **Structure window** in the bottom left expand **Panel Page - Facets - Header -> Output Text - Header**.

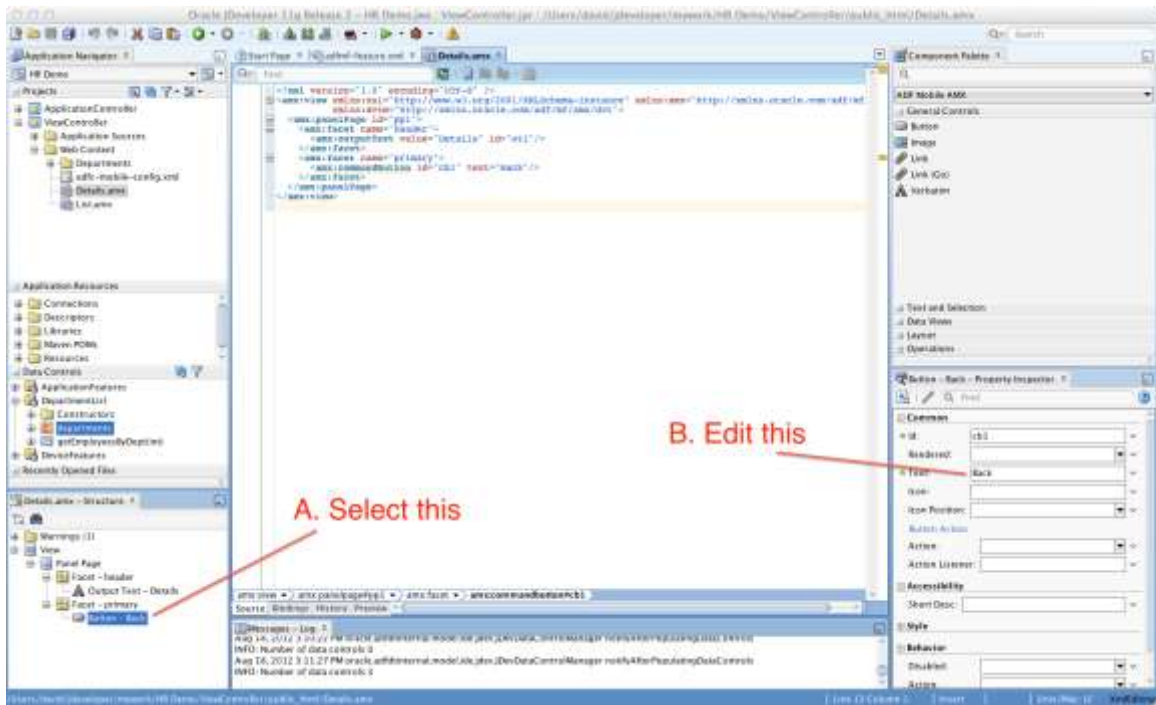
If the **Structure Window** isn't open you can select it from the **View** menu.



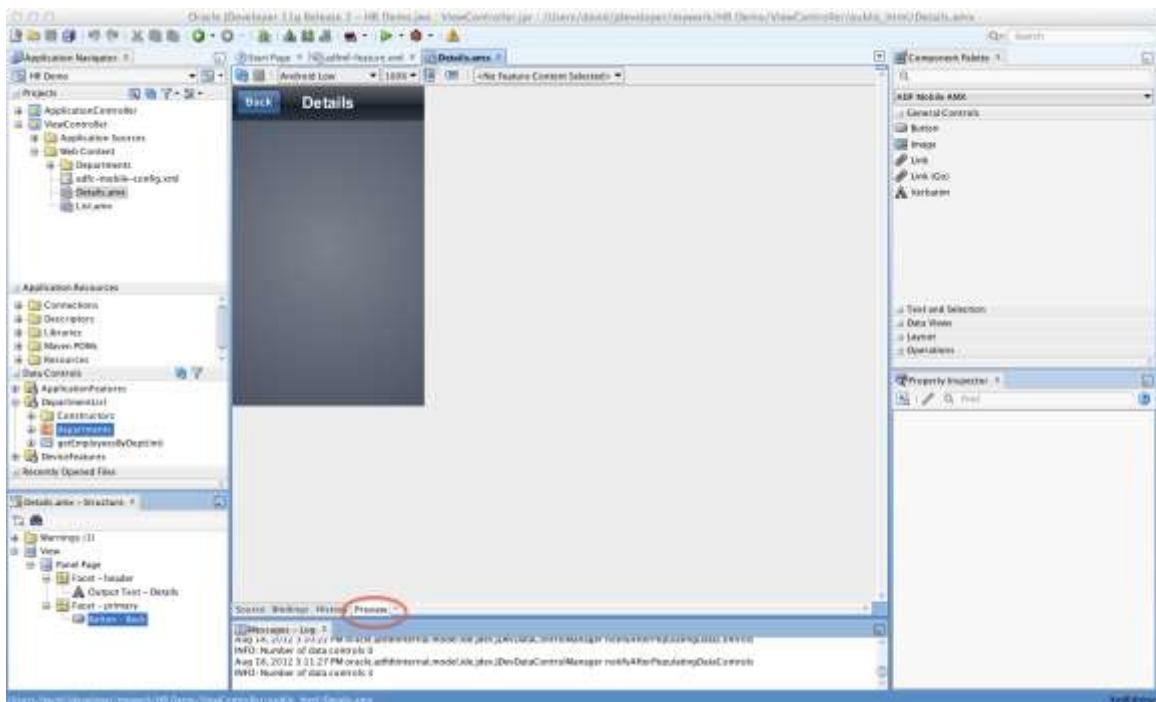
5. Use the **Property Inspector** to edit the **Value** for the **Output Text** so it says "Details" rather than "Header".

(If the **Property Inspector** isn't open you can select it from the **View** menu.)

6. In the **Structure window** in the bottom left expand **Panel Page - Facets - Primary -> Button**.

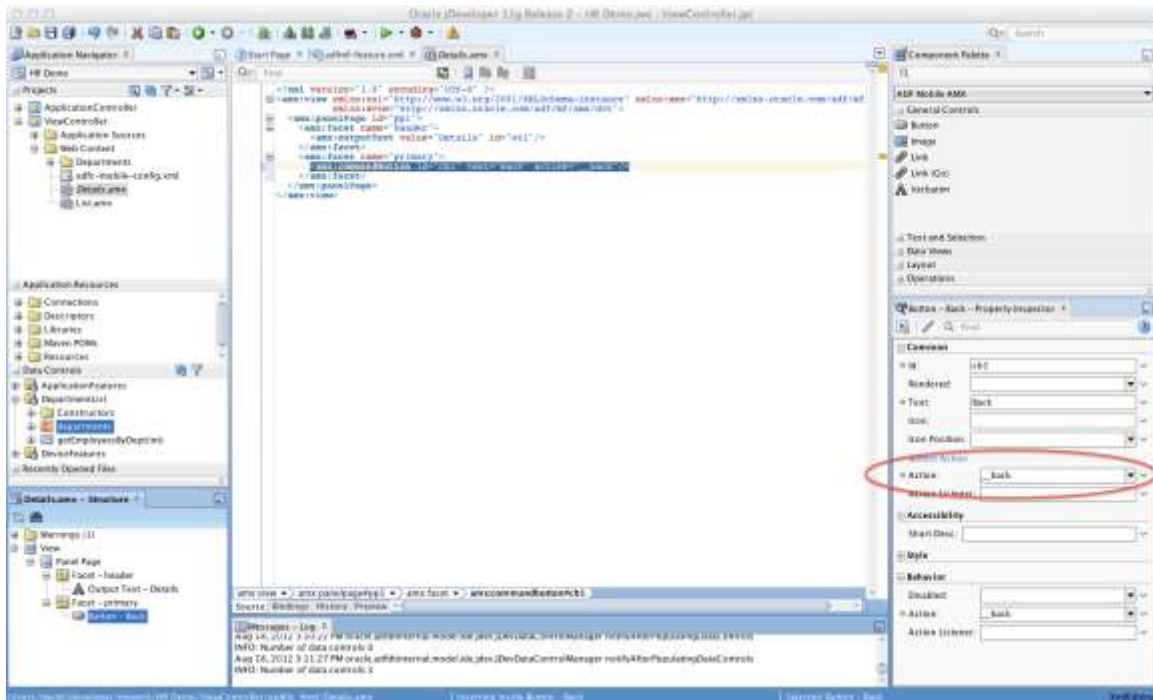


7. Use the **Property Inspector** to edit the **Value** for the **Button** so it says “Back”.
8. Navigate back forth to the **Preview** tab at the bottom of the main editor window you'll see how the header will display.



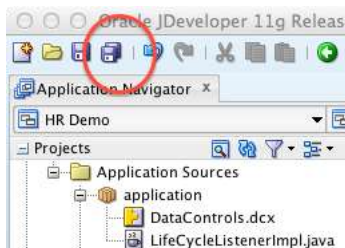
11. In the **Property Inspector**, set the Button's **Text** attribute to Back and its **Action** to `_back`.

Note, `_back` begins with two underscore characters. Either type that carefully, or use the drop-down and select it.

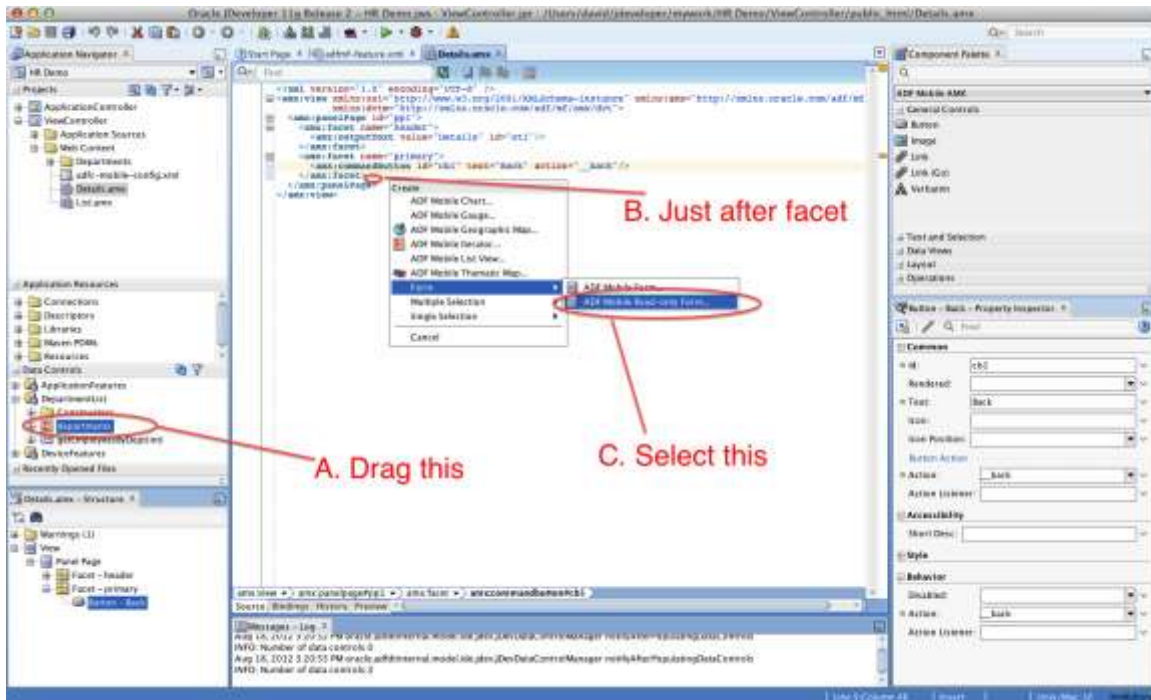


Setting the action to `_back` will enable the page to navigate to the page in the task flow that invoked it.

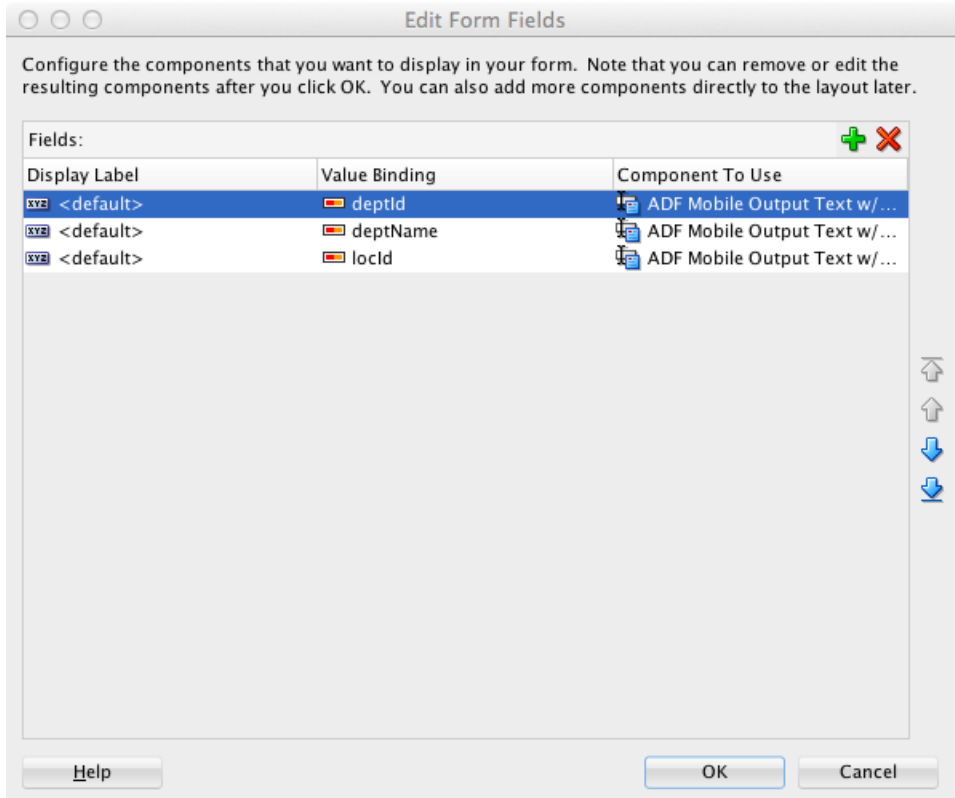
12. Save all files.



13. From the Data Controls palette in the Application Navigator, expand DepartmentList, then drag departments just after the closing `</amx:facet>` tag for the primary facet in the panel page and select **Form -> ADF Mobile Read-only Form**.

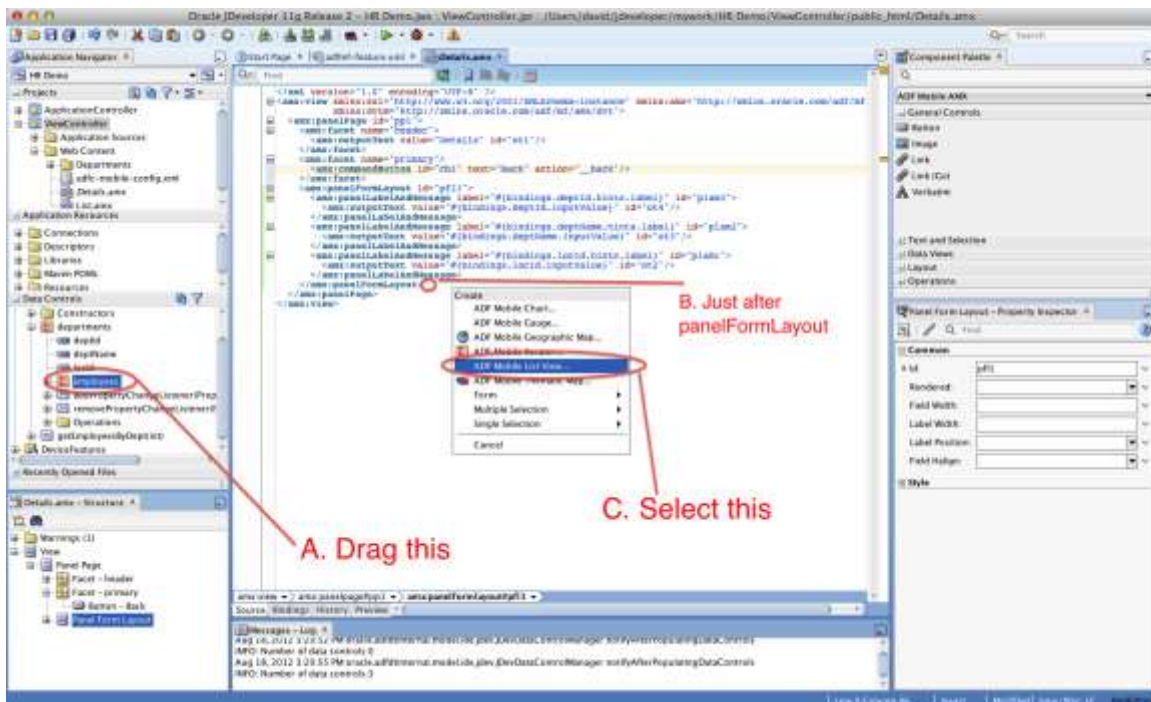


The **Edit Form Fields** dialog displays.



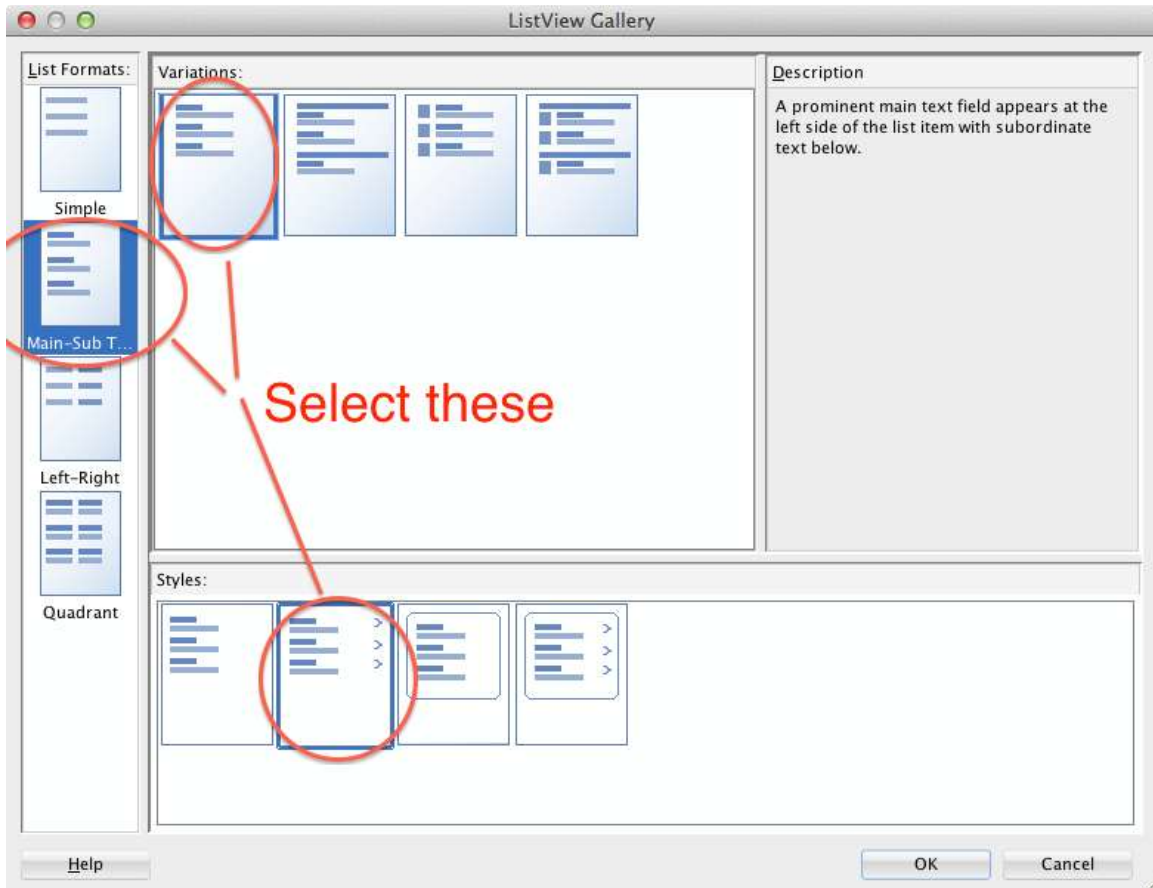
14. Click OK (i.e. accepting defaults).

15. Again, from the Data Controls palette in the Application Navigator, expand Department List, expand **employees**, then drag the employees just after the closing `<amx:panelFormLayout>` tag in the panel page and select **ADF Mobile List View**.



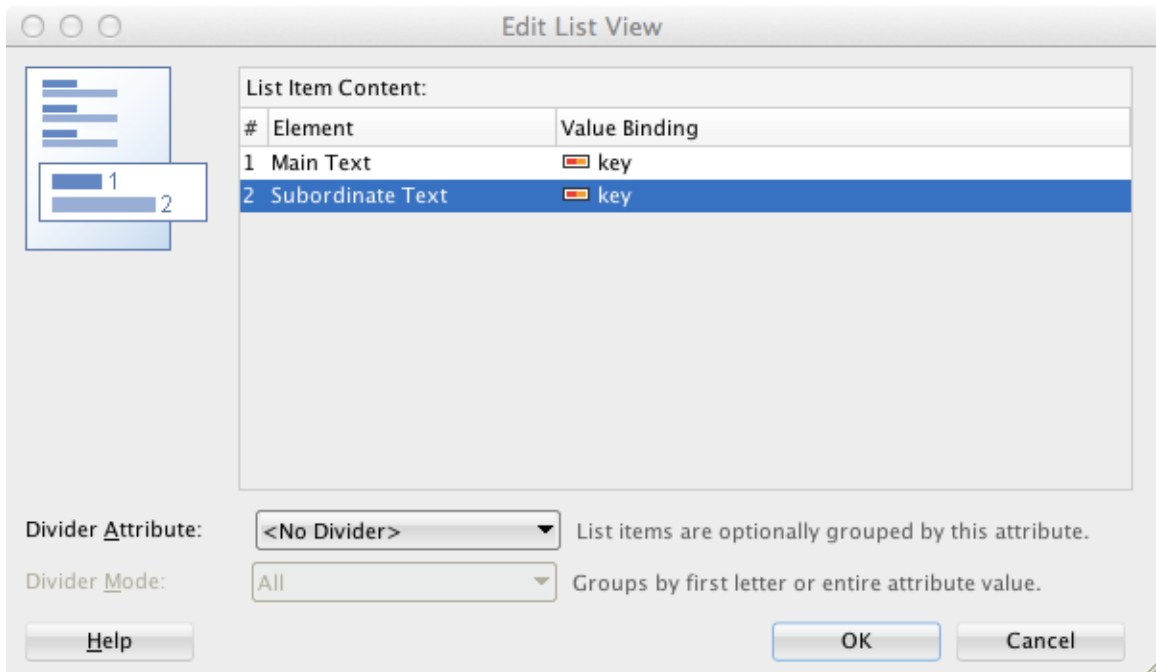
The **ListView Gallery** dialog appears.

21. Select **Main-Sub Text Format** and choose the style as indicated in the following screenshot.

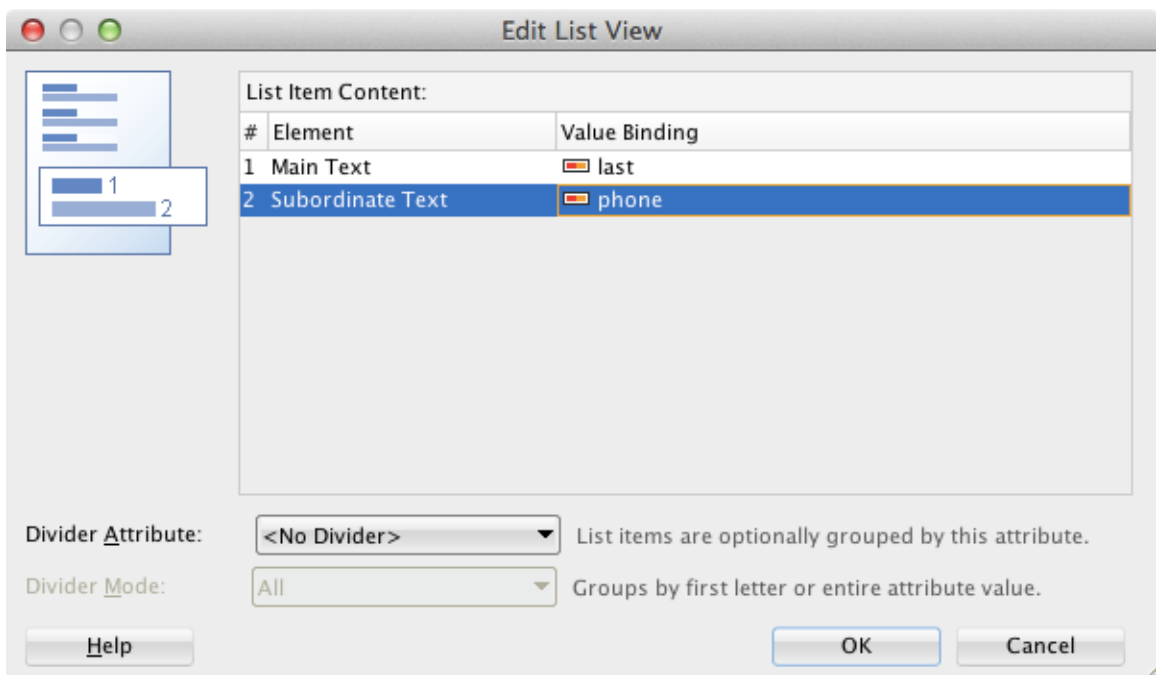


22. Click OK.

The **Edit List View** dialog appears.

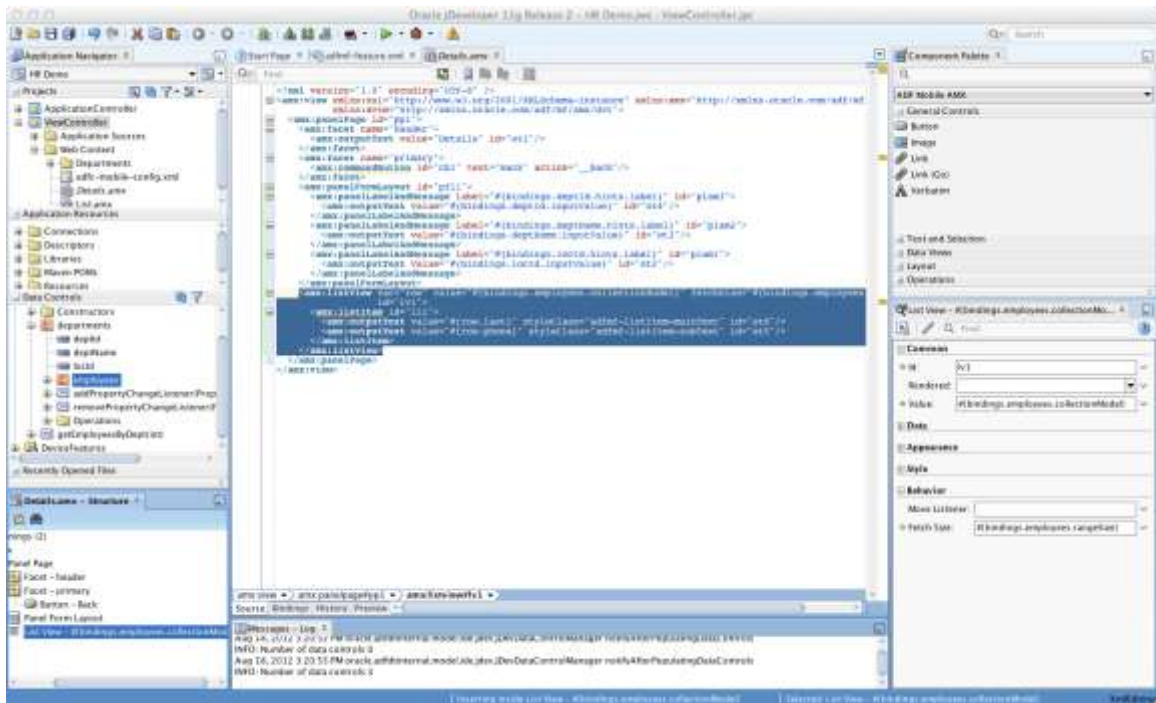


23. Change Main Text **Value Binding** to “last” and Subordinate Text **Value Binding** to “phone”

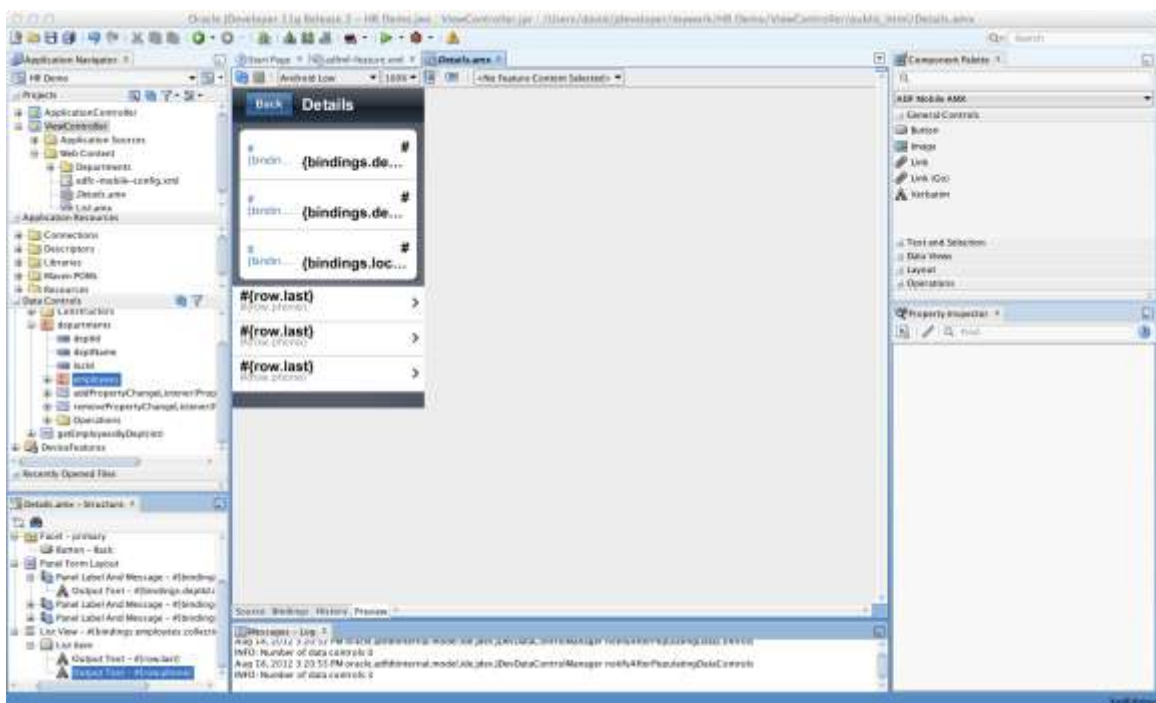


24. Click **OK**.

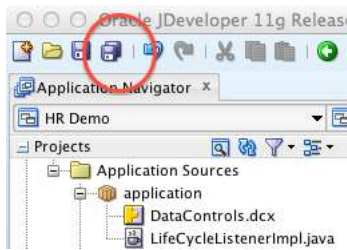
Observe in the source view that the list is set up.



25. Navigate back forth to the **Preview** tab at the bottom of the main editor window you'll see how the header will display.

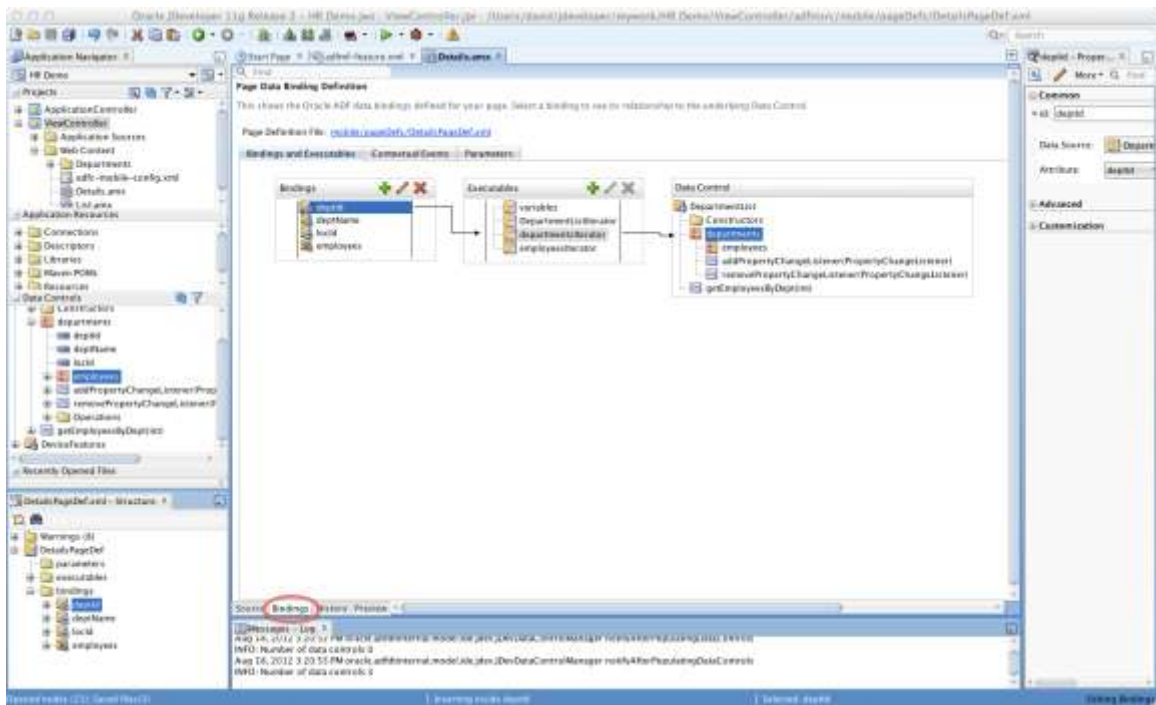


26. Save all files.



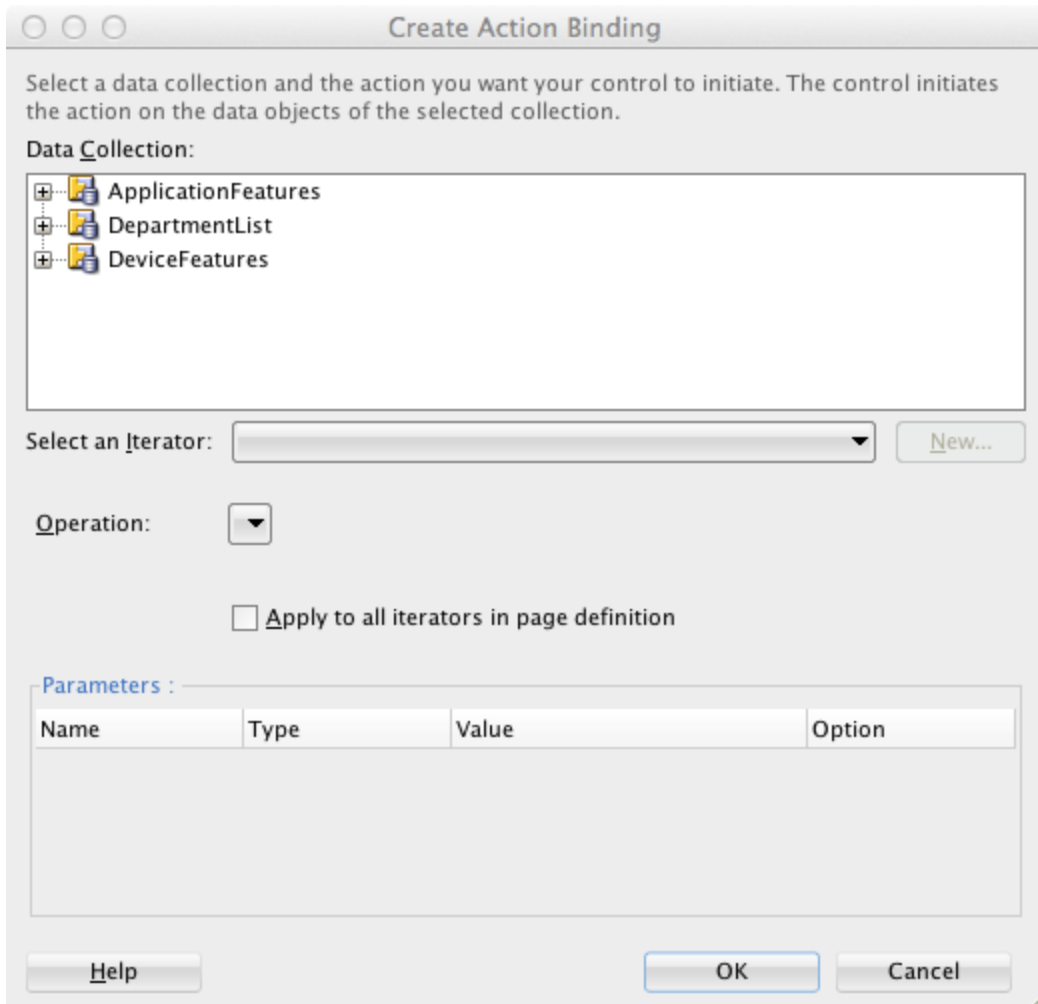
## Set Bindings for Details Page

### 1. Click **Bindings** view for Details.amx



### 2. In **Bindings** click the plus icon to add an item.

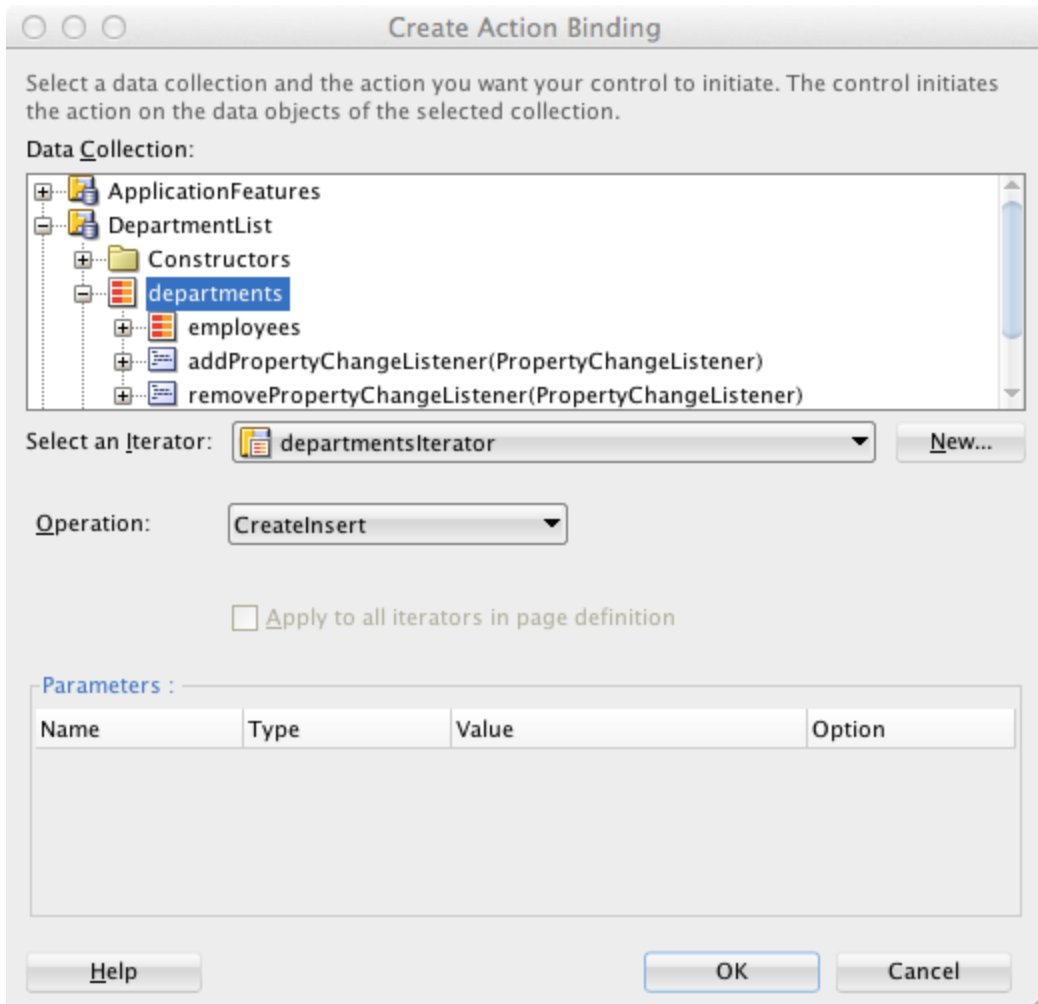




4. Expand **DepartmentList**, select **departments** and then in the '**Select an Iterator**' drop down select **departmentsIterator**

Don't click OK yet. There are more steps on this same dialog.

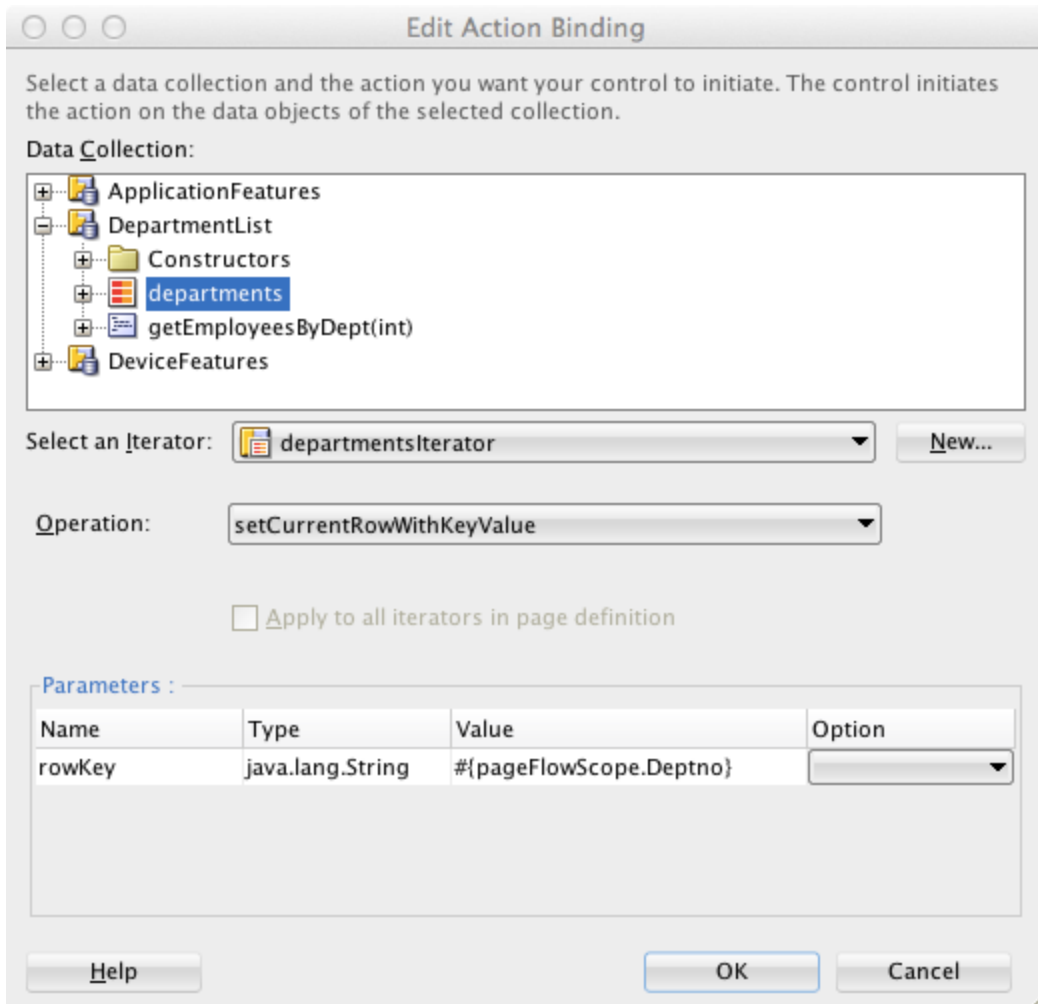
It should look like this:



5. Select **Operation** setCurrentRowWithKeyValue and enter the following for its rowKey **Parameter** value

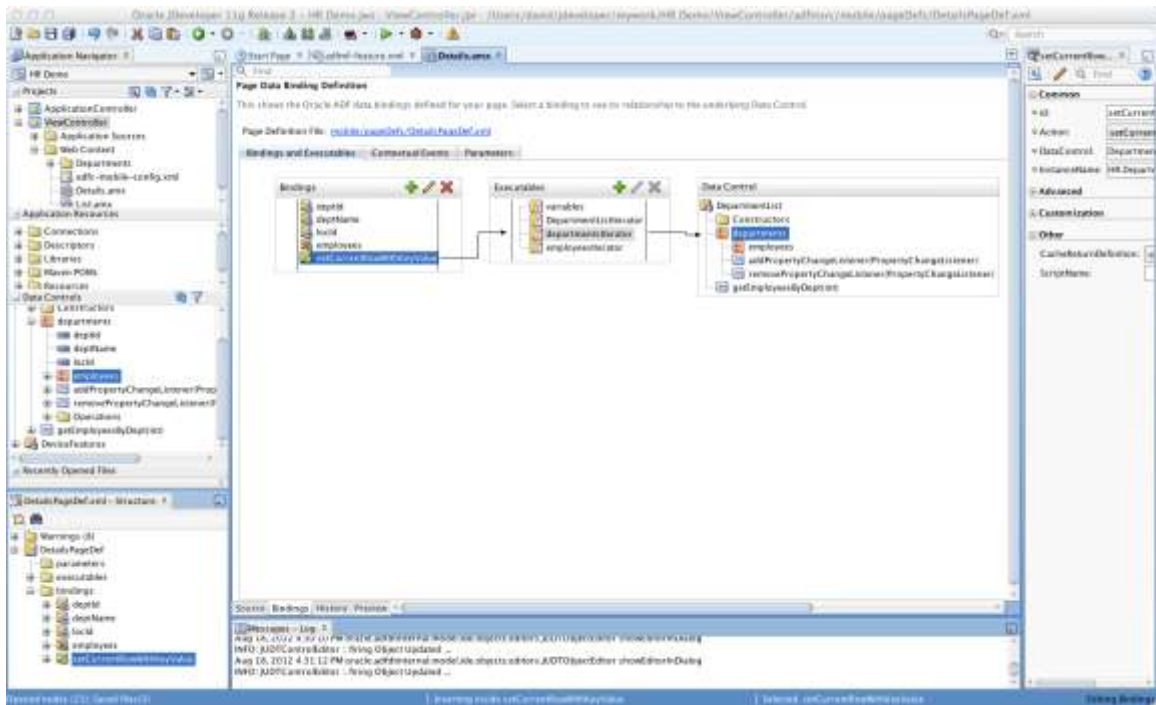
`#{pageFlowScope.Deptno}`

It should look like this:

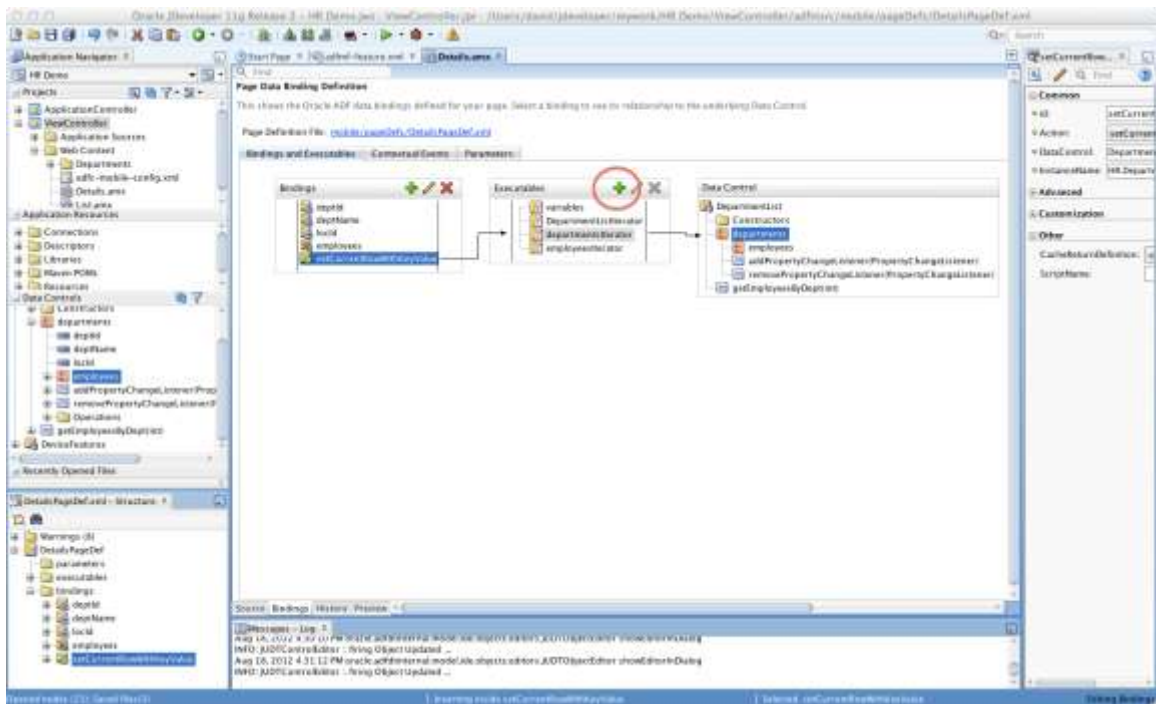


6. Click **OK**.

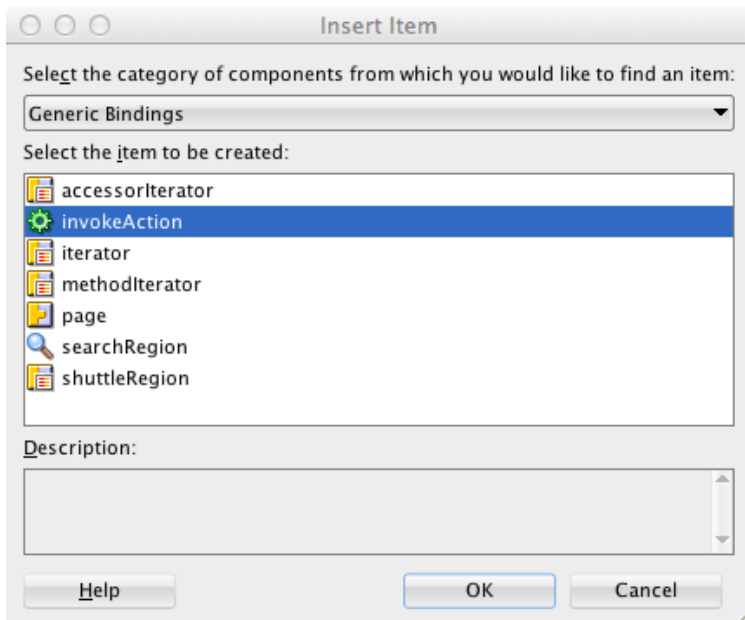
Observe, the setCurrentRowWithKeyValue is added to **Bindings**.



7. In **Executables**, click the plus icon to add an item.



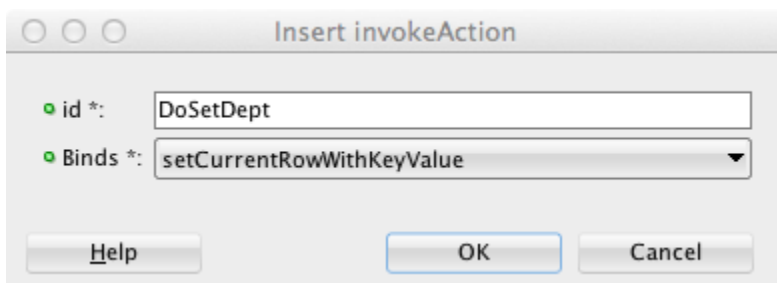
The **Insert Item** dialog appears.



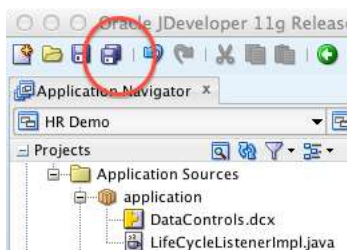
8. Select **invokeAction** and click **OK**.

The **Insert invokeAction** dialog appears.

9. Set the **id** to DoSetDept and set **Binds** to setCurrentRowWithKeyValue  
It should look like this:

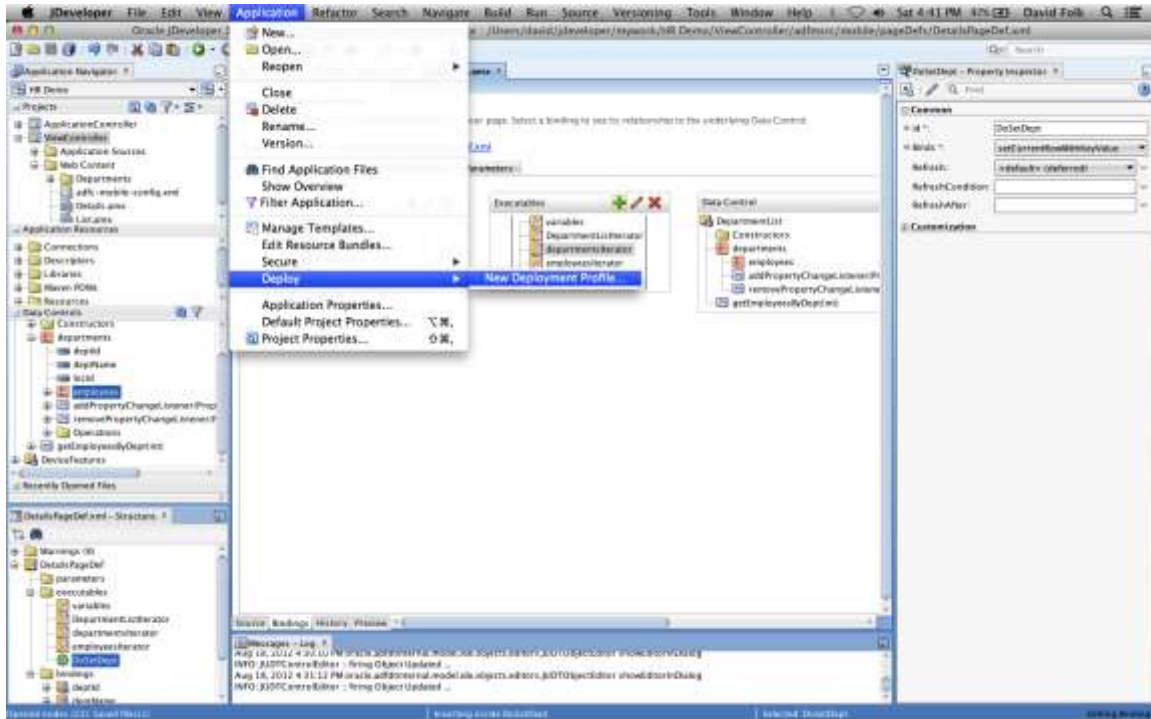


10. Save all files.

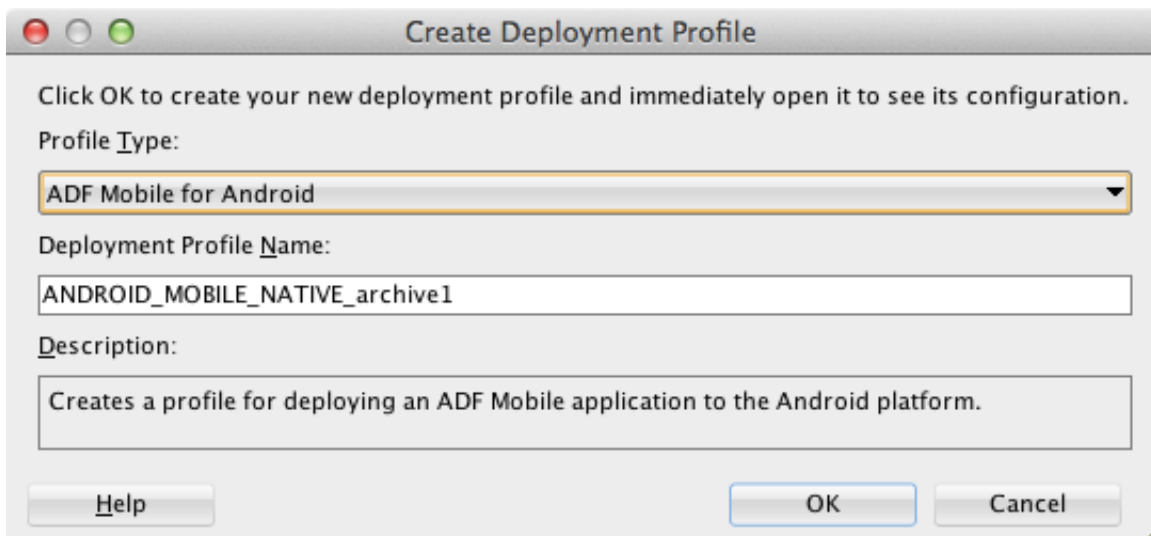


## Deploy to the iOS Simulator

## 1. Choose **Application -> Deploy -> New Deployment Profile**



The **Create Deployment Profile** dialog displays.



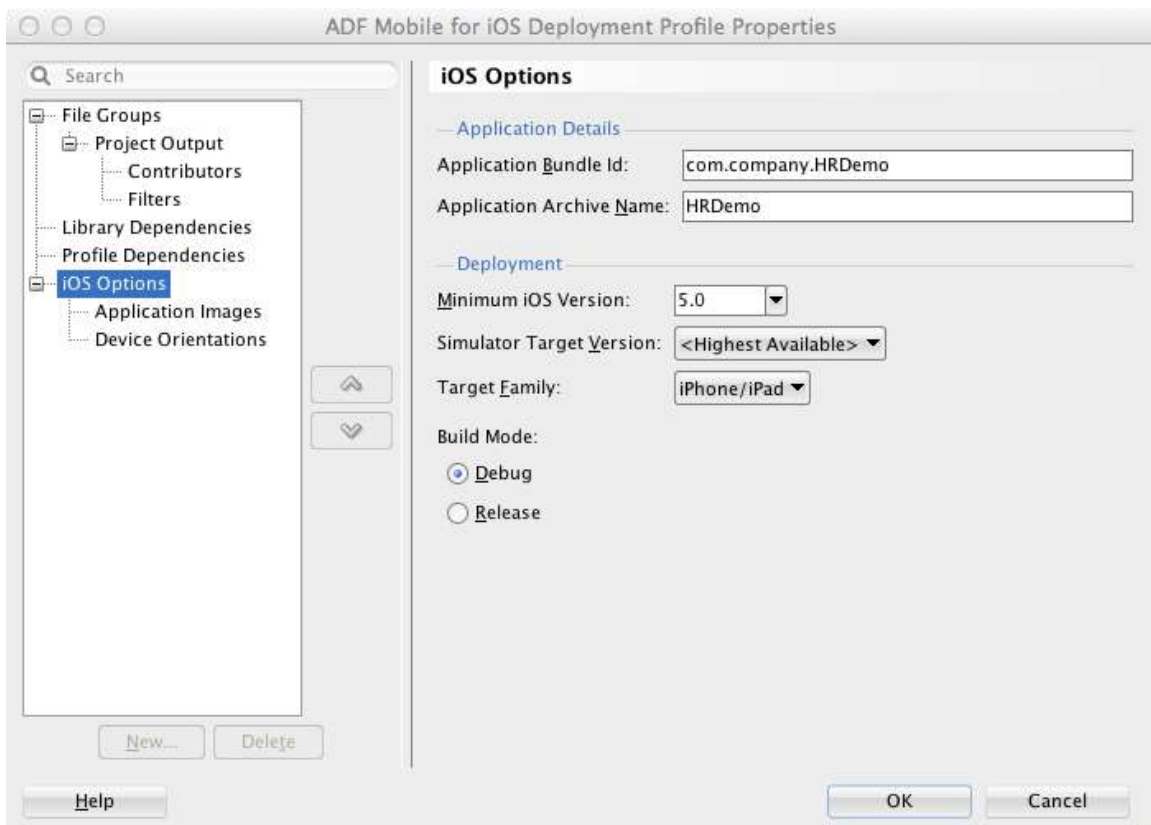
## 2. Change **Profile Type** to “ADF Mobile for iOS”



3. Click **OK**.

The **ADF Mobile for iOS Deployment Profile Properties** dialog displays.

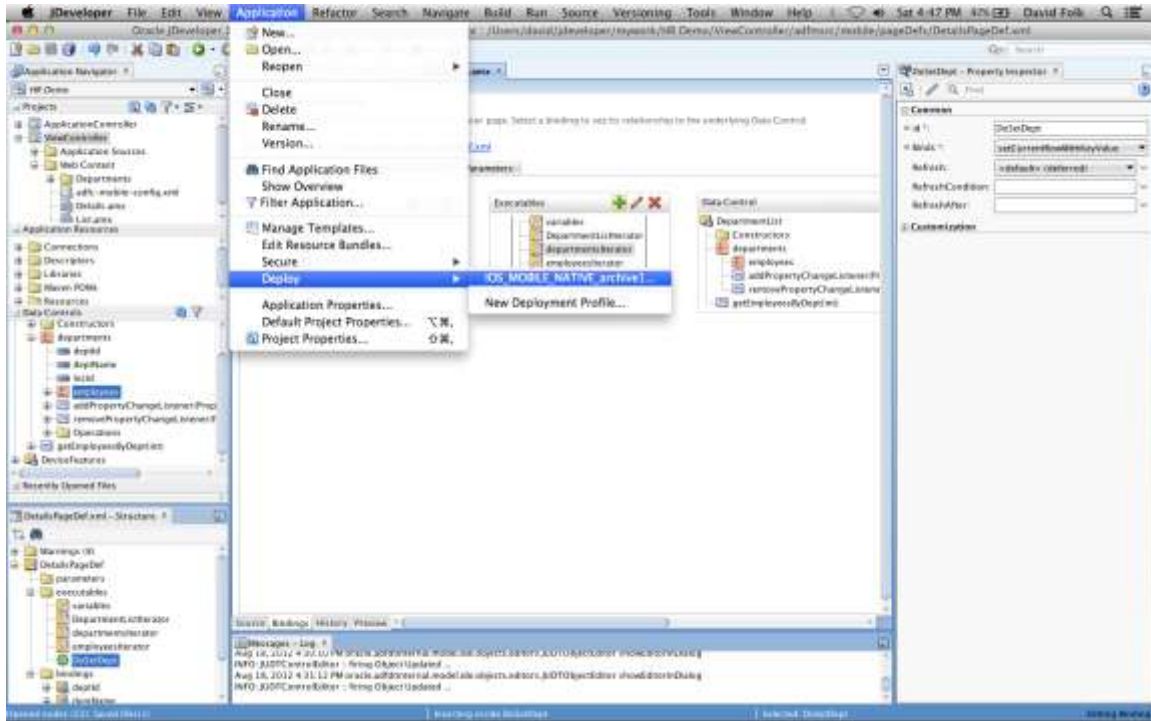
The most interesting things to observe on this dialog is are iOS Options:



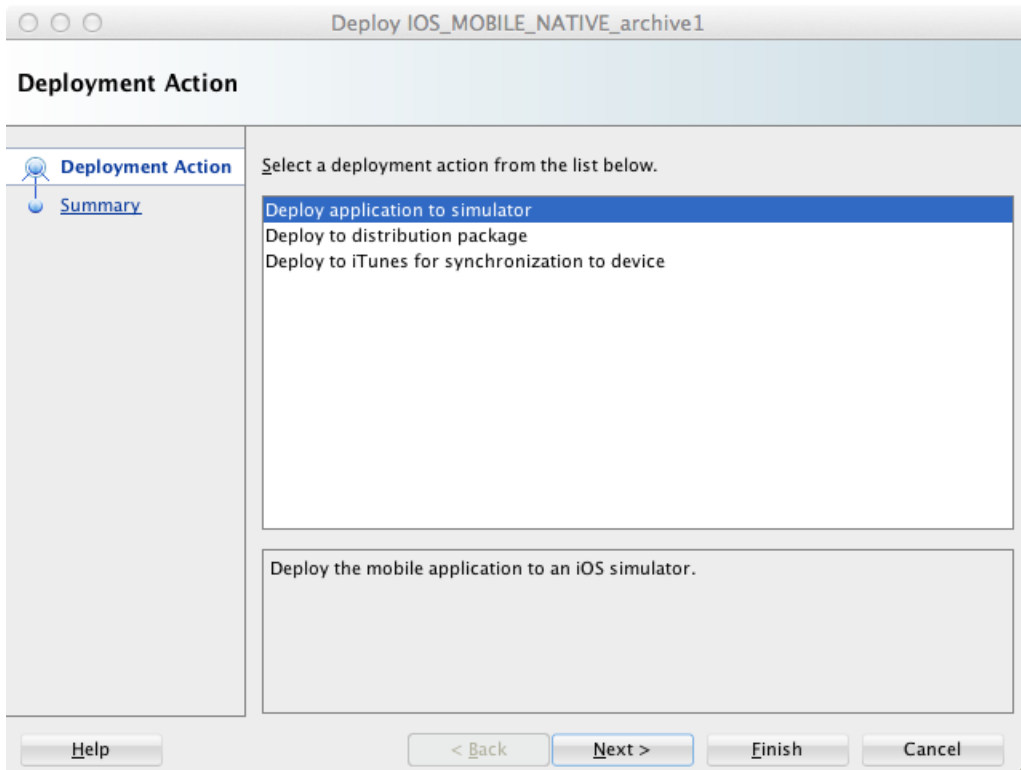
Don't change anything there.

4. Click **OK**.

5. Choose **Application -> Deploy -> IOS\_MOBILE\_NATIVE\_archive1...**



The **Deploy IOS\_MOBILE\_NATIVE\_archive1** dialog displays.



6. Click **Finish**, thus accepting all defaults and deploying to the simulator.

Deployment will commence. If you click the **Deployment** tab you can see status. It will likely take between 10 seconds to a minute or more depending on hardware provided for the lab.



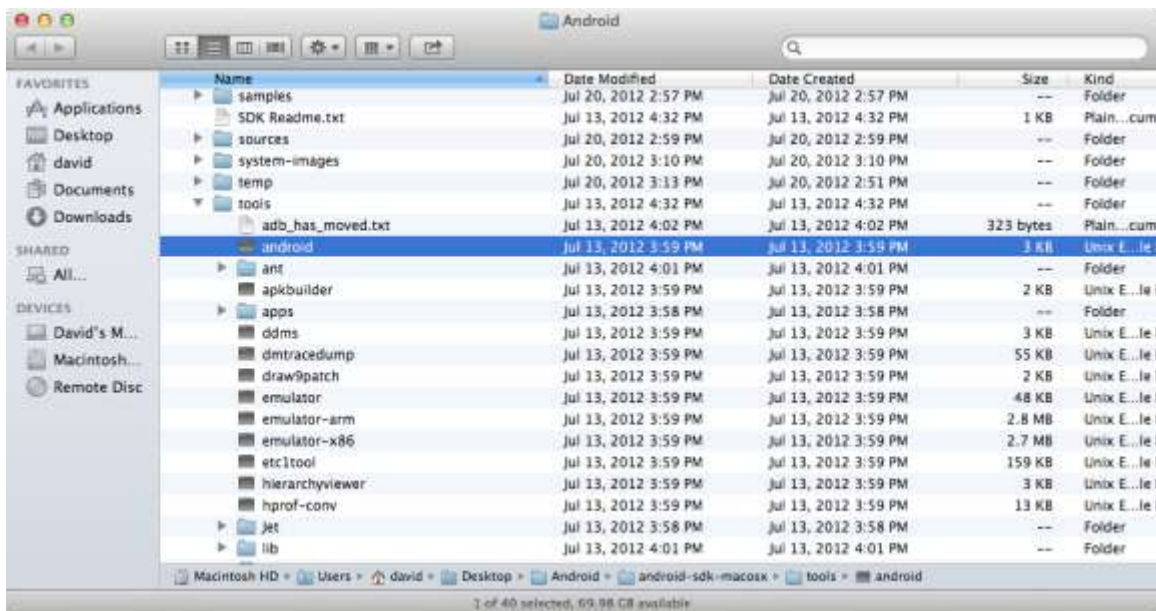
IMPORTANT: Unlike iOS Simulator which gets started during the deployment process in JDeveloper, for Android, the Emulator must be running prior to deployment.

1. Open the **Finder** Application.

The Finder icon looks like this at the far left on your desktop.

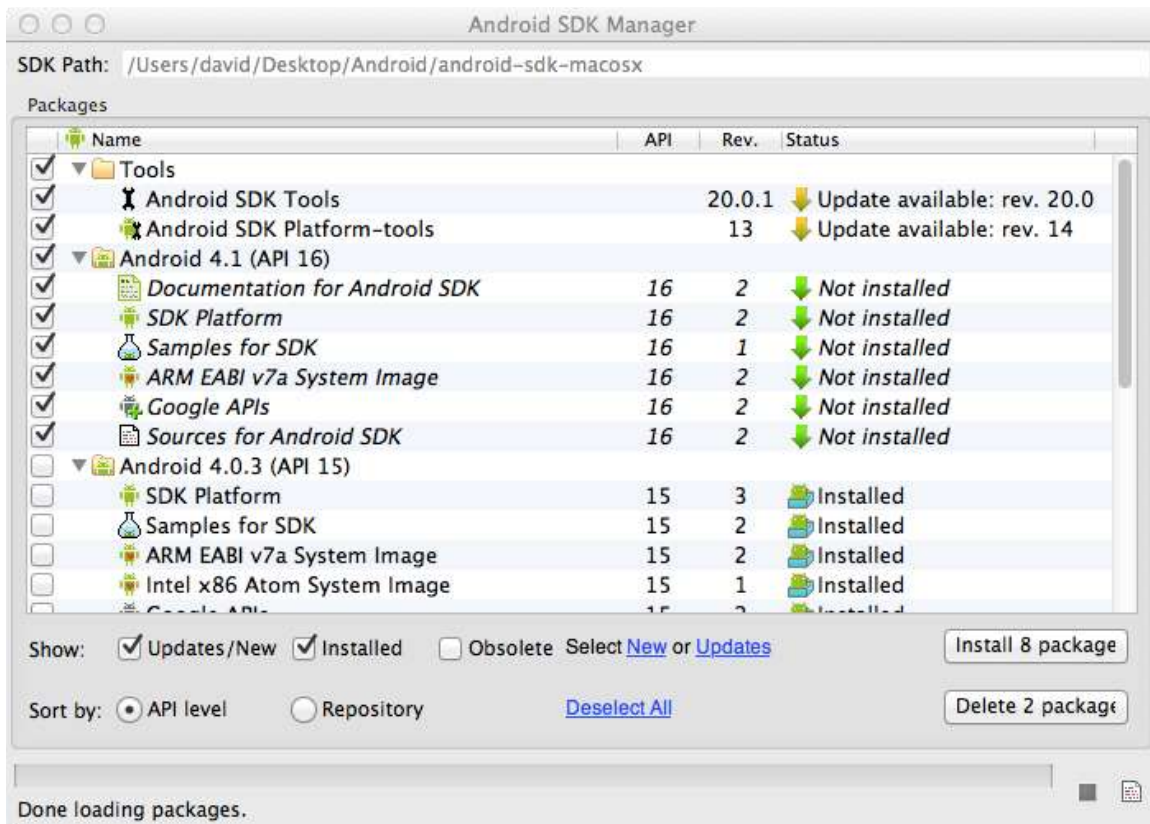


2. Navigate to the folder `/Applications/Android/android-sdk-macosx/tools`

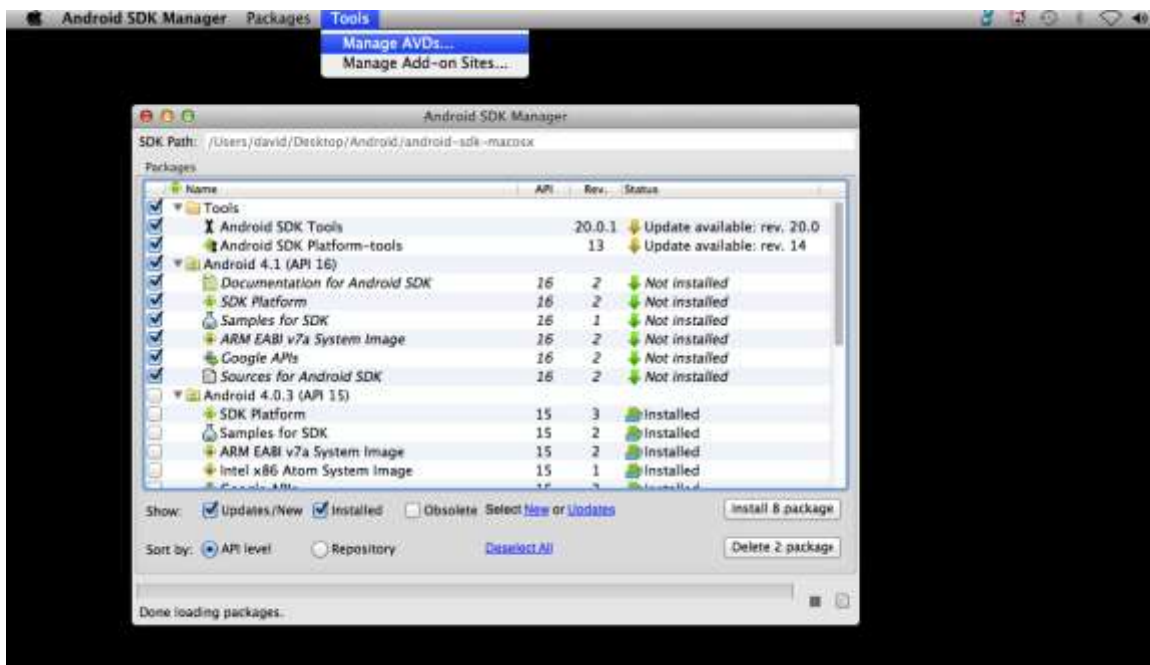


3. Double-click the Android application

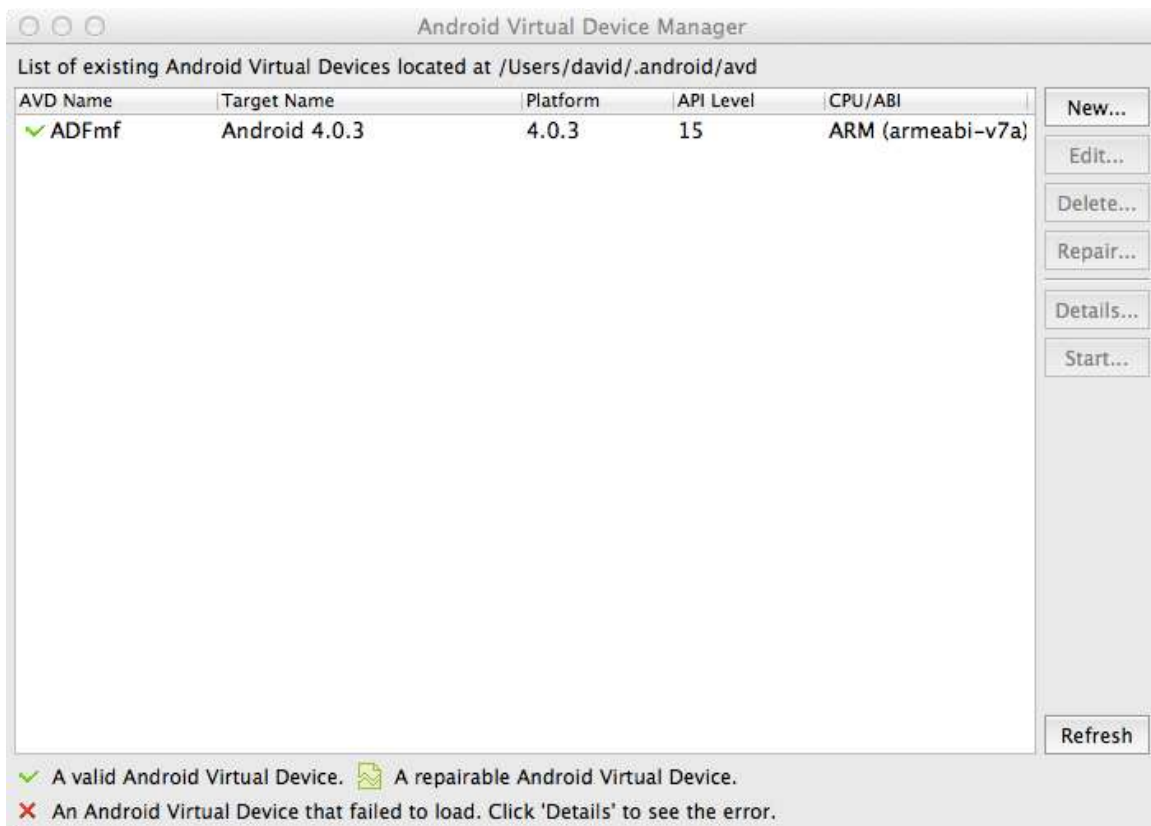
The Android SDK Manager opens.



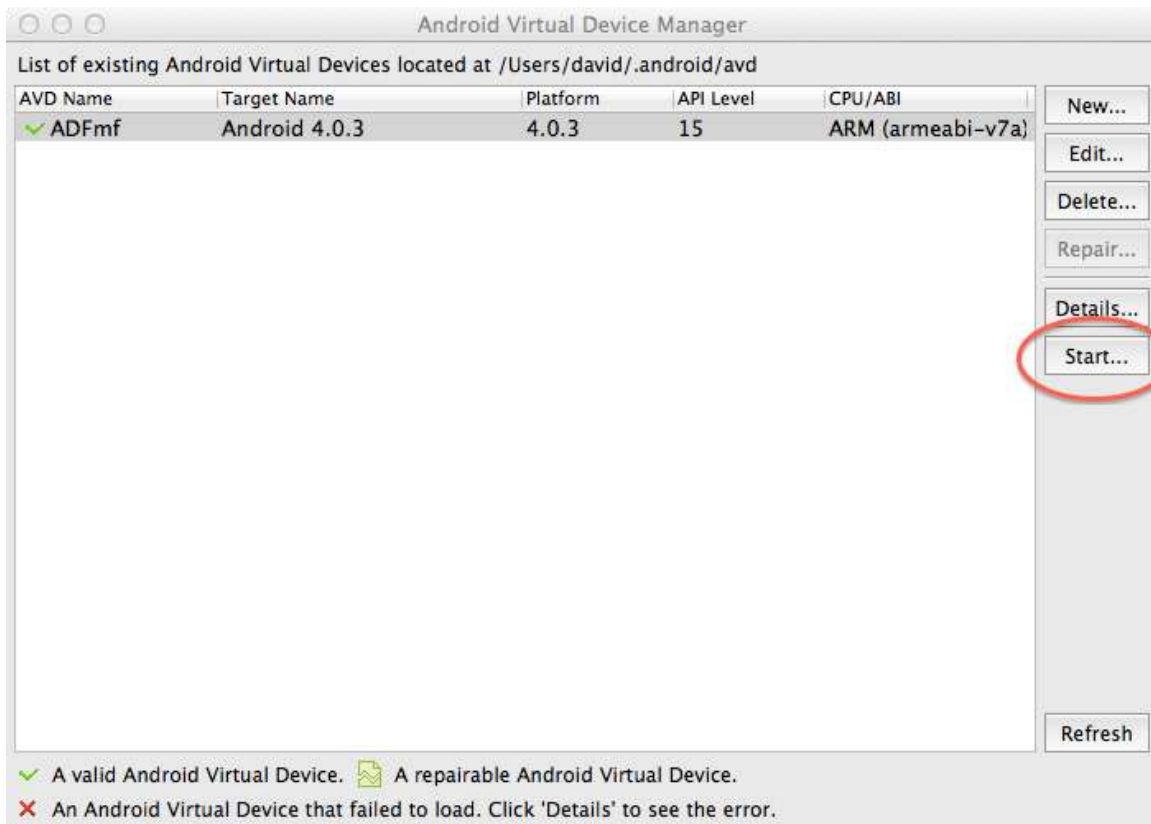
#### 4. Choose Tools -> Manage AVDs...



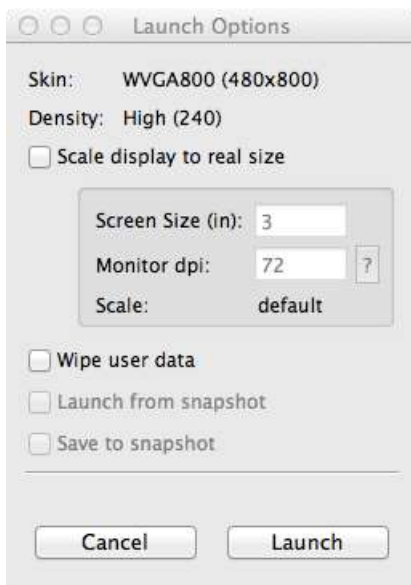
The **Android Virtual Device Manager** dialog displays.



5. Select the ADV and click the Start... button to start it.

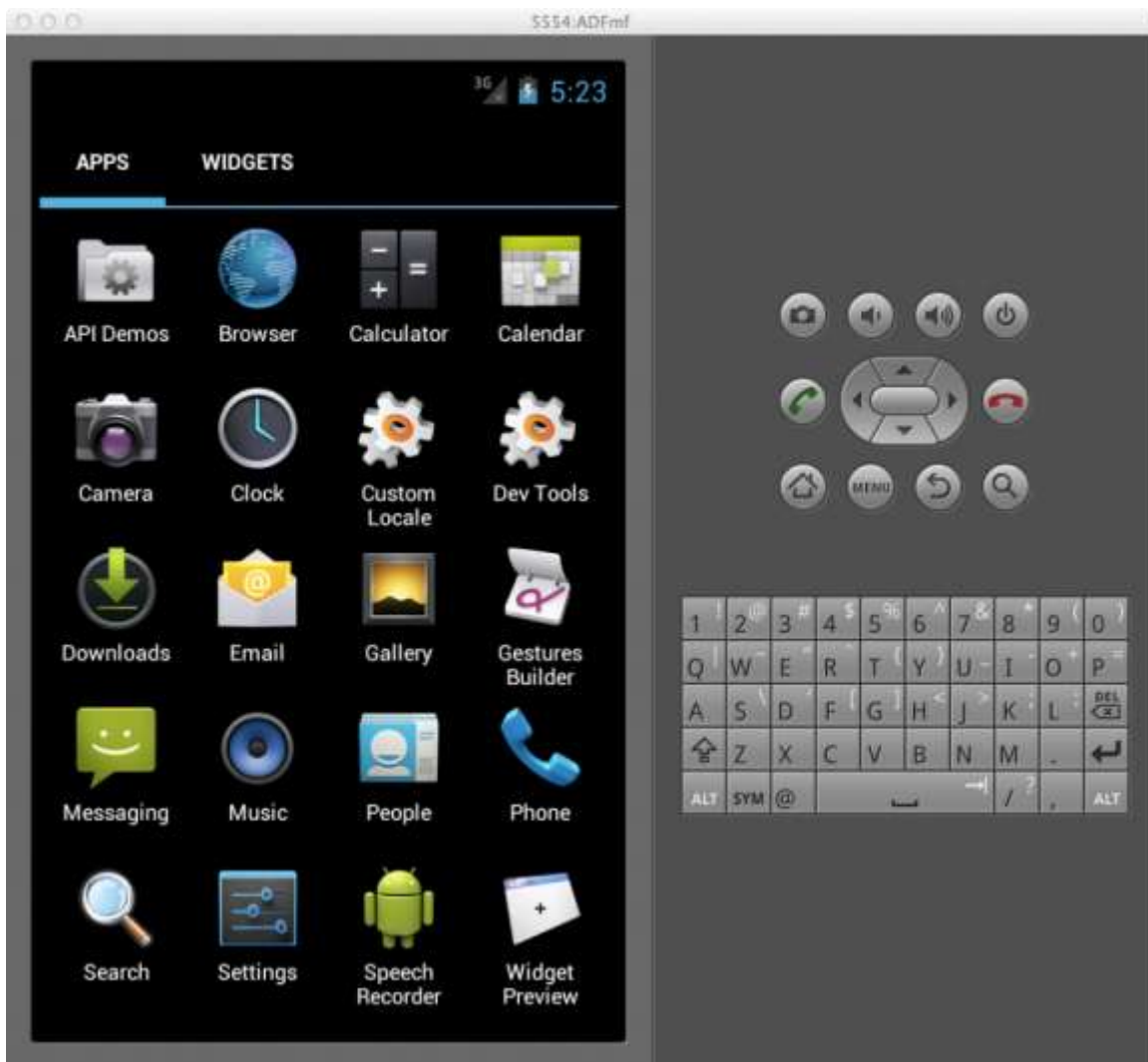


The **Launch Options** dialog displays.



6. Click **Launch**.

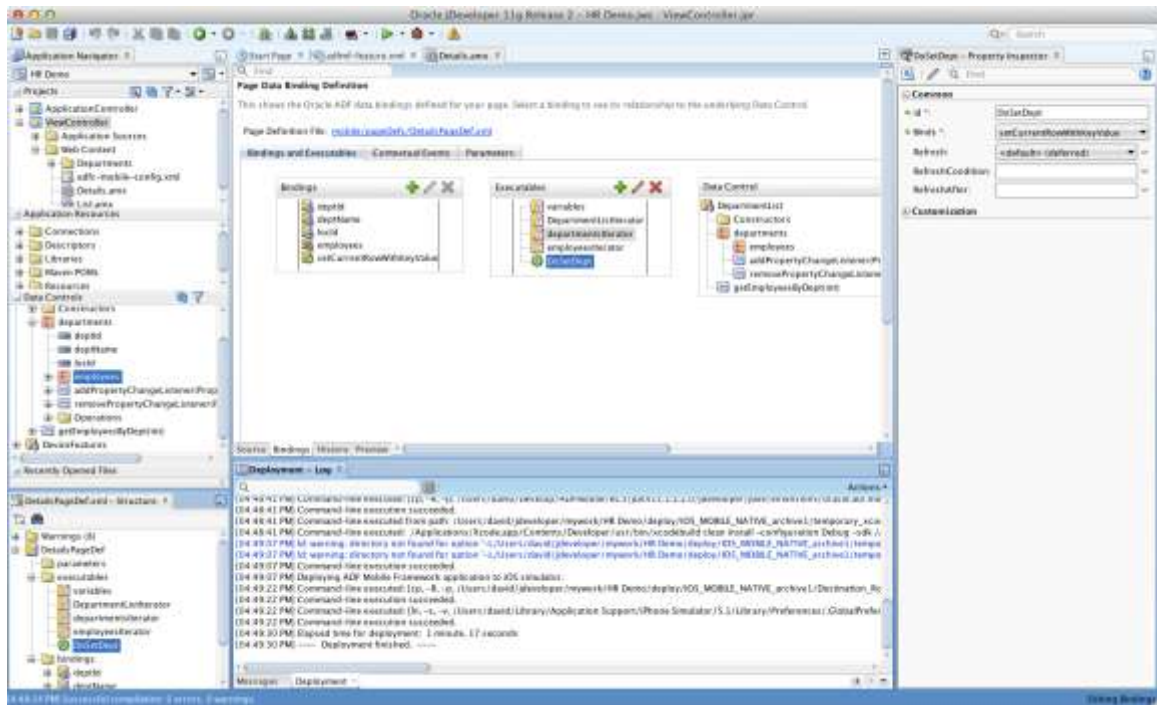
The **Android Emulator** starts. Feel free to click around to get acquainted with it if you aren't already.



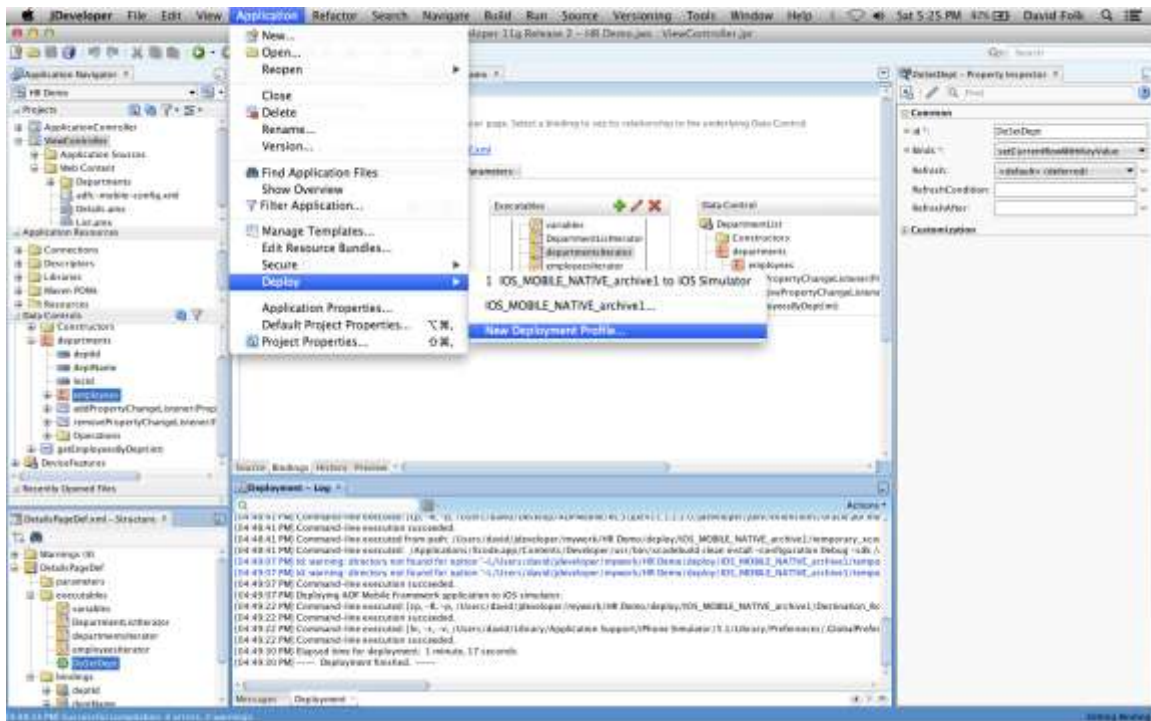
7. Return to **JDeveloper** with the HR Demo application open.

If JDeveloper is closed/minimized click the icon shown below.

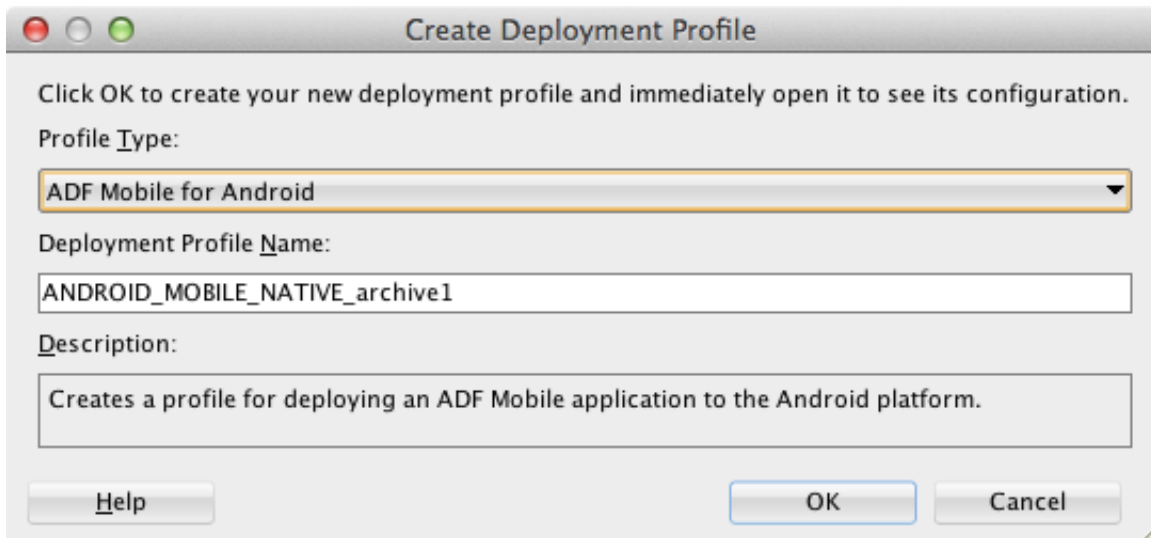




## 7. Choose Application -> Deploy -> New Deployment Profile



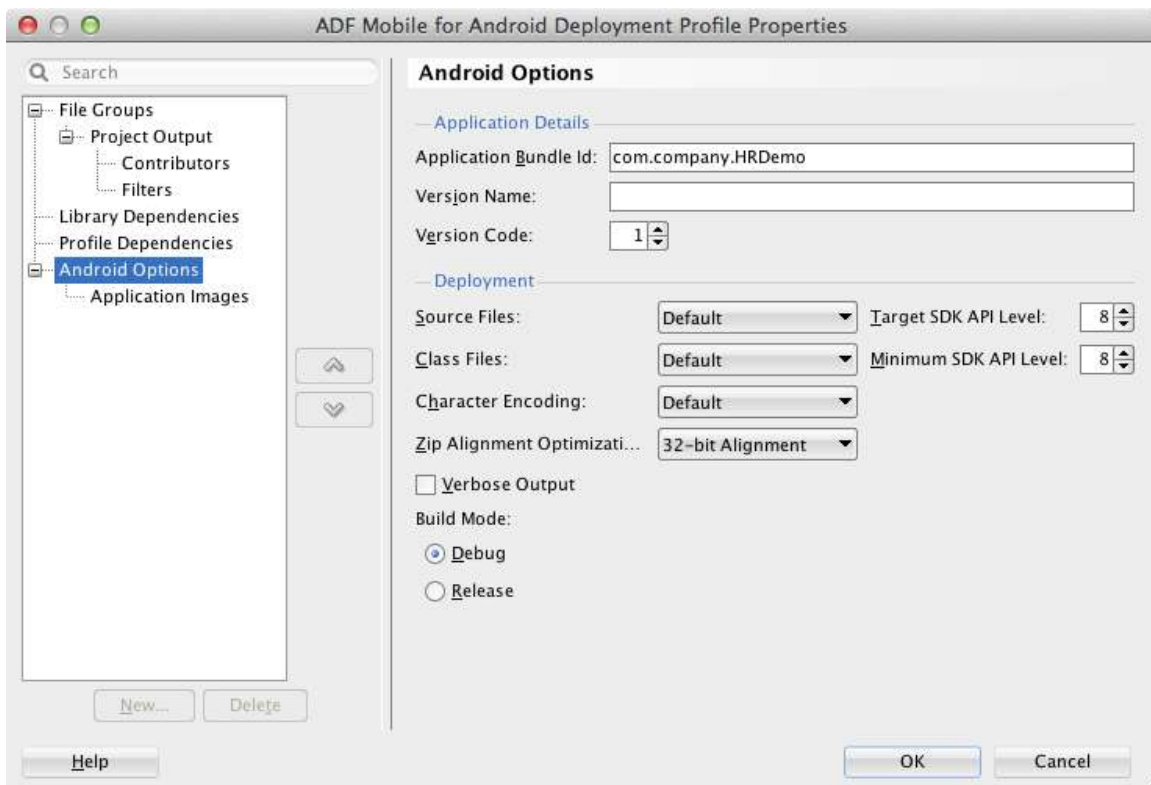
The Create Deployment Profile dialog displays.



8. Click **OK**.

The **ADF Mobile for Android Deployment Profile Properties** dialog displays.

The most interesting things to observe on this dialog is are Android Options:



**Note:** You *shouldn't* have to change anything here. But as we're writing this lab there's an issue in the extension where **Application Bundle Id** fails to concatenate

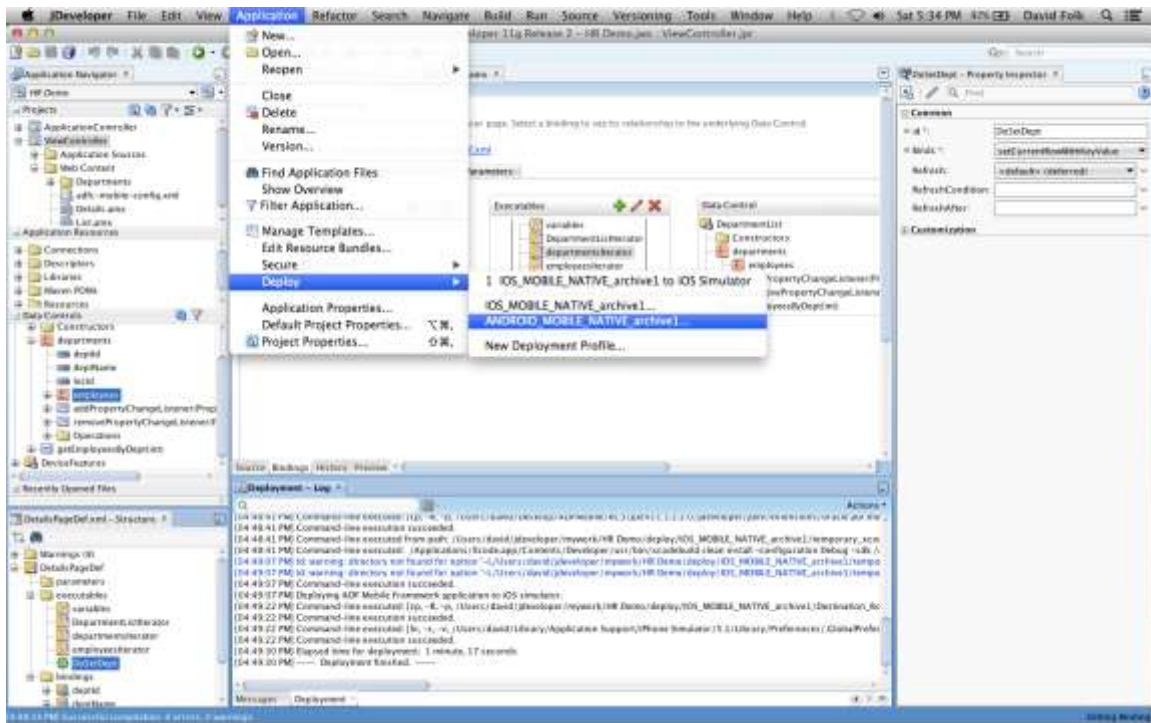
the string HR Demo. If you see an error at the time you take this lab, just make the **Application Bundle Id** as follows:

com.company.HRDemo

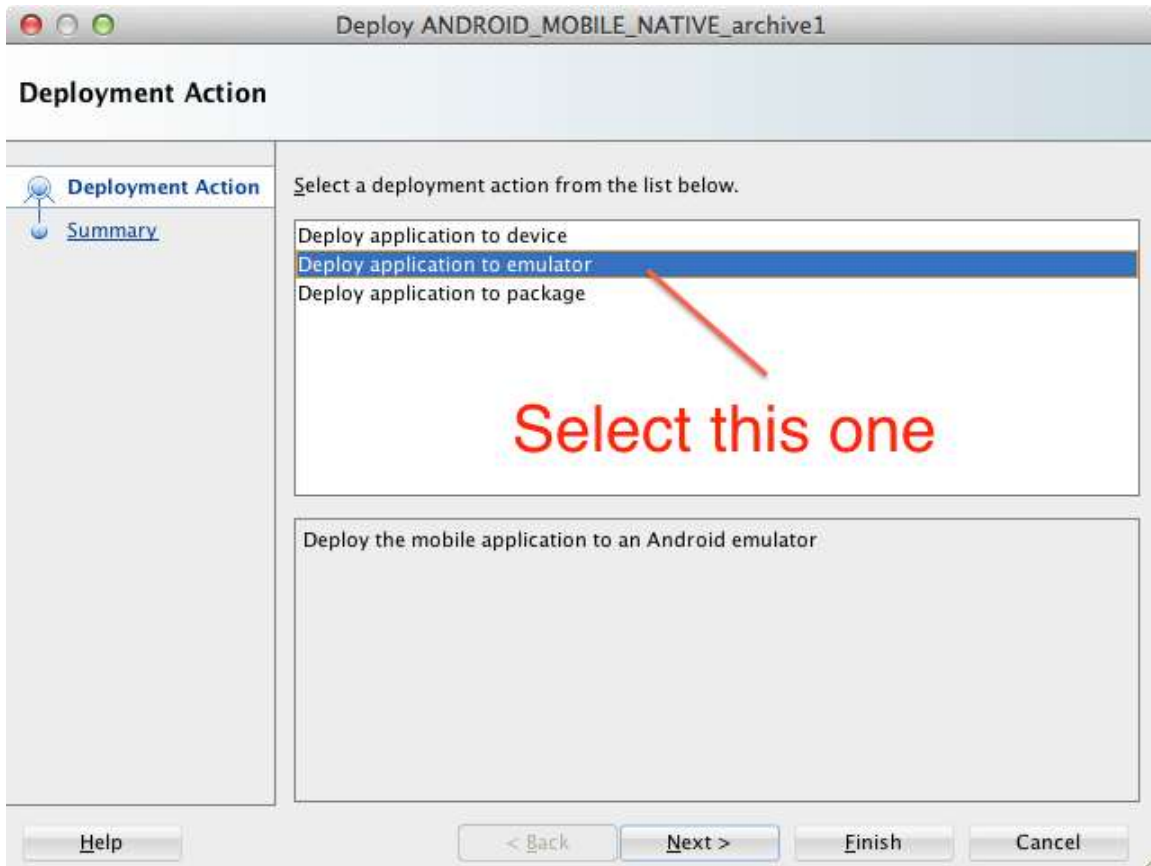
With no space between “HR” and “Demo”.

9. Click **OK**.

10. Choose **Application -> Deploy -> ANDROID\_MOBILE\_NATIVE\_archive1...**

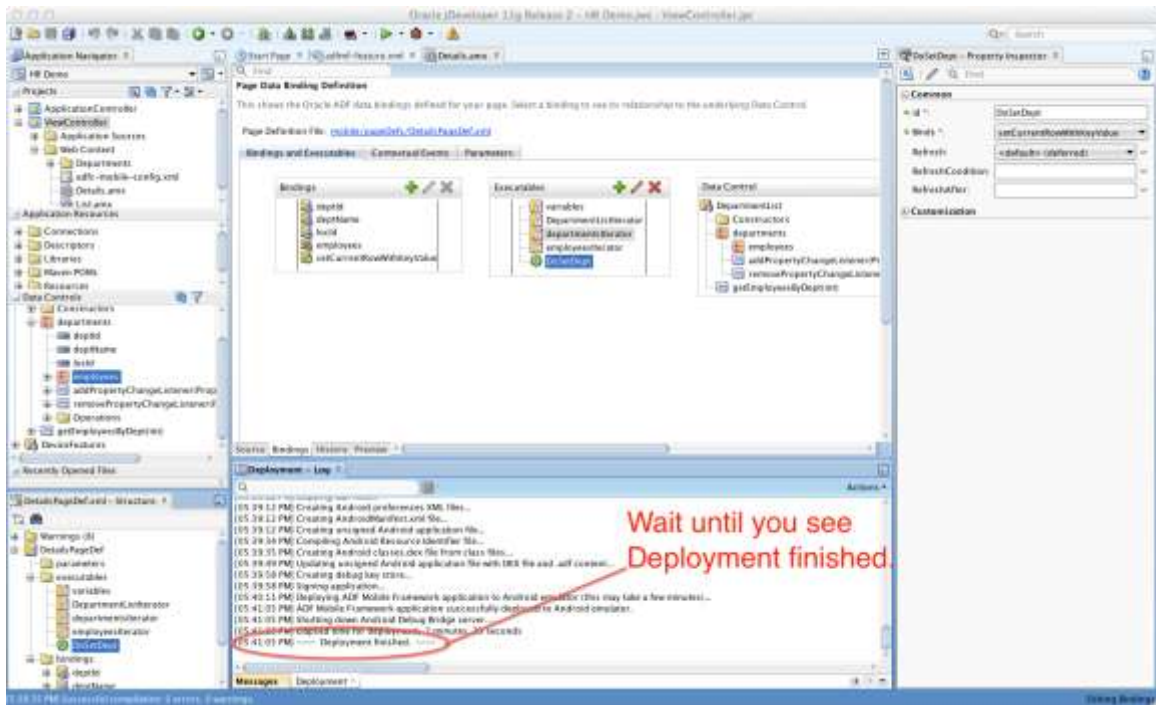


The **Deploy ANDROID\_MOBILE\_NATIVE\_archive1** dialog displays.



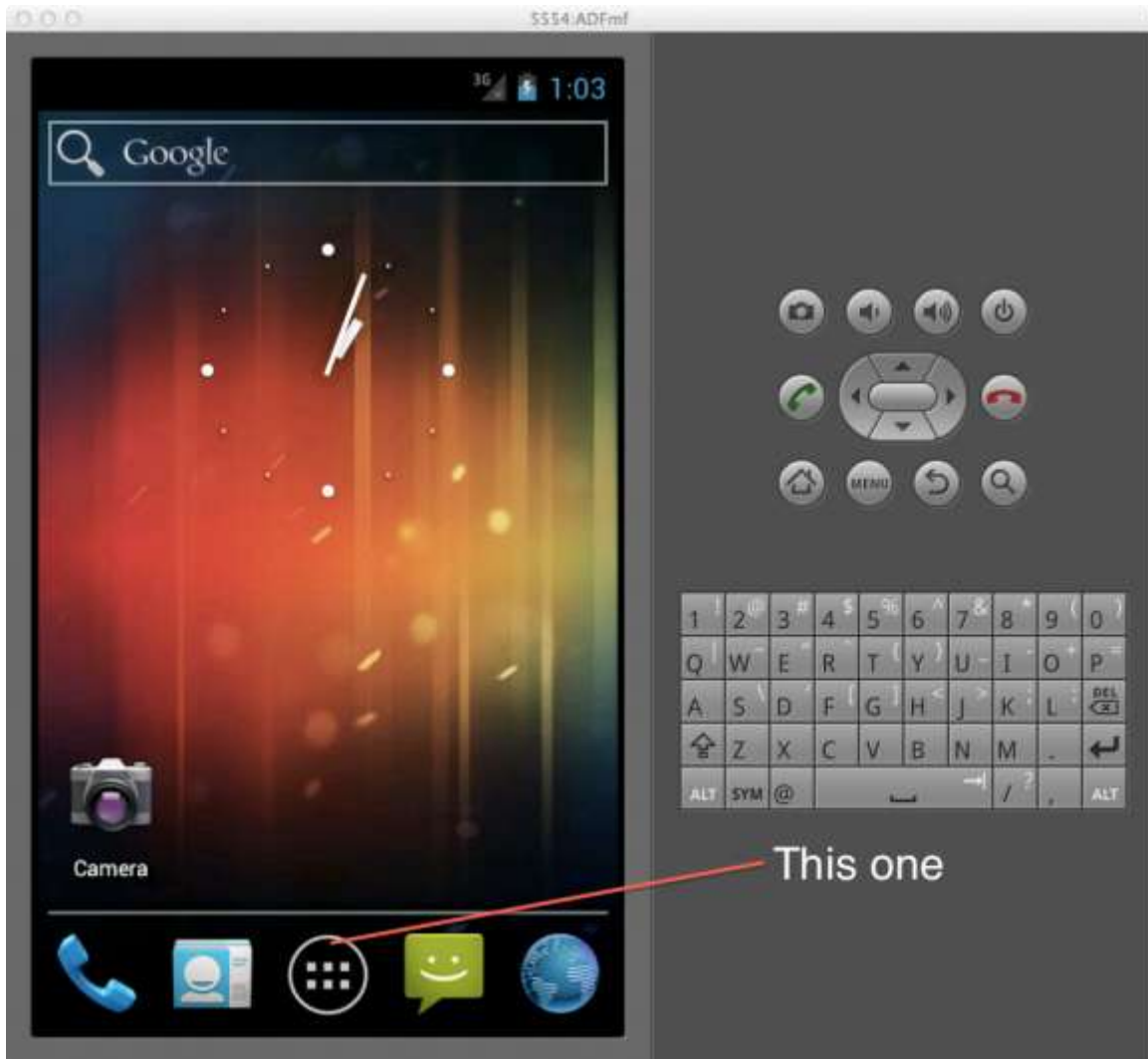
11. Click **Finish**, thus accepting all defaults and deploying to the emulator.

Deployment will commence. This will take longer than deployment did to the iOS Simulator. That's outside the control of ADF Mobile – the emulator is just a different tool.

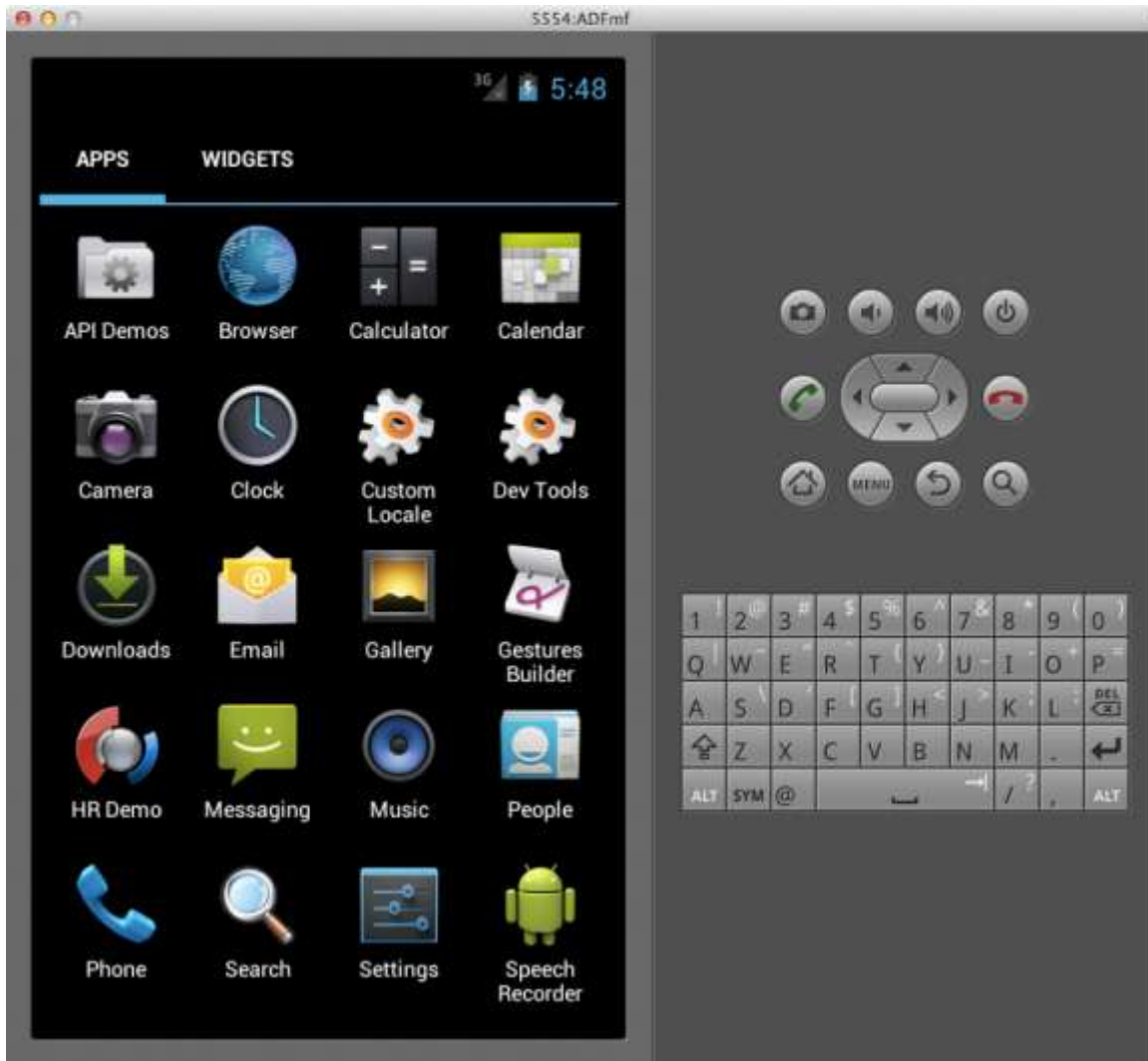


12. When deployment completes, return to the **Android Emulator**.

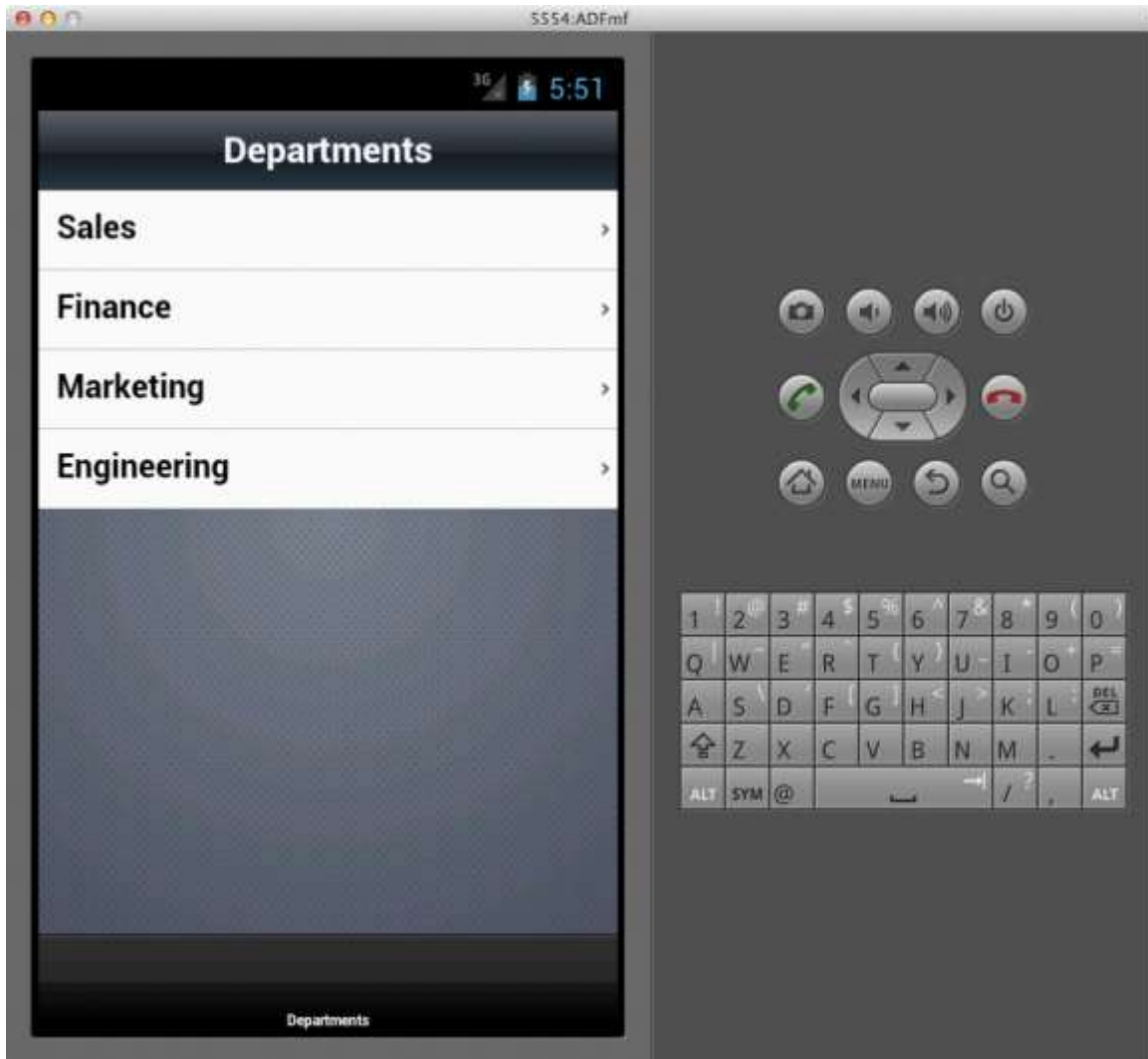
13. Click the icon to navigate to APPS.

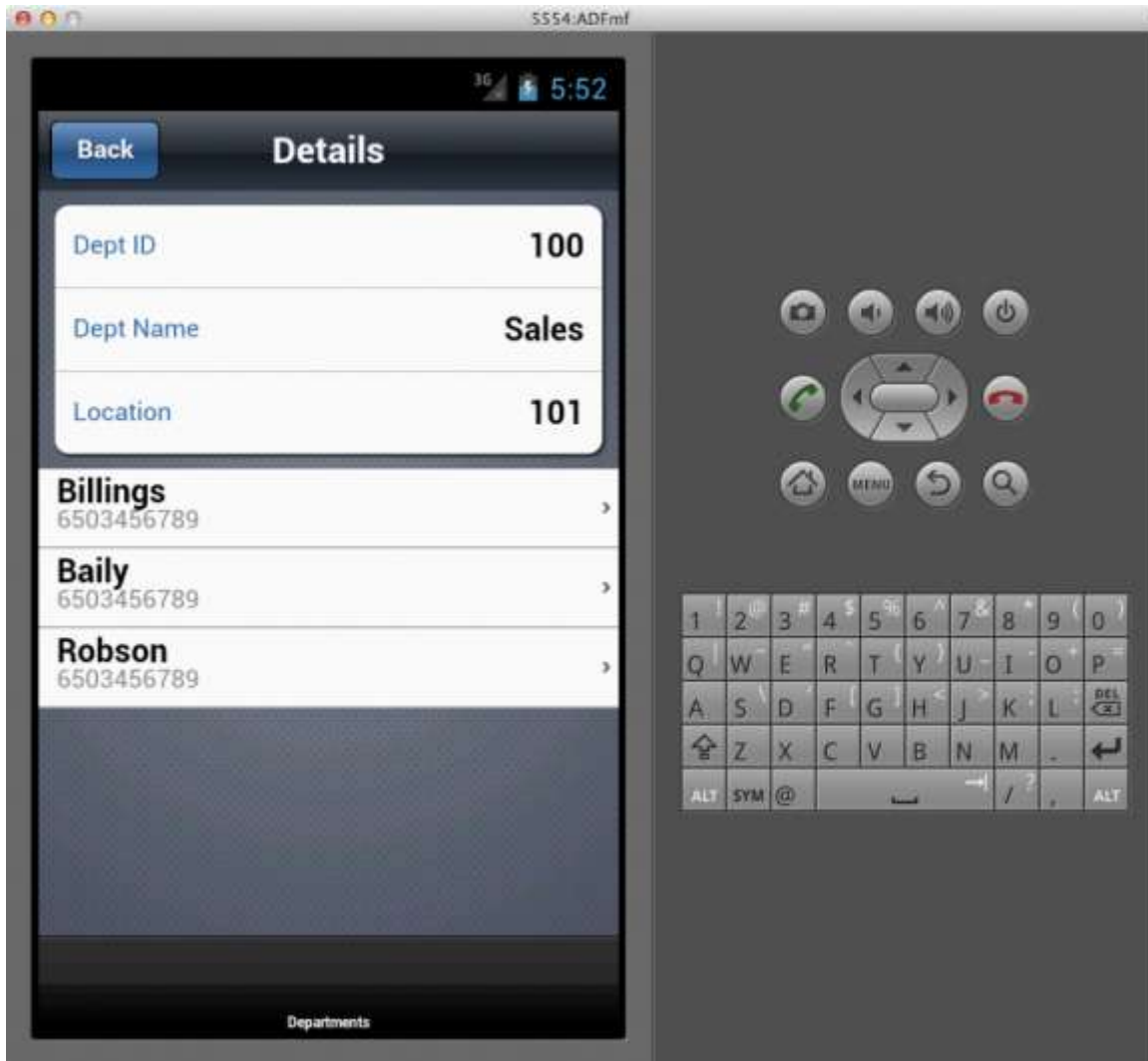


14. Click the HR Demo icon and open and try out the application



The first startup will take a few extra seconds.





## Finished Early?

Take a look at the **PublicSamples** folder on the desktop. These samples are included with ADF Mobile extension (we copied them here for convenience).

### HelloWorld - Basic application

This applications is the proverbial "hello world" application for ADF Mobile. It should you the basic structure of the framework and how things are setup. It has a single mobile feature that is implemented with a local HTML file.

This demo is useful to test if the development environment is setup correctly to compile/deploy an application.

[OBJ]

### CompDemo - Component Demonstration application

This application demonstrates all the components available in ADF Mobile. It allows users to change their attributes and see the effects of those changes in real time without recompiling and deploying a new app each time.

### **LayoutDemo - UI layout demonstration**

This application shows developers how to create the various list and button styles in popular mobile apps. It also demonstrates how to do the action sheet style of popup.

### **JavaDemo - Java Integration application**

This application demonstrates how to hook up your UI to Java beans. It also demonstrates how to invoke EL bindings from Java using the supplied utility classes.

### **Navigation - Navigation techniques**

This application shows developers the various navigation techniques in ADF Mobile including bounded task flows and routers. It also demonstrates the various page transitions.

### **LifecycleEvents - Application and feature lifecycle event handlers**

This application implements lifecycle event handlers on the application and each feature. It shows the developer where they can insert code into the system so they can perform their own logic at certain points in the lifecycle.

### **DeviceDemo - Device Feature Implementation**

This application shows the developer how to use device features like GeoLocation, Email, SMS and Contacts as well as query the device for its properties.

### **GestureDemo - Gesture Demonstration**

This application demonstrates how gestures can be implemented and used in ADF Mobile applications.

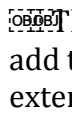
### **StockTracker - Data Change event demonstration**

This application demonstrates how data change events work in order to change data in Java and have it reflected in the UI. It also has a variety of layout use cases, gestures and basic mobile patterns

### **HR - Human Resources Demo**

This is a full CRUD application that demonstrates a variety of real world application techniques. It uses a local SQLite database to store its data and persists the data between application starts and is based on the HR schema that comes with all Oracle databases by default. This application also has different layouts for both iPad and iPhone to show how you can have the same data model but different UIs. There are a variety of other patterns demonstrated in the application as well.

### **Skinning - Application Skinning demonstration**

 This application demonstrates how developers can skin their applications and add their own unique look and feel by either overriding the supplied style sheets or extending them with their own style sheets.