

2016

Sivakumar Gonugunta

USING ADF BC REST SERVICES IN ORACLE MOBILE APPLICATION FRAMEWORK APPLICATION (MAF)

This document works as tutorial to learn about consuming the ADF BC REST Services in Oracle Mobile Application Framework Application (MAF).

Table of Contents

Use case	3
Creating ADF BC REST Service	3
Creating Release Version	4
Exposing VO as REST resource	4
Deployment	7
Integrated WLS.....	8
Standalone WLS.....	8
Developing Mobile Application	11
Creating MAF Application.....	11
Creating REST Connection.....	14
Creating Application Feature	16
Code	19
Employee.Java.....	19
Department.java	22
ItemsList.java	25
CmnRestAdapter.java	27
DepartmentDC.java	30
Creating AMX Pages	33
Deployment	42
Testing.....	45
Downloads	48

Use case

We will create a MAF application displaying list of departments and provide a drill down to list of employees for each department displayed. This information will be fetched using ADF BC REST Service.

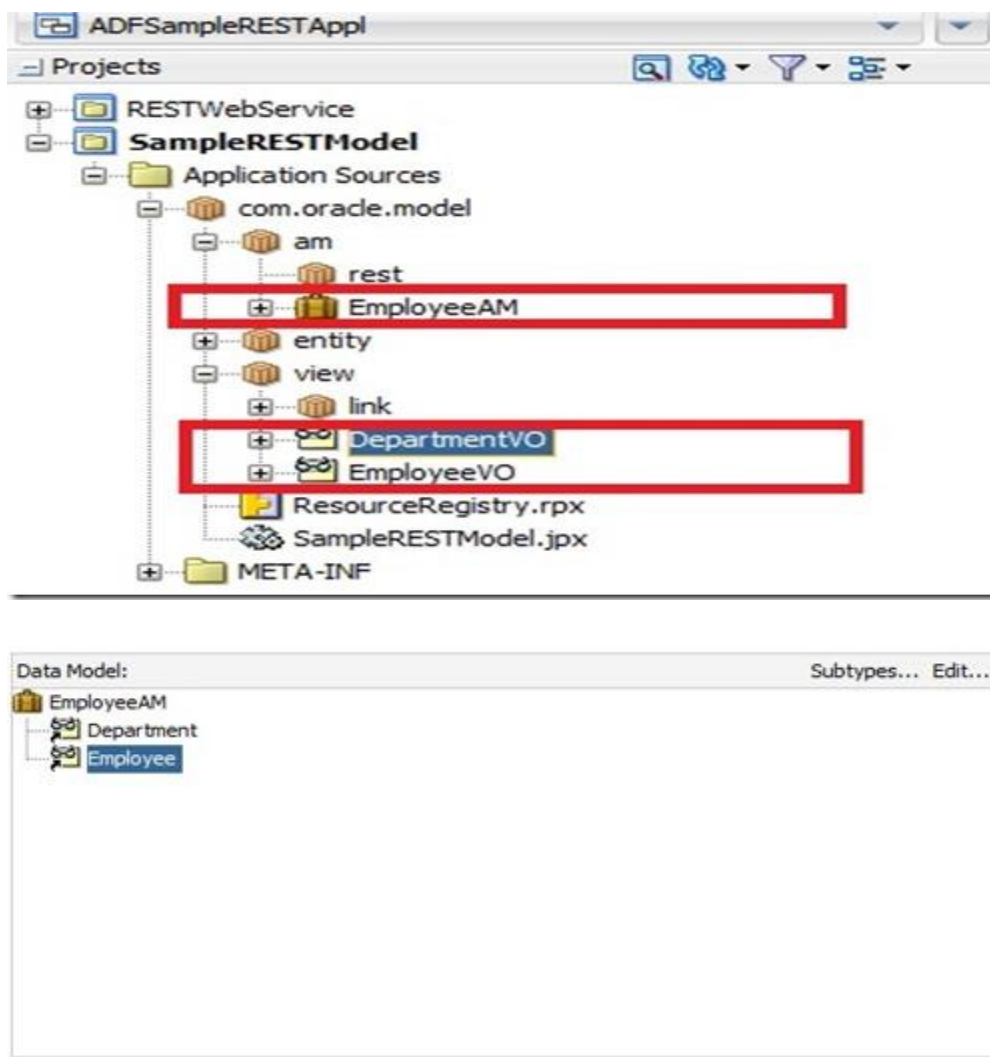
The main focus of the tutorial is using ADF BC REST Services in MAF application. So I would expect readers to have the basic understanding of MAF terminology and deployment options. I used MAF 2.2.1 for this tutorial and documentation for the same can be found [here](#).

Refer [here](#) for MAF installation instructions.

Creating ADF BC REST Service

Oracle added REST Service support in ADF starting from 12.2.1 release. So we should have JDeveloper 12.2.1 to create ADF BC REST Services. For simplicity, I omitted details about creation of Entity Objects (EO), View Objects (VO) and Application Module (AM).

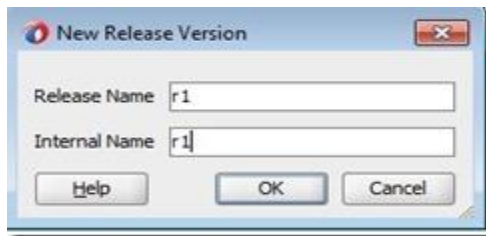
We will use the following VOs and AM Data Model to expose the REST Services.



Creating Release Version

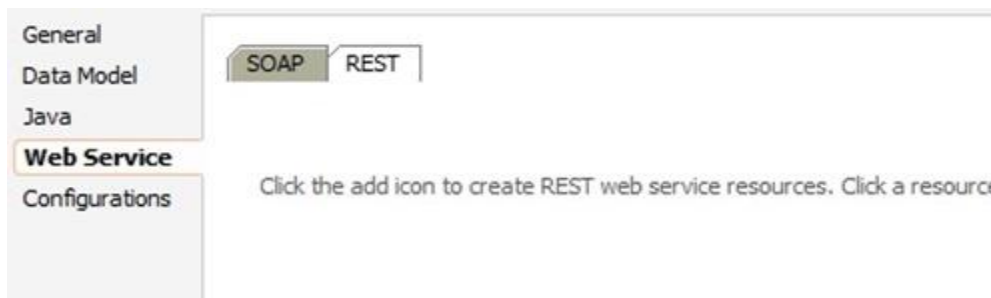
Creating a release version is the first step in developing ADF BC REST resources which will allow us to do the versioning of services. We use **adf-config.xml** to create these release versions.

So open **Application Resources -> Descriptors -> ADF META-INF -> adf-config.xml** and navigate to **Release Versions** by clicking on link. Now use the following steps to create a release version. Here I used r1 as the release version to indicate it's a initial release of our REST Api.

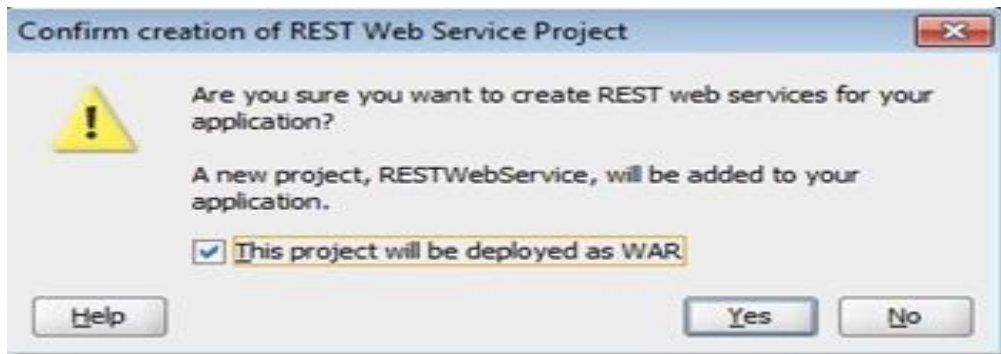


Exposing VO as REST resource

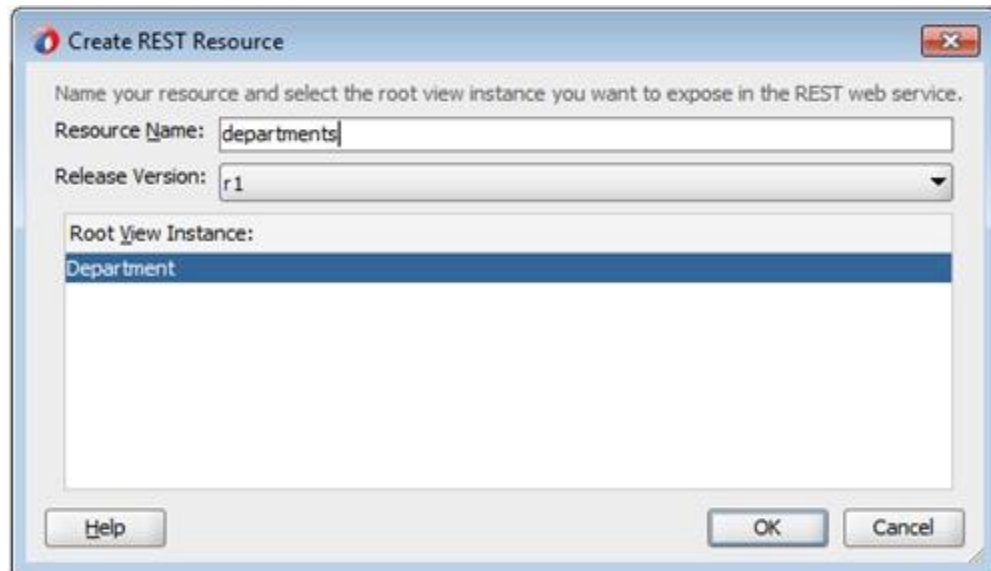
Open **EmployeeAM** and navigate to **Web Service -> REST** and click + icon.



This would ask for the confirmation of new project named **RESTWebService.jpr** in our application which is used to deploy all the REST services created using WAR deployment profile.

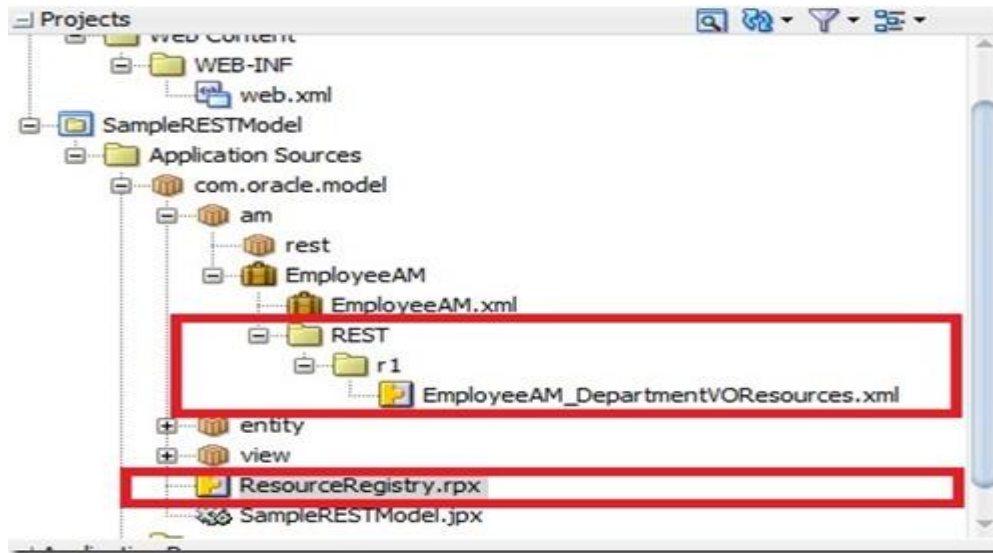


Click **OK** and give the meaningful resource name.



Click **OK** to observe the new set of files/project created because of this action as shown below.





Use following tabs to choose available operations and attributes exposed to consumers.

Representation **Attributes** Custom Methods

Resource Name:

Rowfinder Key:

☐ Canonical Link

☐ Internal

☐ External

☐ Include Discriminator Subtype

Discriminator Name: Value:

Available Operations

☒ Create ☒ Delete ☒ Update

The resource structure will show other VOs as child resource if a View Links is available. Here, we will expose **Employee** as child resource of **Department** by setting check box as shown below.

REST Resources

Resources can expose accessors, attributes and map custom actions.

Application Module: **EmployeeAM** View Object: **DepartmentVO**

Resource:

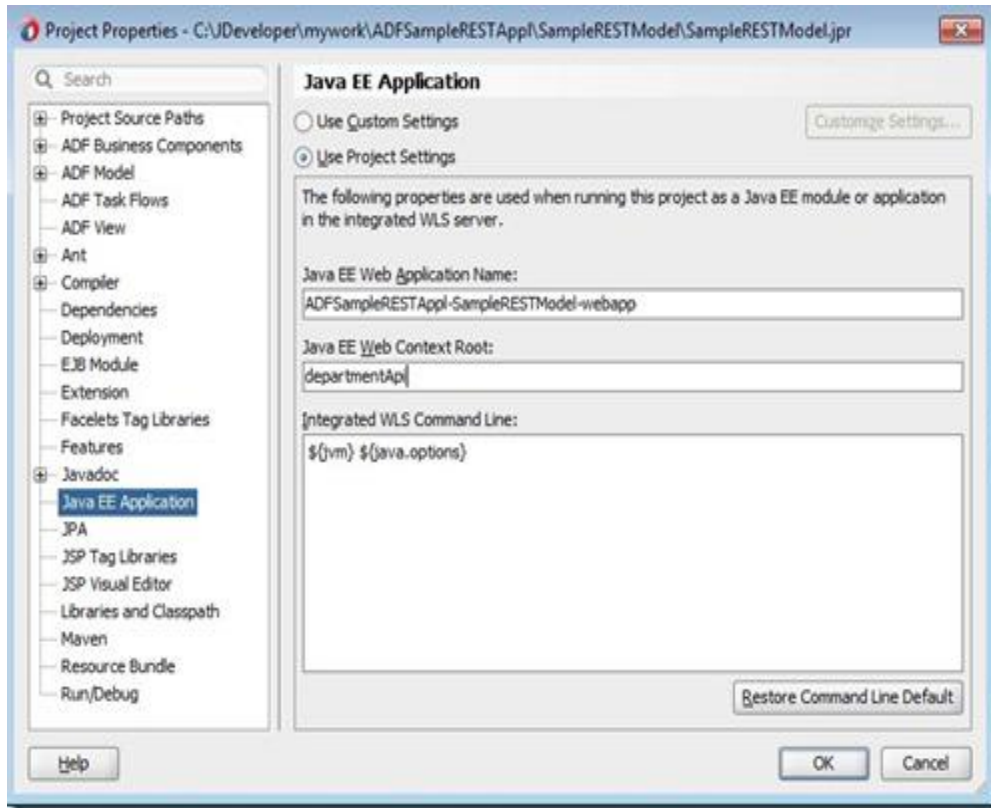
Resource Structure:

☒ **Department**

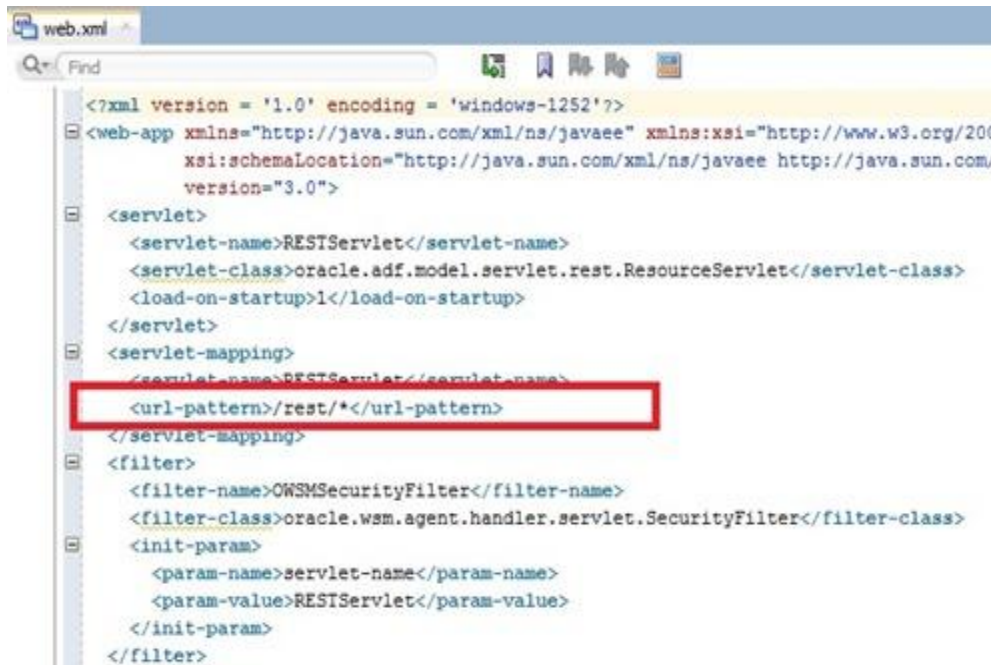
☒ ☐ **Employee**

Deployment

Set the context root of **RESTWebService** project to **departmentApi** as shown below.

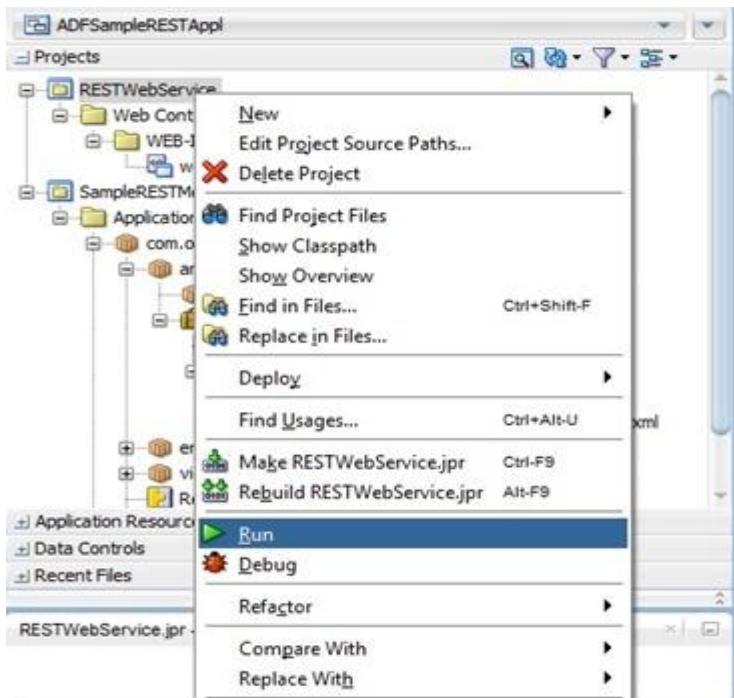


Optionally, we can modify the URL pattern in **web.xml** as shown below.



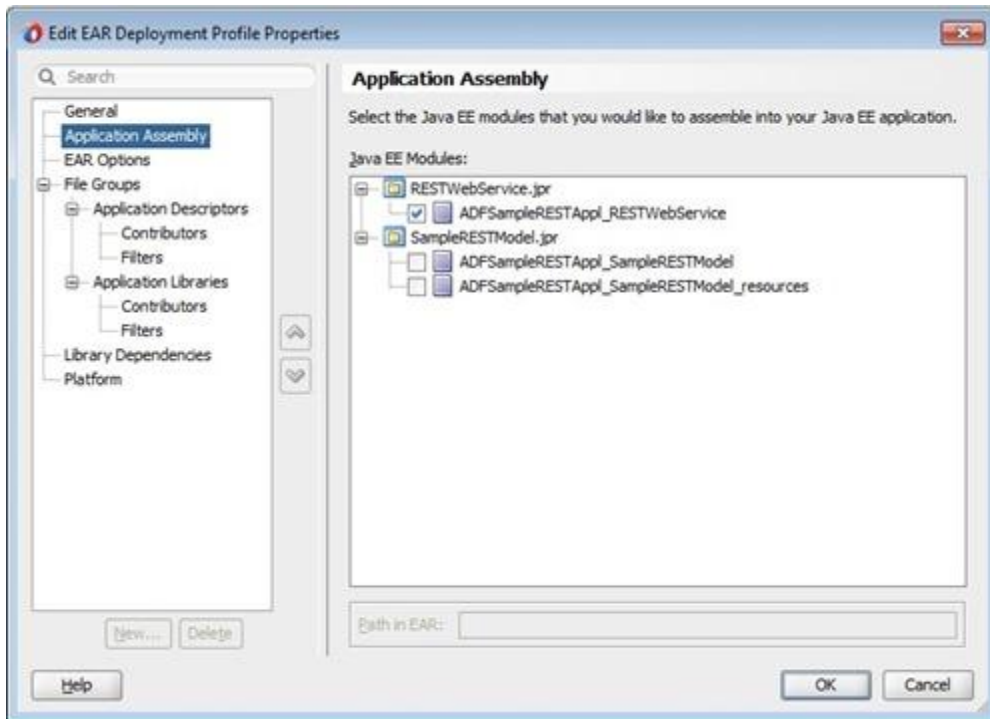
Integrated WLS

Select **RESTWebService** project and click **Run** as shown below.



Standalone WLS

Create EAR profile and include **RESTWebService** project as shown below and deploy to standalone WLS.



Once the deployment is done, we can access the REST service using URL:

`http://<<host>>:<<port>>/<<ContextRoot>>/<<url pattern>>/<<version>>/<<resource name>>`

e.g.: `http://localhost:7001/departmentApi/rest/r1/departments`

Also, we can also use **latest** keyword to access latest version of the resource.

e.g.: `http://localhost:7001/departmentApi/rest/latest/departments`

We can use any of the REST clients to verify different operations using GET, POST, DELETE, PUT, PATCH provided they are exposed to consumers. Following is the sample list of URLs using **departments** and **Employee** child to perform different operations on them. (may not be exhaustive)

Describing Resource – GET

`http://localhost:7001/departmentApi/rest/r1/departments/describe`

Describing Resource Instance – GET

`http://localhost:7001/departmentApi/rest/r1/departments/10/describe`

Querying Departments – GET

`http://localhost:7001/departmentApi/rest/r1/departments`

Querying a particular Department – GET

`http://localhost:7001/departmentApi/rest/r1/departments/{id}`

Creating Department – POST

<http://localhost:7001/departmentApi/rest/r1/departments>

Content-Type: application/vnd.oracle.adf.resourceitem+json

Body:

```
{
  "DepartmentId": 1000,
  "DepartmentName": "Administration",
  "ManagerId": 200,
  "LocationId": 1700
}
```

Deleting a Department – DELETE

`http://localhost:7001/departmentApi/rest/r1/departments/{id}`

Updating a Department – POST

`http://localhost:7001/departmentApi/rest/r1/departments/{id}`

Content-Type: application/vnd.oracle.adf.resourceitem+json

X-HTTP-Method-Override: PATCH

Body: (contains only fields to be modified)

```
{
  "DepartmentName": "Administration-Modified"
}
```

Replacing a Department – PUT

http://localhost:7001/departmentApi/rest/r1/departments/{id}
Content-Type: application/vnd.oracle.adf.resourceitem+json
Body: (Values not sent in body will be set to null)
{
 "DepartmentId":10,
 "DepartmentName": "Administration-Replace",
 "ManagerId": 100
}

Querying Department for a few fields – GET

http://localhost:7001/departmentApi/rest/r1/departments?fields=DepartmentName,ManagerId

Querying a Department using an attribute – GET

http://localhost:7001/departmentApi/rest/r1/departments/{id}?q=DepartmentName=Administration

Querying a Department for only Data – GET

http://localhost:7001/departmentApi/rest/r1/departments?onlyData=true (Does not fetch any links/metadata in response)

Sorting Departments – GET

http://localhost:7001/departmentApi/rest/r1/departments?orderBy=DepartmentName:asc
http://localhost:7001/departmentApi/rest/r1/departments?orderBy=DepartmentName: desc

Limiting the records in Querying Departments – GET

http://localhost:7001/departmentApi/rest/r1/departments?limit=2 (Fetches only 2 records)

Querying Departments from a particular record– GET

http://localhost:7001/departmentApi/rest/r1/departments?offset=2 (Fetches only 2 records)

http://localhost:7001/departmentApi/rest/r1/departments?offset=2&limit=5 (Fetches 5 records starting from 2nd record)

Expanding a Child Resource – GET

http://localhost:7001/departmentApi/rest/r1/departments?expand=Employee (Child Resource Name)

Querying Child Resource - GET

http://localhost:7001/departmentApi/rest/r1/departments/{id}/child/Employee

Querying a particular Child Resource - GET

http://localhost:7001/departmentApi/rest/r1/departments/{id}/child/Employee/{Child Resource Id}

Querying a Child Resource using an attribute – GET

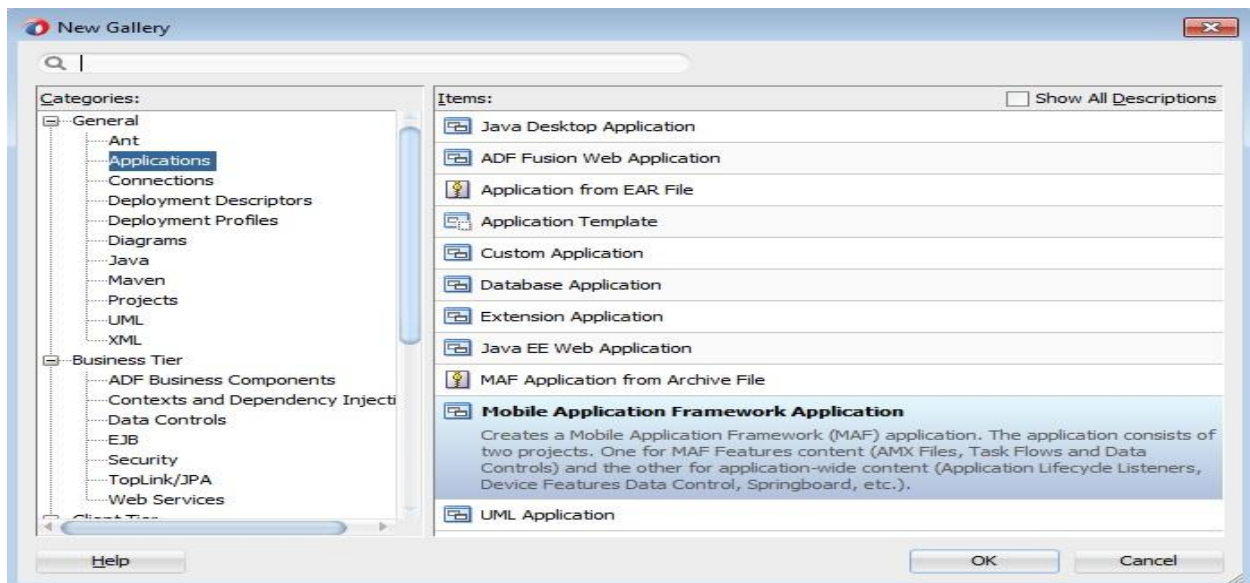
http://localhost:7001/departmentApi/rest/r1/departments/{id}/child/Employee?q=FirstName=Jennifer

Developing Mobile Application

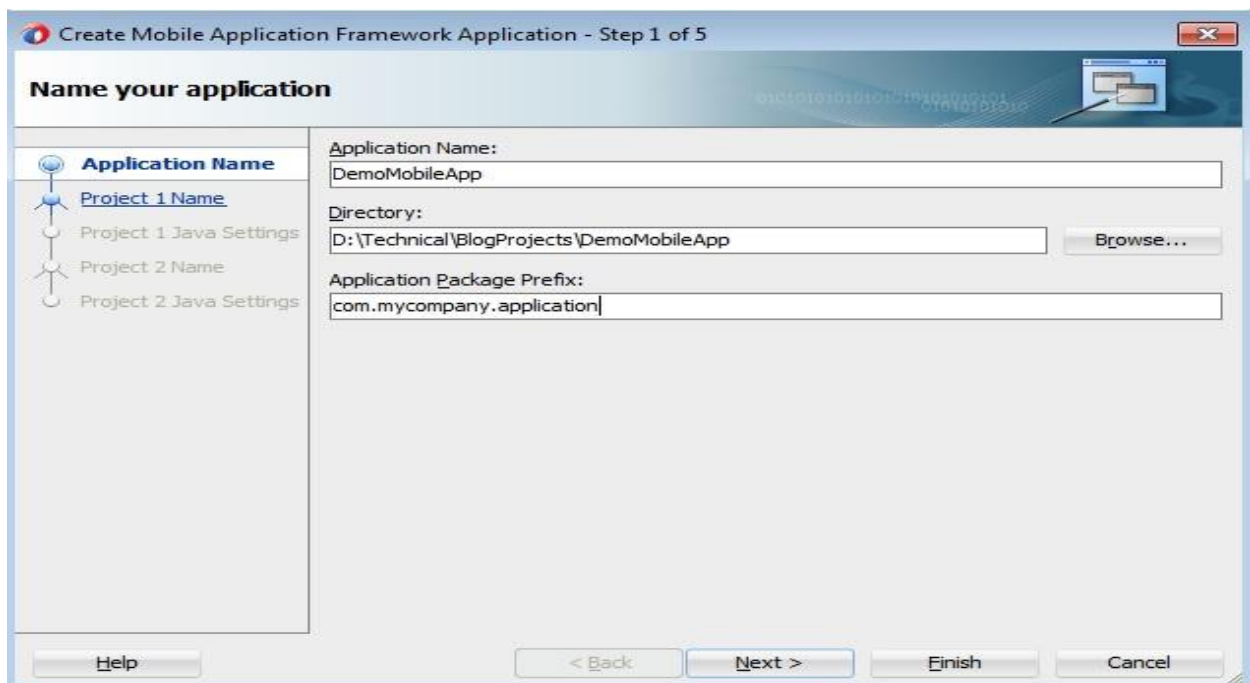
In this section, we will see how to use the above ADF BC REST Service in a MAF application.

Creating MAF Application

As mentioned earlier, we are using MAF 2.2.1 which supports only JDeveloper 12.1.3, so we should install the same with this MAF extension before proceeding further. To create new MAF Application, click **File-> New** and select **Mobile Application Framework Application**.



Enter Application Name, Package prefix and choose the directory to create new application. Click Next.



By default, MAF application gets created with **ApplicationController** and **ViewController** projects. But we can always modify these project names. Now follow the below steps to finish application creation.

Create Mobile Application Framework Application - Step 2 of 5

Name your project

Application Name

Project 1 Name

Project 1 Java Settings

Project 2 Name

Project 2 Java Settings

Project Name:

Directory: [Browse...](#)

Project Features:

HTML and CSS

HyperText Markup Language (HTML) is the standard syntax for publishing hypertext on the World Wide Web.

Java

The Java programming language is a simple object-oriented language designed to meet the challenges of application development in the context of heterogeneous, network-wide distributed environments.

JavaScript

JavaScript is a client-side scripting language implemented as part of web browsers. Combining elements of procedural and object-oriented languages, JavaScript supports enhanced user interfaces and dynamic websites.

Mobile Application Framework

Mobile Application Framework (MAF) adds support for the development of a MAF application.

REST Web Services

[Help](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

Create Mobile Application Framework Application - Step 3 of 5

Configure Java settings

Application Name

Project 1 Name

Project 1 Java Settings

Project 2 Name

Project 2 Java Settings

Your new project starts with a default package, a source root directory, and an output directory.

Default Package:

Java Source Path: [Browse...](#)

Output Directory: [Browse...](#)

[Help](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

Create Mobile Application Framework Application - Step 4 of 5

Name your project

Application Name
Project 1 Name
Project 1 Java Settings
Project 2 Name
Project 2 Java Settings

Project Name:

Directory:

Project Features:

HTML and CSS
HyperText Markup Language (HTML) is the standard syntax for publishing hypertext on the World Wide Web.

Java
The Java programming language is a simple object-oriented language designed to meet the challenges of application development in the context of heterogeneous, network-wide distributed environments.

JavaScript
JavaScript is a client-side scripting language implemented as part of web browsers. Combining elements of procedural and object-oriented languages, JavaScript supports enhanced user interfaces and dynamic websites.

Mobile Application Framework
Mobile Application Framework (MAF) adds support for the development of a MAF application.

REST Web Services

Create Mobile Application Framework Application - Step 5 of 5

Configure Java settings

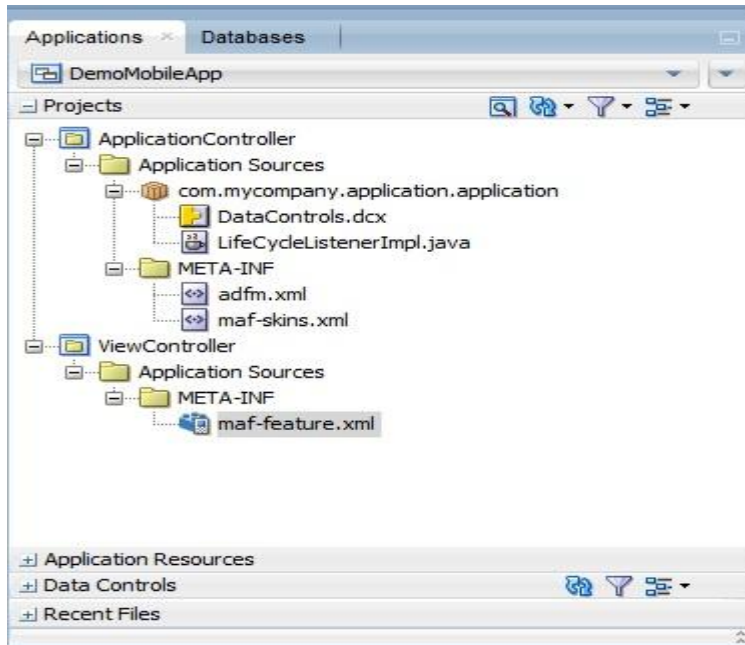
Application Name
Project 1 Name
Project 1 Java Settings
Project 2 Name
Project 2 Java Settings

Your new project starts with a default package, a source root directory, and an output directory.

Default Package:

Java Source Path:

Output Directory:

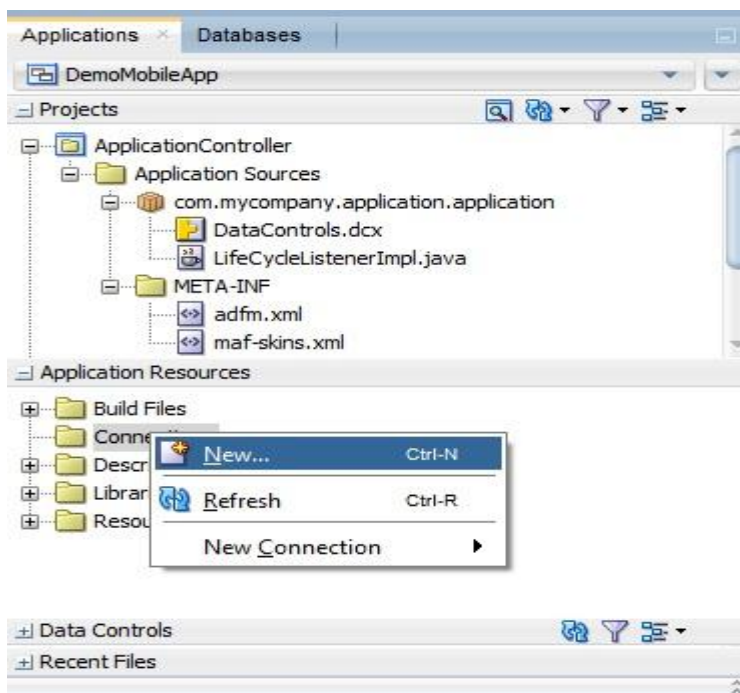


Creating REST Connection

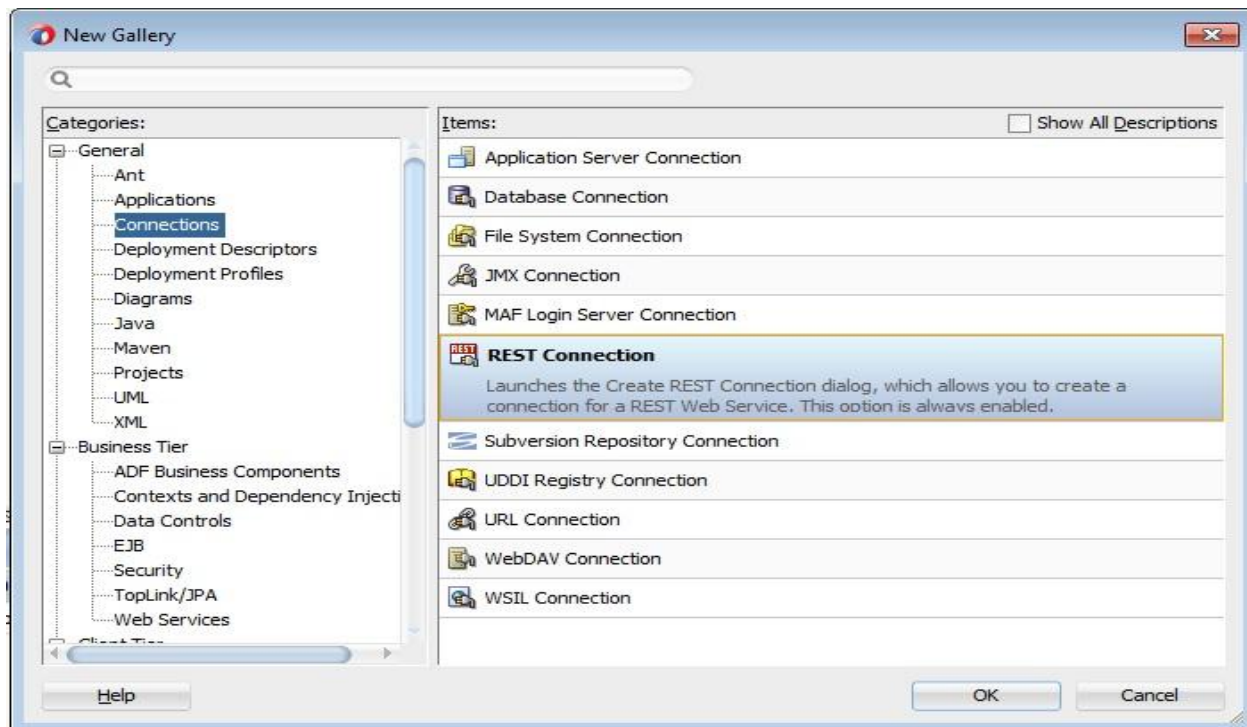
MAF provides two options to connect to REST services.

- REST Webservice Data control
- RestServiceAdapter class

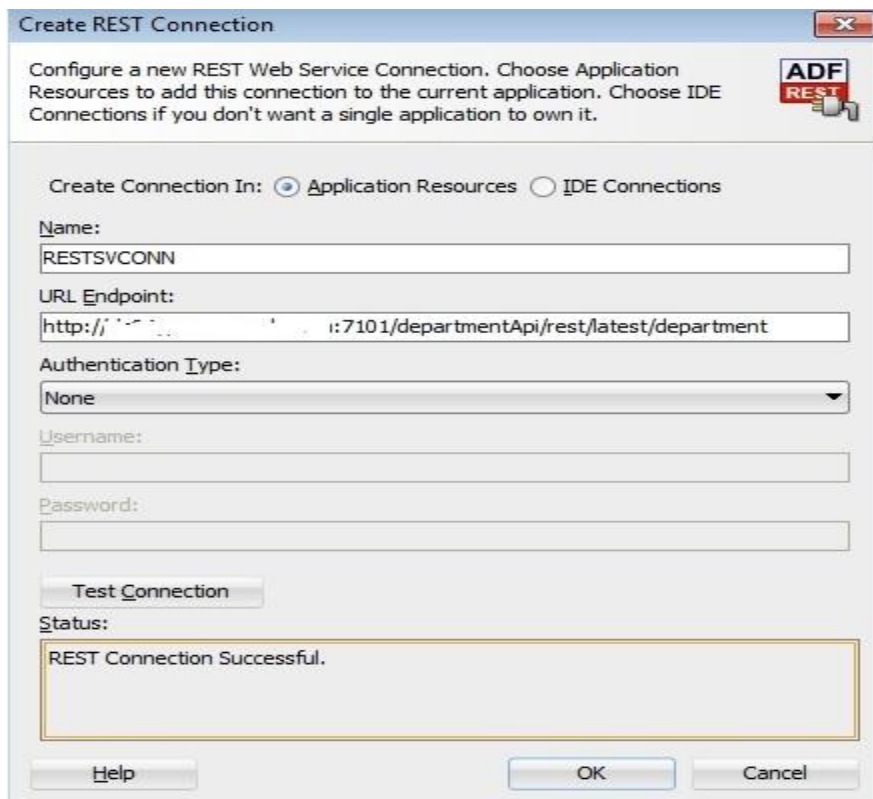
Here in this tutorial, we will make use of **RestServiceAdapter** class to invoke ADF BC REST services which makes use of a REST connection which need to be created. To do that, go to **Application Resources -> Connections** and select **New** as shown below.



Select **General -> Connections -> REST Connection**.

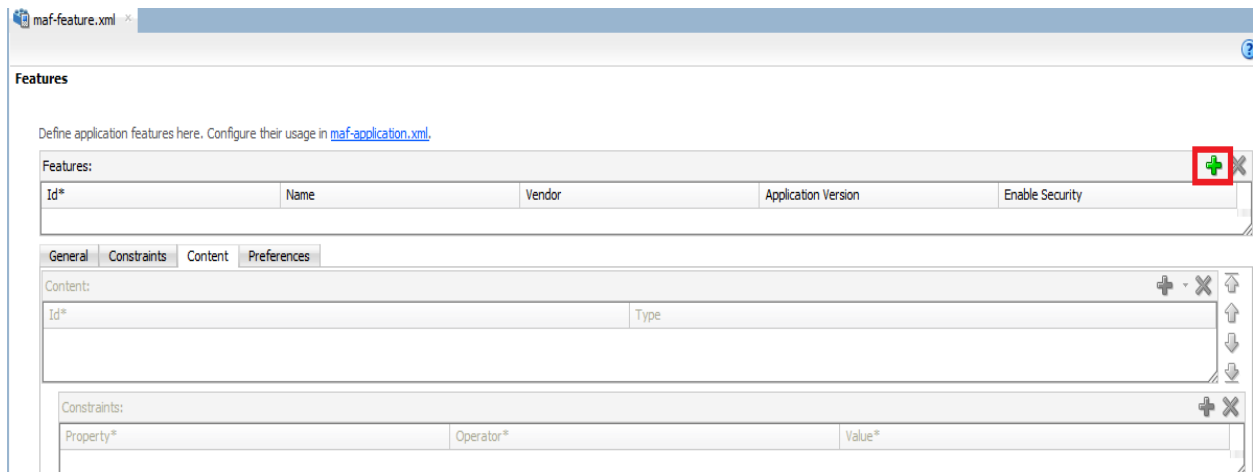


Give Name and fully qualified REST service URL. Use **Test Connection** to confirm connectivity.

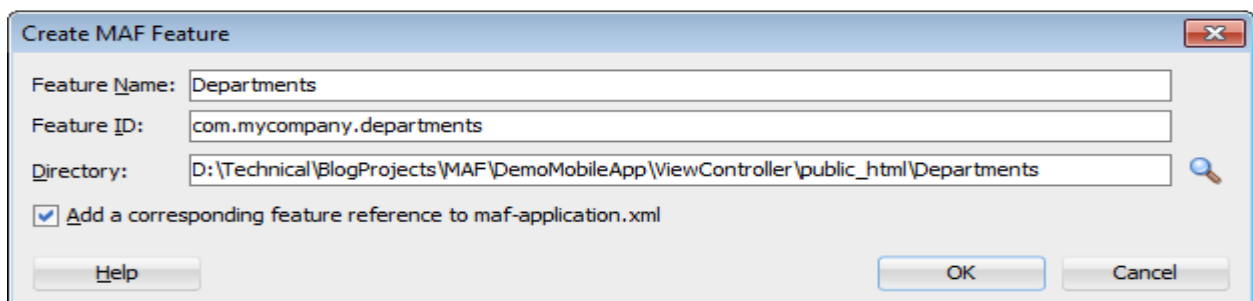


Creating Application Feature

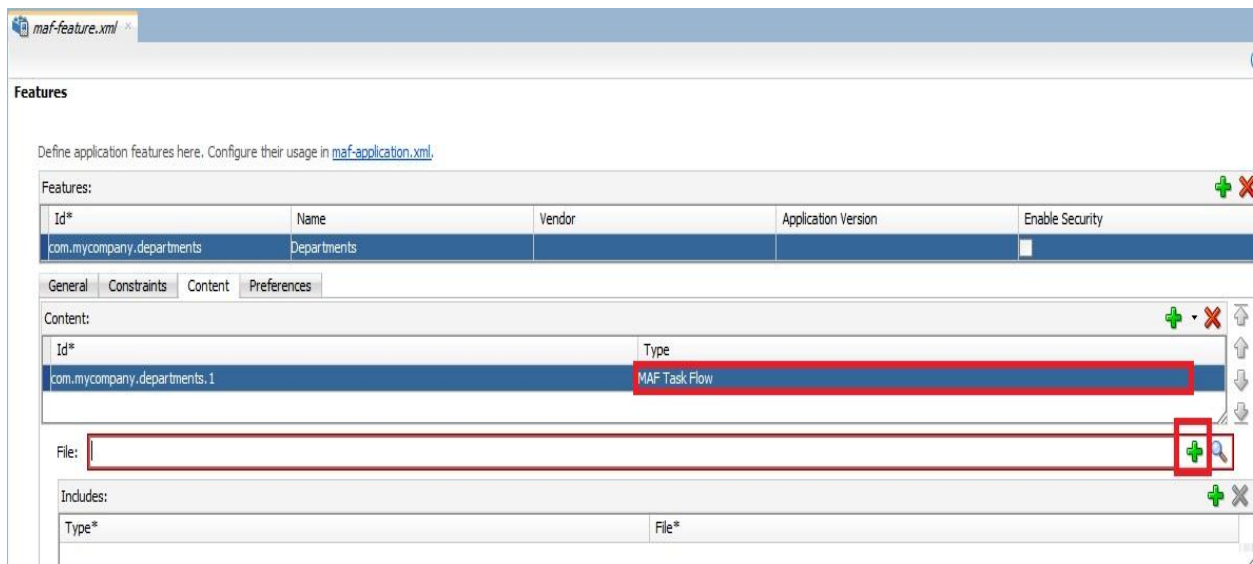
We implement our use case using single application feature. Open **maf-feature.xml** in **ViewController -> META-INF** and click + icon to create new feature.



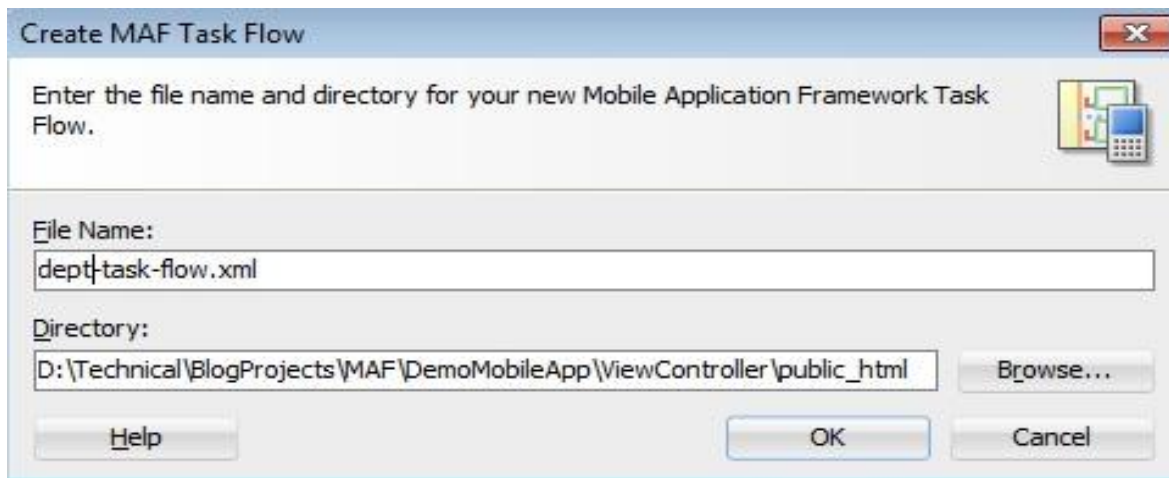
Provide feature name and id and click **OK**.



Go to **Content** tab, select type as **MAF Task Flow** and click + to create new task flow.



Provide task flow name and click OK.



The dialog box titled "Create MAF Task Flow" prompts the user to enter the file name and directory for a new Mobile Application Framework Task Flow. It includes a "File Name" field with the text "dept-task-flow.xml", a "Directory" field with the path "D:\Technical\BlogProjects\MAF\DemoMobileApp\ViewController\public_html", and buttons for "Help", "OK", "Cancel", and "Browse...".

Create MAF Task Flow

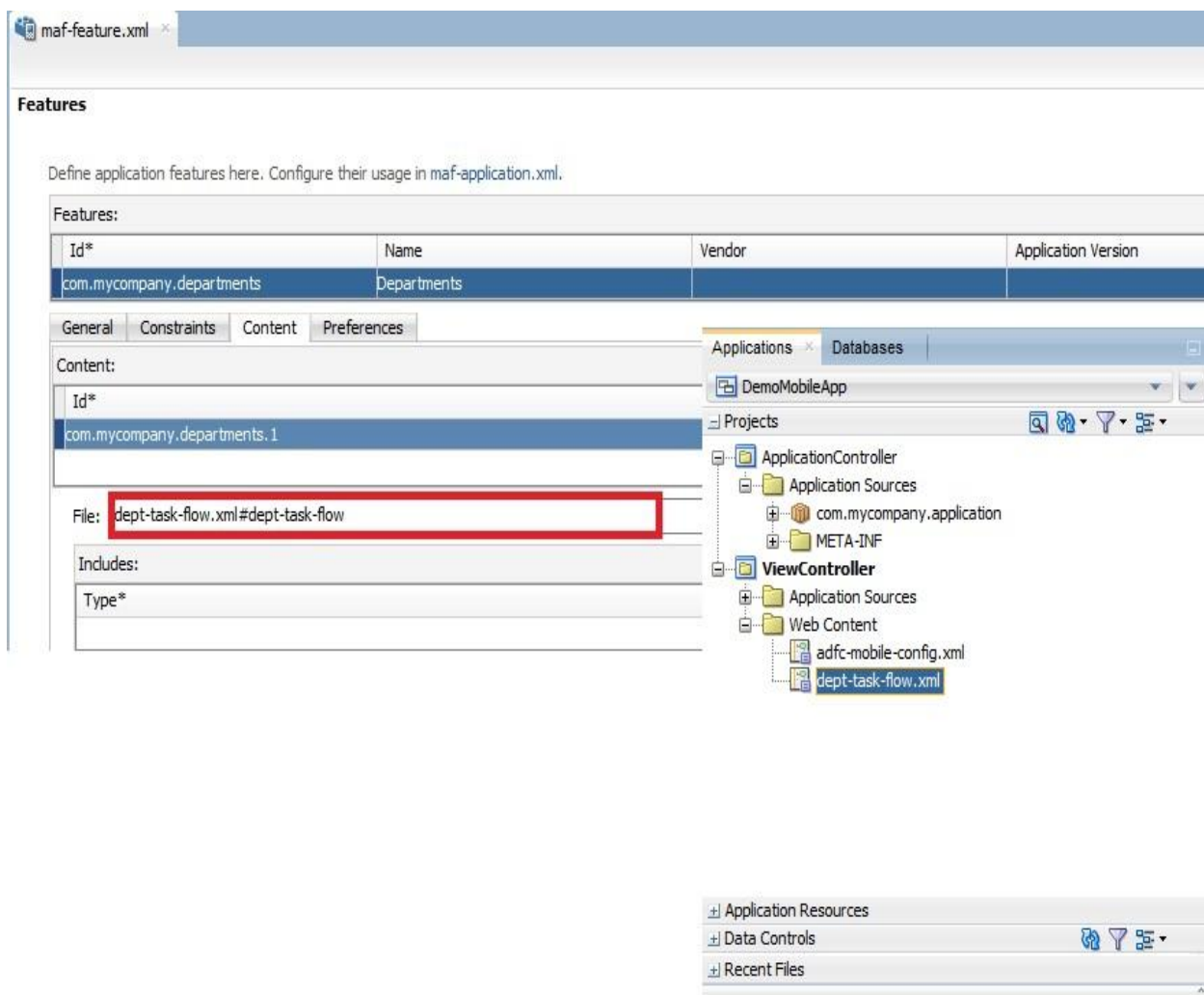
Enter the file name and directory for your new Mobile Application Framework Task Flow.

File Name:
dept-task-flow.xml

Directory:
D:\Technical\BlogProjects\MAF\DemoMobileApp\ViewController\public_html

Help OK Cancel Browse...

Now we can observe new task flow is created in **ViewController** -> **Web Content**.



The screenshot shows the MAF IDE interface. The "Features" tab is active, displaying a table of application features. The "Content" tab is also visible, showing the "File" field with the value "dept-task-flow.xml#dept-task-flow" highlighted in red. The "Projects" pane on the right shows the project structure, including "ApplicationController", "Application Sources", "META-INF", "ViewController", "Application Sources", "Web Content", "adfc-mobile-config.xml", and "dept-task-flow.xml".

maf-feature.xml

Features

Define application features here. Configure their usage in maf-application.xml.

Features:

Id*	Name	Vendor	Application Version
com.mycompany.departments	Departments		

General Constraints Content Preferences

Content:

Id*

com.mycompany.departments.1

File: dept-task-flow.xml#dept-task-flow

Includes:

Type*

Applications Databases

DemoMobileApp

Projects

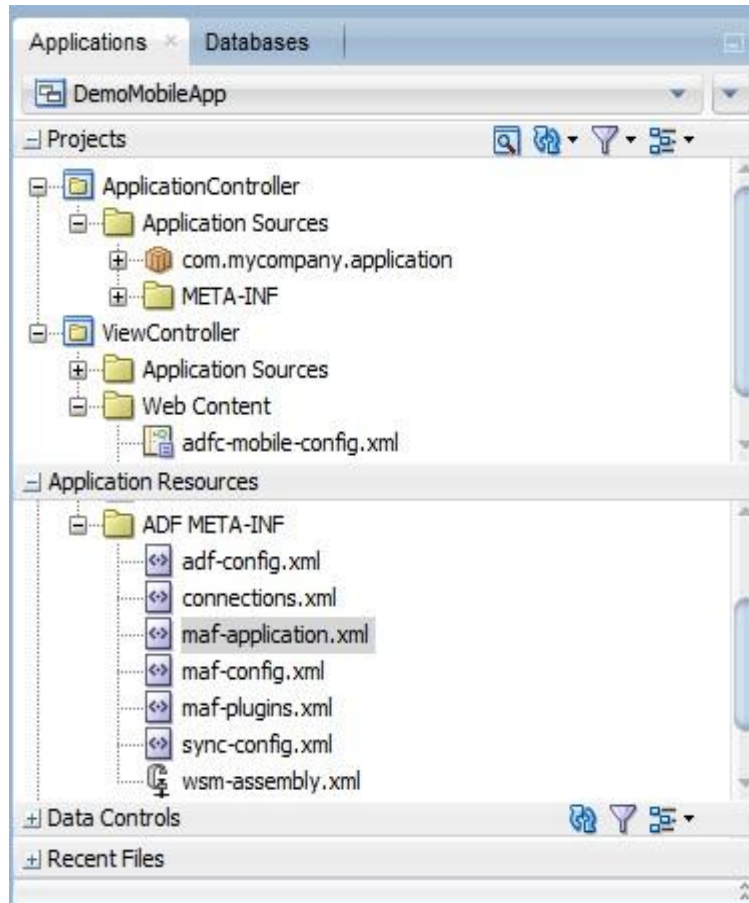
- ApplicationController
 - Application Sources
 - com.mycompany.application
 - META-INF
- ViewController
 - Application Sources
 - Web Content
 - adfc-mobile-config.xml
 - dept-task-flow.xml

Application Resources

Data Controls

Recent Files

Since we set 'Add corresponding feature reference to maf-application.xml' while creating feature, this feature reference will be automatically created in **maf-application.xml**. This file contains information about all application features, Cordova plug-ins, springboard settings, security and preferences of a MAF application and available in **Application Resources -> Descriptors -> ADF META-INF**.



maf-feature.xml x maf-application.xml x			
Application			
Plugins			
Feature References	Feature References:		
Preferences			
Security			
	Feature Id*	Show on Navigation Bar	Show on Springboard
	com.mycompany.departments	<default> (true)	<default> (true)

Code

We have to create java beans for representing **Department** and **Employee** information. These classes should be created with attribute names matching the actual REST resource attributes.

We can get metadata related to any ADFBC REST Resource in following way:

`http://<<host>>:<<port>>/<<context-root>>/rest/<<release>>/<<resourcename>>/describe`

In our scenario, the resource URL for **departments** is:

`http://<<host>>:<<port>>/departmentApi/rest/r1/departments`

The child resource URL for **Employee** is:

`http://<<host>>:<<port>>/departmentApi/rest/r1/departments/{id}child/Employee`

So use the following URLs to get metadata and create Employee and Department classes as below.

`http://<<host>>:<<port>>/departmentApi/rest/r1/departments/describe`

`http://<<host>>:<<port>>/departmentApi/rest/r1/departments/{id}child/Employee/describe`

Employee.Java

```
package com.mycompany.mobile.entity;
```

```
import java.util.Date;
```

```
import oracle.adfmf.java.beans.PropertyChangeListener;
```

```
import oracle.adfmf.java.beans.PropertyChangeSupport;
```

```
public class Employee
```

```
{
```

```
    private long employeeId;
```

```
    private String firstName;
```

```
    private String lastName;
```

```
    private String email;
```

```
    private String phoneNumber;
```

```
    private Date hireDate;
```

```
    private String jobId;
```

```
    private Double salary;
```

```
    private Double commissionPct;
```

```
    private long managerId;
```

```
    private long departmentId;
```

```
    private String isActive;
```

```
    private PropertyChangeSupport propertyChangeSupport = new PropertyChangeSupport(this);
```

```
    public void setEmployeeId(long employeeId)
```

```
    {
```

```
        long oldEmployeeId = this.employeeId;
```

```
        this.employeeId = employeeId;
```

```
        propertyChangeSupport.firePropertyChange("employeeId", oldEmployeeId, employeeId);
```

```
    }
```

```
    public long getEmployeeId()
```

```
    {
```

```
        return employeeId;
```

```
    }
```

```
public void setFirstName(String firstName)
{
    String oldFirstName = this.firstName;
    this.firstName = firstName;
    propertyChangeSupport.firePropertyChange("firstName", oldFirstName, firstName);
}

public String getFirstName()
{
    return firstName;
}

public void setLastName(String lastName)
{
    String oldLastName = this.lastName;
    this.lastName = lastName;
    propertyChangeSupport.firePropertyChange("lastName", oldLastName, lastName);
}

public String getLastName()
{
    return lastName;
}

public void setEmail(String email)
{
    String oldEmail = this.email;
    this.email = email;
    propertyChangeSupport.firePropertyChange("email", oldEmail, email);
}

public String getEmail()
{
    return email;
}

public void setPhoneNumber(String phoneNumber)
{
    String oldPhoneNumber = this.phoneNumber;
    this.phoneNumber = phoneNumber;
    propertyChangeSupport.firePropertyChange("phoneNumber", oldPhoneNumber, phoneNumber);
}

public String getPhoneNumber()
{
    return phoneNumber;
}

public void setHireDate(Date hireDate)
{
    Date oldHireDate = this.hireDate;
    this.hireDate = hireDate;
    propertyChangeSupport.firePropertyChange("hireDate", oldHireDate, hireDate);
}

public Date getHireDate()
{

```

```
        return hireDate;
    }

    public void addPropertyChangeListener(PropertyChangeListener l)
    {
        propertyChangeSupport.addPropertyChangeListener(l);
    }

    public void setJobId(String jobId) {
        String oldJobId = this.jobId;
        this.jobId = jobId;
        propertyChangeSupport.firePropertyChange("jobId", oldJobId, jobId);
    }

    public String getJobId() {
        return jobId;
    }

    public void setSalary(Double salary) {
        Double oldSalary = this.salary;
        this.salary = salary;
        propertyChangeSupport.firePropertyChange("salary", oldSalary, salary);
    }

    public Double getSalary() {
        return salary;
    }

    public void setCommissionPct(Double commissionPct) {
        Double oldCommissionPct = this.commissionPct;
        this.commissionPct = commissionPct;
        propertyChangeSupport.firePropertyChange("commissionPct", oldCommissionPct, commissionPct);
    }

    public Double getCommissionPct() {
        return commissionPct;
    }

    public void setManagerId(long managerId) {
        long oldManagerId = this.managerId;
        this.managerId = managerId;
        propertyChangeSupport.firePropertyChange("managerId", oldManagerId, managerId);
    }

    public long getManagerId() {
        return managerId;
    }

    public void setDepartmentId(long departmentId) {
        long oldDepartmentId = this.departmentId;
        this.departmentId = departmentId;
        propertyChangeSupport.firePropertyChange("departmentId", oldDepartmentId, departmentId);
    }

    public long getDepartmentId() {
        return departmentId;
    }
}
```

```

public void setIsactive(String isactive) {
    String oldIsactive = this.isactive;
    this.isactive = isactive;
    propertyChangeSupport.firePropertyChange("isactive", oldIsactive, isactive);
}

public String getIsactive() {
    return isactive;
}

public void setPropertyChangeSupport(PropertyChangeSupport propertyChangeSupport) {
    PropertyChangeSupport oldPropertyChangeSupport = this.propertyChangeSupport;
    this.propertyChangeSupport = propertyChangeSupport;
    propertyChangeSupport.firePropertyChange("propertyChangeSupport", oldPropertyChangeSupport,
        propertyChangeSupport);
}

public PropertyChangeSupport getPropertyChangeSupport() {
    return propertyChangeSupport;
}

public void removePropertyChangeListener(PropertyChangeListener l)
{
    propertyChangeSupport.removePropertyChangeListener(l);
}
}

```

Department.java

```
package com.mycompany.mobile.entity;
```

```
import oracle.adfmf.java.beans.PropertyChangeListener;
import oracle.adfmf.java.beans.PropertyChangeSupport;
```

```

public class Department
{
    private long departmentId;
    private String departmentName;
    private long managerId;
    private long locationId;

    private Employee[] employee;
    private PropertyChangeSupport propertyChangeSupport = new PropertyChangeSupport(this);

    public void setDepartmentId(long departmentId) {
        long oldDepartmentId = this.departmentId;
        this.departmentId = departmentId;
        propertyChangeSupport.firePropertyChange("departmentId", oldDepartmentId, departmentId);
    }

    public long getDepartmentId() {
        return departmentId;
    }

    public void setDepartmentName(String departmentName) {
        String oldDepartmentName = this.departmentName;
        this.departmentName = departmentName;
        propertyChangeSupport.firePropertyChange("departmentName", oldDepartmentName, departmentName);
    }
}

```

```

public String getDepartmentName() {
    return departmentName;
}

public void setManagerId(long managerId) {
    long oldManagerId = this.managerId;
    this.managerId = managerId;
    propertyChangeSupport.firePropertyChange("managerId", oldManagerId, managerId);
}

public long getManagerId() {
    return managerId;
}

public void setLocationId(long locationId) {
    long oldLocationId = this.locationId;
    this.locationId = locationId;
    propertyChangeSupport.firePropertyChange("locationId", oldLocationId, locationId);
}

public long getLocationId() {
    return locationId;
}

public void setEmployee(Employee[] employee) {
    Employee[] oldEmployee = this.employee;
    this.employee = employee;
    propertyChangeSupport.firePropertyChange("employee", oldEmployee, employee);
}

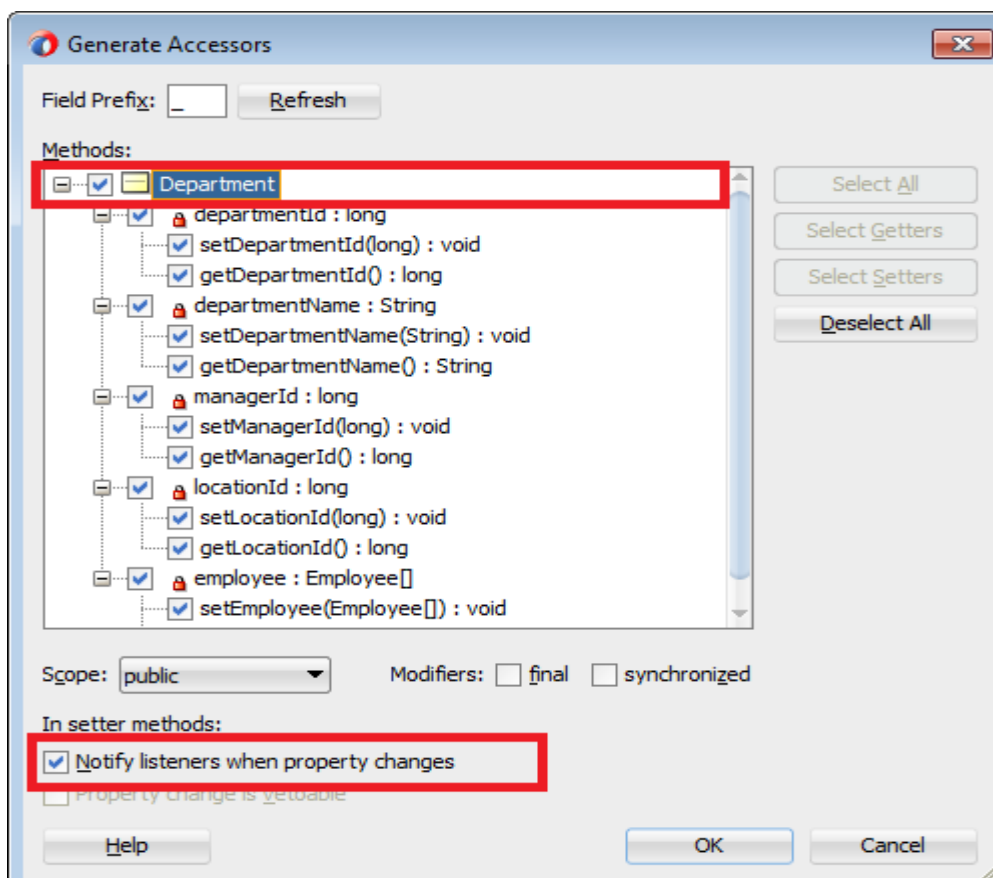
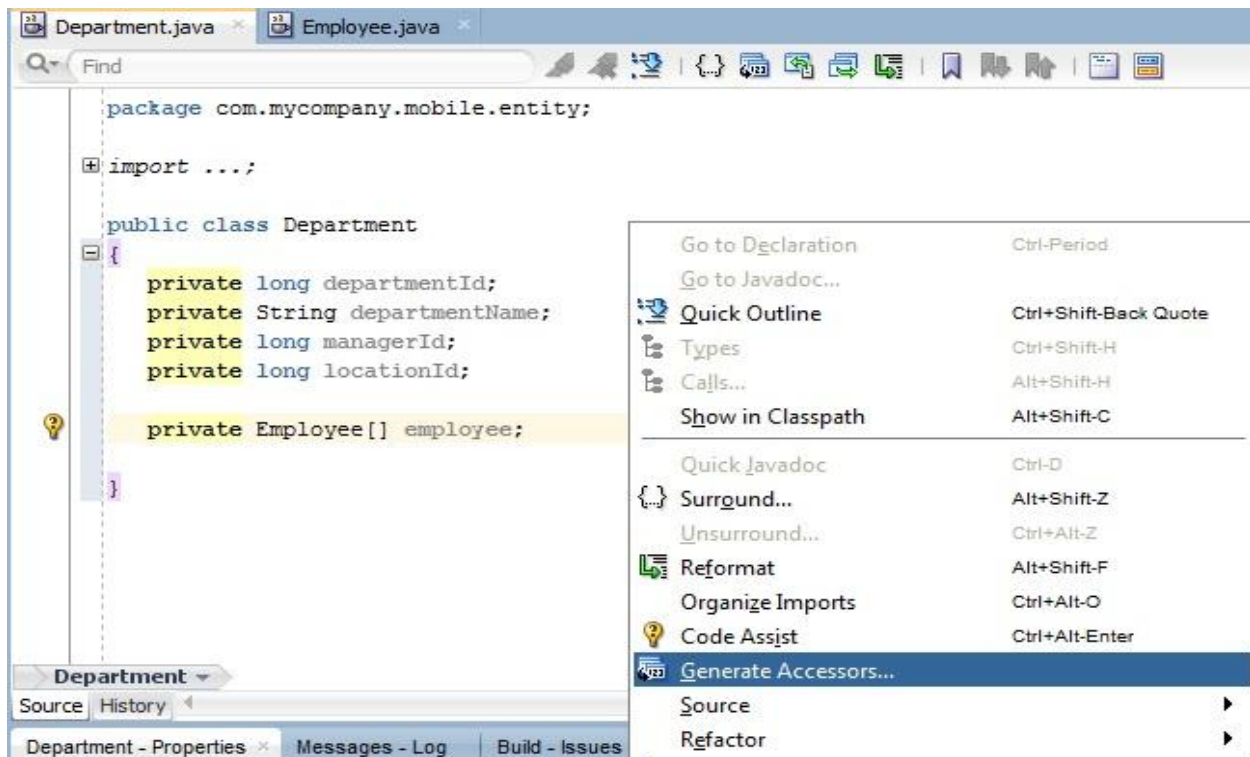
public Employee[] getEmployee() {
    return employee;
}

public void addPropertyChangeListener(PropertyChangeListener l) {
    propertyChangeSupport.addPropertyChangeListener(l);
}

public void removePropertyChangeListener(PropertyChangeListener l) {
    propertyChangeSupport.removePropertyChangeListener(l);
}
}

```

We can generate attribute accessors for attributes as shown below.



ItemsList.java

When we access ADF BC resource, all resource items of parent resource are returned as JSON array. This class is used for de-serialization of these resource items and we made it generic so that it can be used with any parent resource.

```
package com.mycompany.mobile.common;

import java.lang.reflect.Array;

import oracle.adfml.java.beans.PropertyChangeListener;
import oracle.adfml.java.beans.PropertyChangeSupport;

//utility class to store items[] returned in ADF BC service
public class ItemsList<T>
{
    private T[] items;
    private int count = 0;
    private boolean hasMore = true;
    private int offset = 0;
    private PropertyChangeSupport propertyChangeSupport = new PropertyChangeSupport(this);

    //used to append the list
    public void append(ItemsList<T> newList, Class<T> classType)
    {
        if (newList != null && newList.getItems() != null)
        {
            //if no items present
            if (getItems() == null)
            {
                //add all attributes from new list to member attribute
                setItems(newList.getItems());
                setCount(newList.getCount());
                setHasMore(newList.isHasMore());
                setOffset(newList.getOffset());
                return;
            }

            int newItemLength = newList.getItems().length;
            int currentItemLength = getItems().length;

            //generic array type creation of combined length (existing item length + new items length)
            @SuppressWarnings("unchecked")
            T[] concatenatedItems =
                (T[]) Array.newInstance(classType, newItemLength + currentItemLength);

            //copy the contents to intermediate array as we can't extend the current array
            System.arraycopy(getItems(), 0, concatenatedItems, 0, currentItemLength);
            System.arraycopy(newList.getItems(), 0, concatenatedItems, currentItemLength,
                newItemLength);

            //set the member attributes with concatenated array and other attributes
            setItems(concatenatedItems);
            setCount(newList.getCount());
            setHasMore(newList.isHasMore());
            setOffset(newList.getOffset());
        }
    }
}
```

```

public void setItems(T[] items)
{
    T[] oldItems = this.items;
    this.items = items;
    propertyChangeSupport.firePropertyChange("items", oldItems, items);
}

public T[] getItems()
{
    return items;
}

public void setCount(int count)
{
    int oldCount = this.count;
    this.count = count;
    propertyChangeSupport.firePropertyChange("count", oldCount, count);
}

public int getCount()
{
    return count;
}

public void setHasMore(boolean hasMore)
{
    boolean oldHasMore = this.hasMore;
    this.hasMore = hasMore;
    propertyChangeSupport.firePropertyChange("hasMore", oldHasMore, hasMore);
}

public boolean isHasMore()
{
    return hasMore;
}

public void setOffset(int offset)
{
    int oldOffset = this.offset;
    this.offset = offset;
    propertyChangeSupport.firePropertyChange("offset", oldOffset, offset);
}

public int getOffset()
{
    return offset;
}

public int getEffOffset()
{
    return getCount() + getOffset();
}

public void addPropertyChangeListener(PropertyChangeListener l)
{
    propertyChangeSupport.addPropertyChangeListener(l);
}

```

```

    public void removePropertyChangeListener(PropertyChangeListener l)
    {
        propertyChangeSupport.removePropertyChangeListener(l);
    }
}

```

CmnRestAdapter.java

This class contains utility methods to initialize **RestServiceAdapter** and for initiating **GET, PUT, POST, PATCH** and **DELETE** requests for a REST resource. This also contains utility methods to de-serialize JSON response using **JSONBeanSerializationHelper** class (available in MAF).

Observe usage of REST connection **RESTSVCCONN** i.e. created above and media types used by ADF BC REST resources.

```

package com.mycompany.mobile.common;

```

```

import java.lang.reflect.Array;
import java.util.Map;
import oracle.adfmf.dc.ws.rest.RestServiceAdapter;
import oracle.adfmf.framework.api.JSONBeanSerializationHelper;
import oracle.adfmf.framework.api.Model;
import oracle.adfmf.framework.exception.AdfException;
import oracle.adfmf.json.JSONArray;
import oracle.adfmf.json.JSONObject;
import oracle.adfmf.util.Utility;

```

```

//super class having all utility methods to perform service calls
public class CmnRestAdapter
{
    public static RestServiceAdapter restAdapter = Model.createRestServiceAdapter();

    //connection name
    public static String connName = "RESTSVCCONN";
    public static String connURL;

    //media types
    public static final String jsonRsrcItemContentType =
        "application/vnd.oracle.adf.resourceitem+json";
    public static final String jsonRsrcCollContentType =
        "application/vnd.oracle.adf.resourcecollection+json";

    //HTTP PATCH
    public static final String REQUEST_TYPE_PATCH = "PATCH";

    //HTTP headers
    public static final String CONTENT_TYPE = "Content-Type";

    //initilize the RestServiceAdapter with conn
    public static RestServiceAdapter initializeAdapter()
    {
        // Clear any previously set request properties, if any
        restAdapter.clearRequestProperties();
    }
}

```

```

//Set the connection name
restAdapter.setConnectionName(connName);

// Specify the number of retries
restAdapter.setRetryLimit(1);
return restAdapter;
}

//method to invoke service accepting HTTP method
public static String invokeHTTPMethod(String httpMethod, String requestURI, String payload,
                                     Map<String, String> reqHeaders)
{
    String response = "";

    initializeAdapter();

    //set http method
    restAdapter.setRequestType((httpMethod.equals(REQUEST_TYPE_PATCH)?
                               RestServiceAdapter.REQUEST_TYPE_POST: httpMethod));

    if (reqHeaders != null && !reqHeaders.isEmpty())
    {
        reqHeaders.forEach((k, v) -> restAdapter.addRequestProperty(k, v));
        if (reqHeaders.get(CONTENT_TYPE) == null)
        {
            //set content type
            restAdapter.addRequestProperty(CONTENT_TYPE, jsonRsrcItemContentType);
        }
    }
    else
    {
        //set content type
        restAdapter.addRequestProperty(CONTENT_TYPE, jsonRsrcItemContentType);
    }

    //set request URI
    restAdapter.setRequestURI(requestURI);

    try
    {
        switch (httpMethod)
        {
            case RestServiceAdapter.REQUEST_TYPE_GET:
                response = restAdapter.send("");
                break;
            case RestServiceAdapter.REQUEST_TYPE_POST:
                response = restAdapter.send(payload);
                break;
            case RestServiceAdapter.REQUEST_TYPE_PUT:
                response = restAdapter.send(payload);
                break;
            case RestServiceAdapter.REQUEST_TYPE_DELETE:
                response = restAdapter.send("");
                break;
            case REQUEST_TYPE_PATCH:
                //tunnel the request as PATCH
                restAdapter.addRequestProperty("X-HTTP-Method-Override", "PATCH");
                response = restAdapter.send(payload);
        }
    }
}

```

```

        break;
    }
}
catch (Exception e)
{
    //can give different error messages based on restAdapter.getResponseStatus()
    //here we are throwing generic error message
    String errMsg = (Utility.isEmpty(e.getCause().getMessage()))? e.getMessage(): e.getCause().getMessage();
    throw new AdfException(errMsg, AdfException.ERROR);
}
return response;
}

//GET method
public static String doGET(String requestURI, Map<String, String> reqHeaders)
{
    return invokeHTTPMethod(RestServiceAdapter.REQUEST_TYPE_GET, requestURI, null, reqHeaders);
}

//POST method
public static String doPOST(String requestURI, String payload,
                           Map<String, String> reqHeaders)
{
    return invokeHTTPMethod(RestServiceAdapter.REQUEST_TYPE_POST, requestURI, payload, reqHeaders);
}

//PUT method
public static String doPUT(String requestURI, String payload,
                          Map<String, String> reqHeaders)
{
    return invokeHTTPMethod(RestServiceAdapter.REQUEST_TYPE_PUT, requestURI, payload, reqHeaders);
}

//DELETE method
public static String doDELETE(String requestURI, Map<String, String> reqHeaders)
{
    return invokeHTTPMethod(RestServiceAdapter.REQUEST_TYPE_DELETE, requestURI, null, reqHeaders);
}

//PATCH method
public static String doPATCH(String requestURI, String payload,
                             Map<String, String> reqHeaders)
    throws Exception
{
    return invokeHTTPMethod(REQUEST_TYPE_PATCH, requestURI, payload, reqHeaders);
}

//Utility method to deserialize JSON response into specific java class
public static <T> ItemsList<T> deserializeToItemsList(String response, Class<T> classType)
    throws Exception
{
    if (Utility.isEmpty(response))
    {
        return null;
    }

    ItemsList<T> itemList = new ItemsList<T>();

```

```

//deserialize total JSON response
JSONObject root =
    (JSONObject) JSONBeanSerializationHelper.fromJSON(JSONObject.class, response);

//deserialize items[]
JSONArray itemArray = (JSONArray) root.get("items");
int size = itemArray.length();

//initialize record count and all
itemList.setHasMore(root.getBoolean("hasMore"));
itemList.setCount(root.getInt("count"));
itemList.setOffset(root.getInt("offset"));

//create generic array with size of JSONArray
@SuppressWarnings("unchecked")
T[] itemTypeArray = (T[]) Array.newInstance(classType, size);

for (int i = 0; i < size; i++)
{
    //insert element into generic array
    T elem =
        classType.cast(JSONBeanSerializationHelper.fromJSON(classType,
                                                                itemArray.getJSONObject(i)));
    itemTypeArray[i] = elem;
}

//set items
itemList.setItems(itemTypeArray);
return itemList;
}
}

```

DepartmentDC.java

This class is used to expose a Data Control and contains a method to fetch list of departments with associated Employees. We use **expand** notation to get list of employees basically a child resource.

Please note that, the URL we set for **RestServiceAdapter** using **setRequestURI** is appended with actual URL mentioned for REST Connection in runtime.

```

package com.mycompany.mobile.dc;

import com.mycompany.mobile.common.CmnRestAdapter;
import com.mycompany.mobile.common.ItemsList;
import com.mycompany.mobile.entity.Department;

import oracle.adfmf.framework.exception.AdfException;

public class DepartmentDC
{
    public Department[] getDepartments()
    {
        ItemsList<Department> deptList = null;

        //make GET request to get list of departments along with associated employees using expand
        String response = CmnRestAdapter.doGET("?expand=Employee", null);
        try

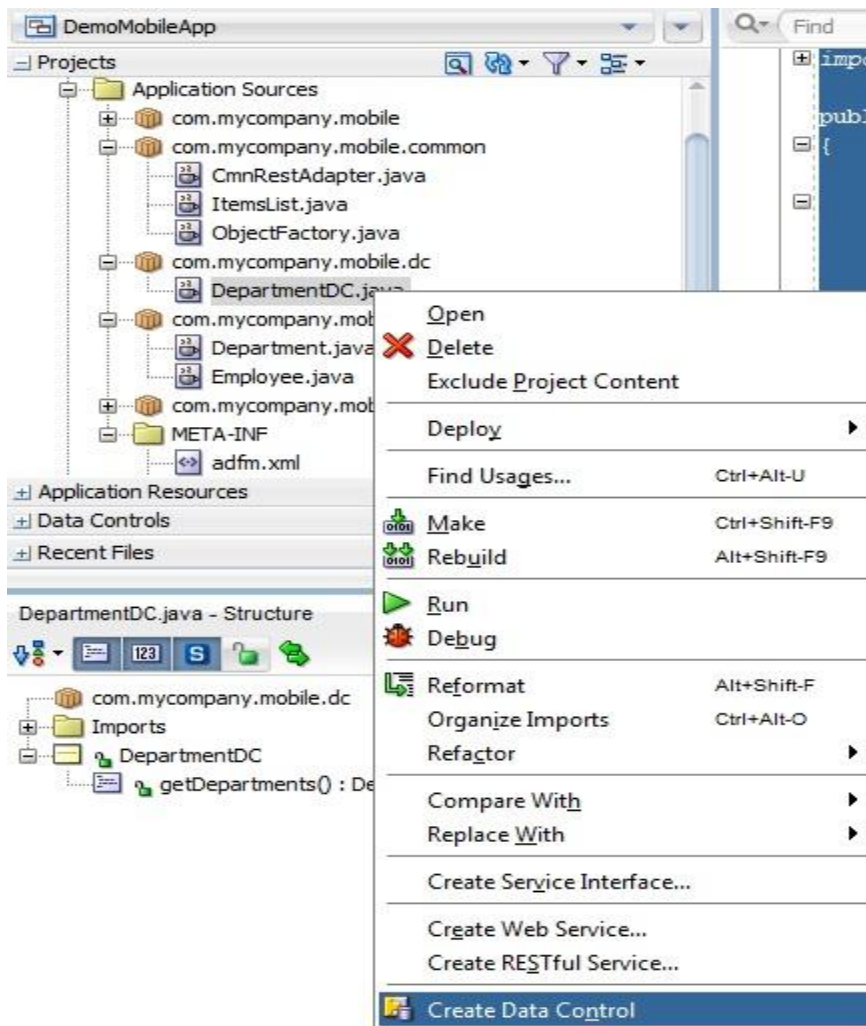
```

```

{
    //de-serialize the JSON response
    deptList = CmnRestAdapter.deserializeToItemsList(response, Department.class);
} catch (Exception e)
{
    throw new AdfException(e.getCause(), AdfException.ERROR);
}
return deptList.getItems();
}
}

```

To expose this as Data Control, right click and select **Create Data Control** as shown below.



This creates **DataControls.dcx** in Default Package and **adfm.xml** file in **META-INF** directory.

adfm.xml

```

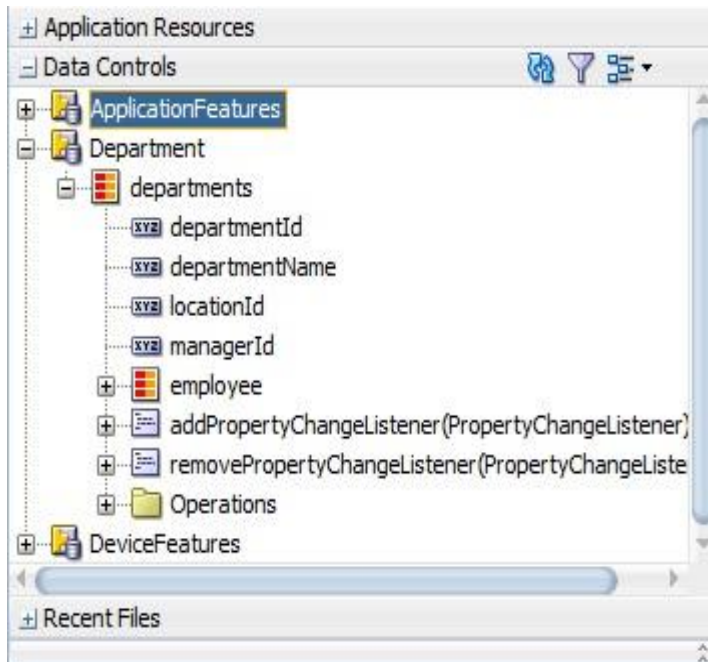
<?xml version="1.0" encoding="UTF-8" ?>
<MetadataDirectory xmlns="http://xmlns.oracle.com/adfm/metainf" version="11.1.1.0.0">
  <DataControlRegistry path="com/mycompany/mobile/DataControls.dcx"/>
</MetadataDirectory>

```

DataControls.dcx

```
<?xml version="1.0" encoding="UTF-8" ?>
<DataControlConfigs xmlns="http://xmlns.oracle.com/adfm/configuration" version="12.1.3.2.1" id="DataControls"
    Package="com.mycompany.mobile">
    <AdapterDataControl id="Department" FactoryClass="oracle.adf.model.adapter.bean.BeanDCFactoryImpl"
        ImplDef="oracle.adf.model.adapter.bean.BeanDCDefinition" SupportsTransactions="false"
        SupportsSortCollection="true" SupportsResetState="false" SupportsRangesize="false"
        SupportsFindMode="false" SupportsUpdates="true" Definition="com.mycompany.mobile.dc.DepartmentDC"
        BeanClass="com.mycompany.mobile.dc.DepartmentDC" xmlns="http://xmlns.oracle.com/adfm/datacontrol">
    <Source>
        <bean-definition BeanClass="com.mycompany.mobile.dc.DepartmentDC"
            DataControlHandler="oracle.adf.model.adapter.bean.DataFilterHandler" AccessMode="scrollable"
            EagerPersist="false" xmlns="http://xmlns.oracle.com/adfm/adapter/bean"/>
    </Source>
</AdapterDataControl>
</DataControlConfigs>
```

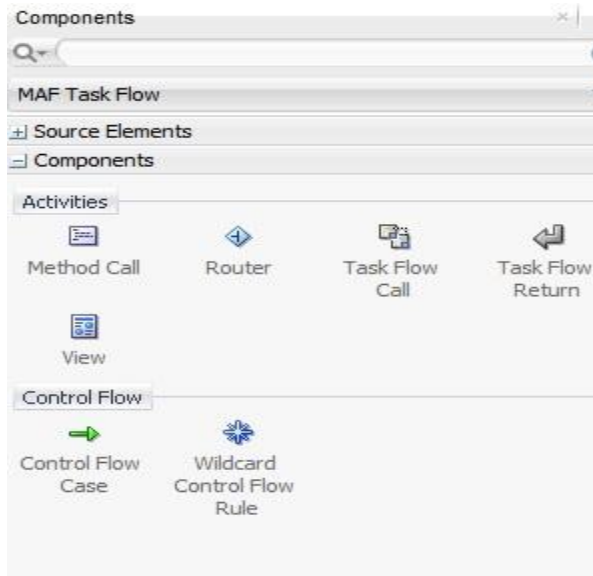
We can also observe this new data control in **Data Controls** tab.



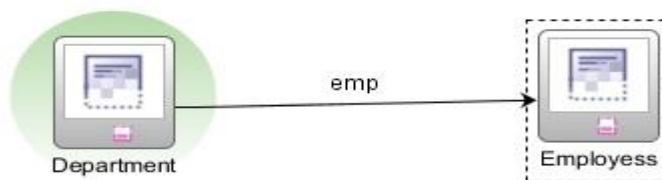
With this, we are done with model and let us proceed to create AMX pages in next section.

Creating AMX Pages

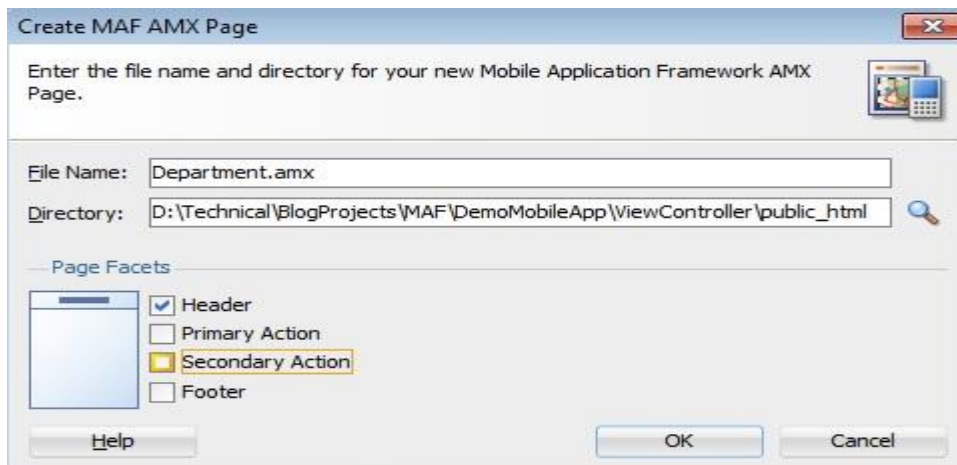
We need 2 AMX pages for our use case to display **Departments** and associated **Employees** Information. Open **dept-task-flow.xml** and drag 2 **View** activities from **Components** window as shown below.

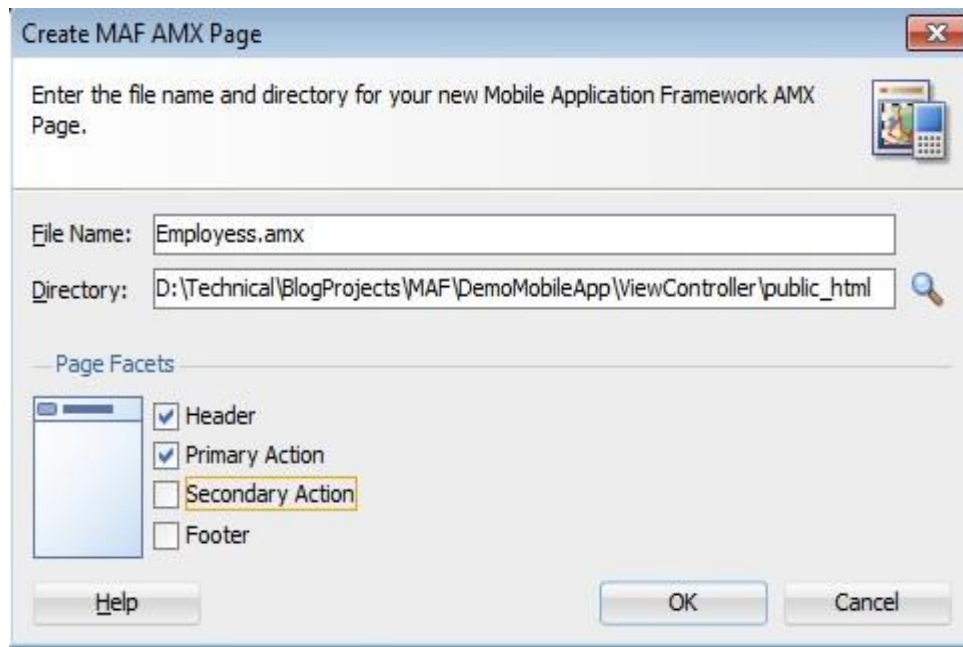


Rename View activities as **Department** and **Employees** respectively. Use **Control Flow Case** from **Components** to define navigation rule **emp** as shown below.

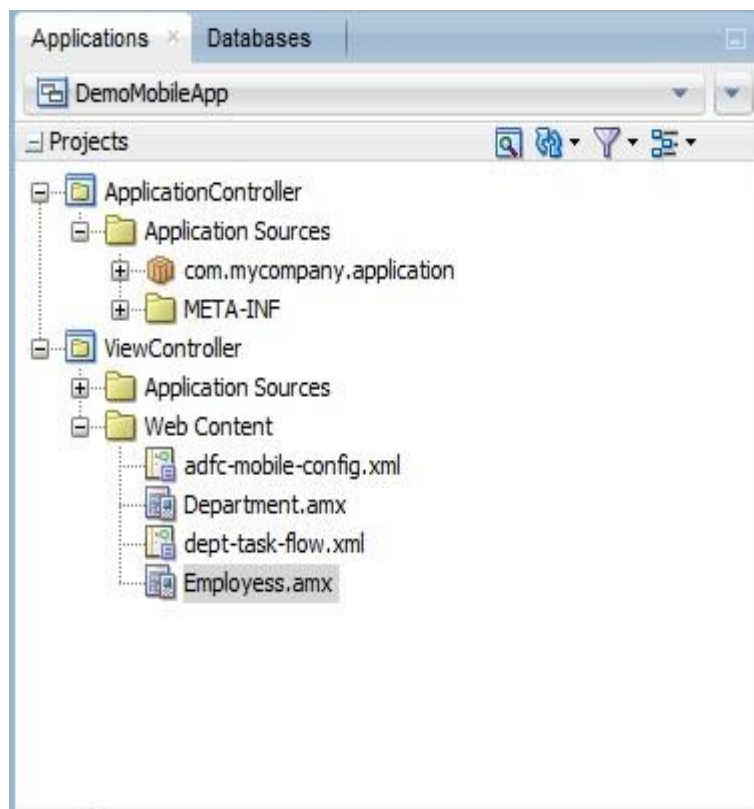


Double click each View activity and create both AMX pages as shown below.

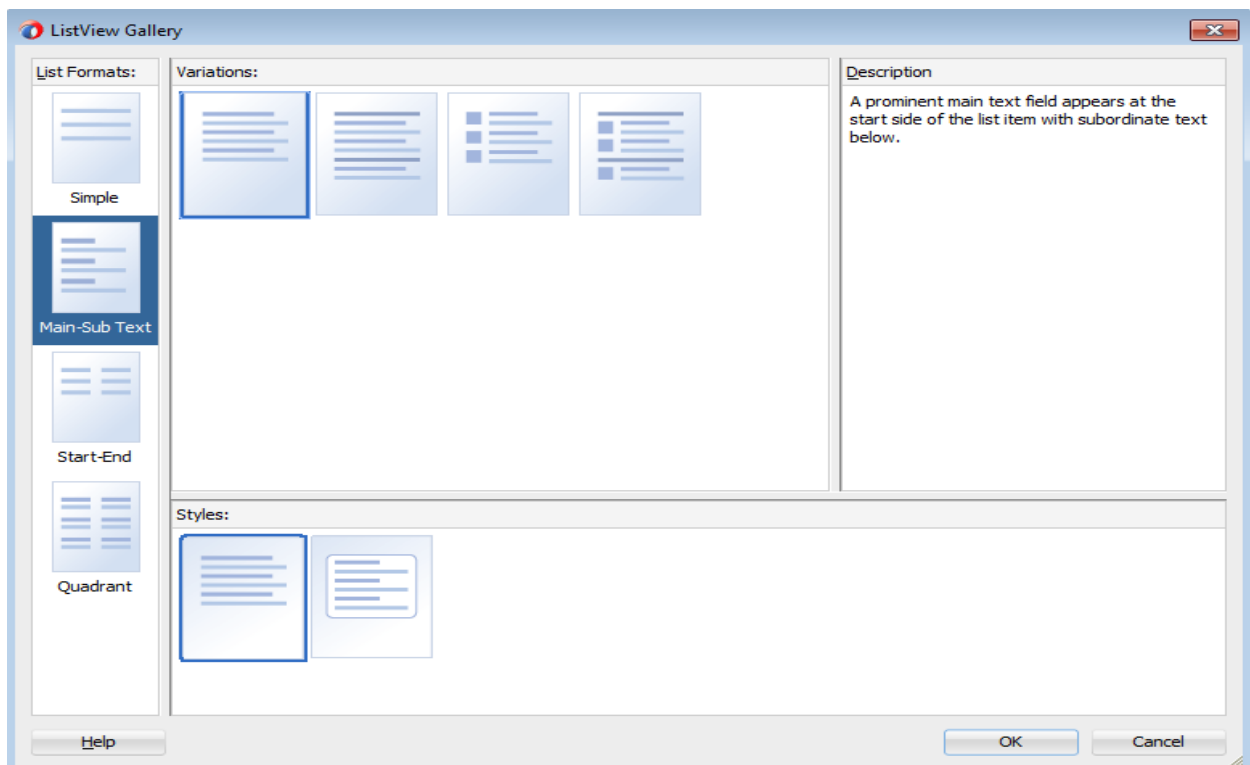
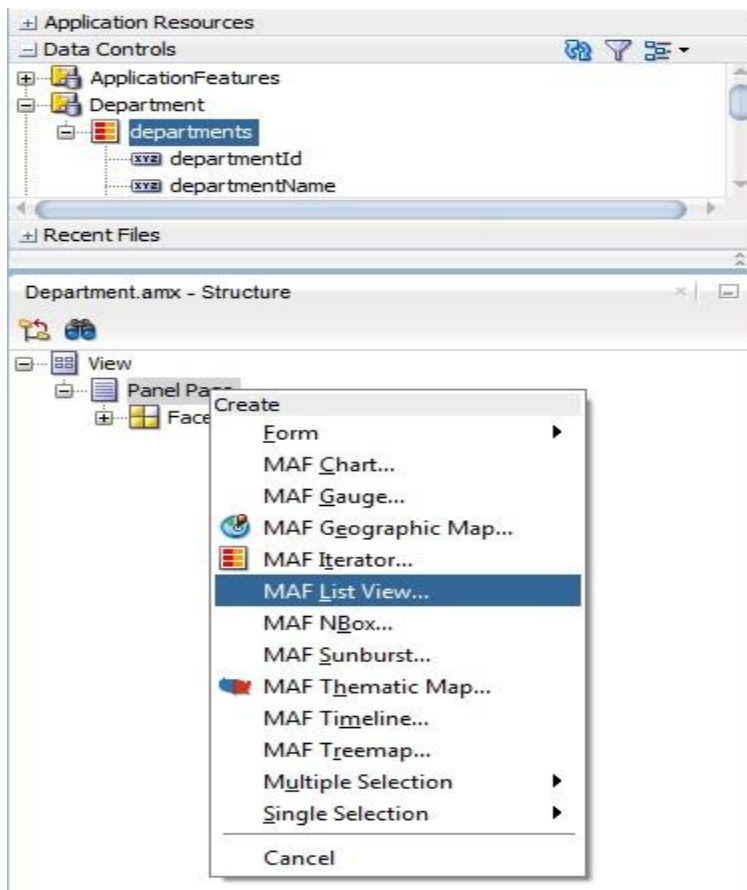


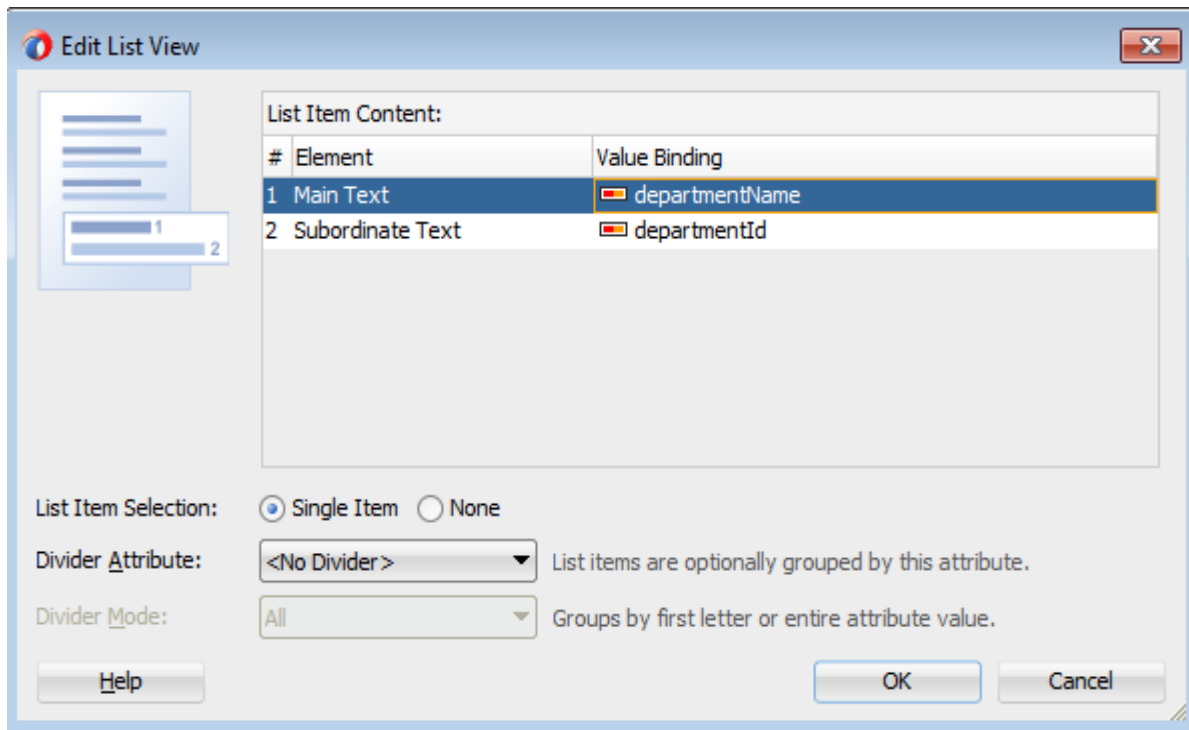


We can observe these AMX pages created in **ViewController** project as shown below.

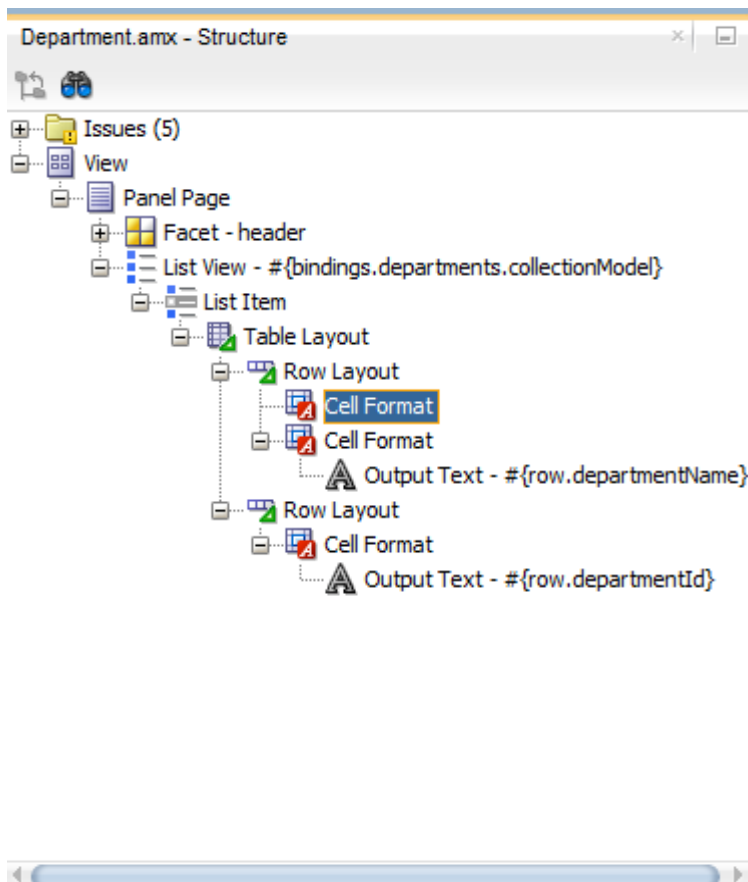


To create Departments UI showing list of departments, drag **departments** collection from **Data Controls** on to **Panel Page** of **Department.amx** as shown below and select **MAF List View**. Select the List View format as required and follow below steps to finish UI.

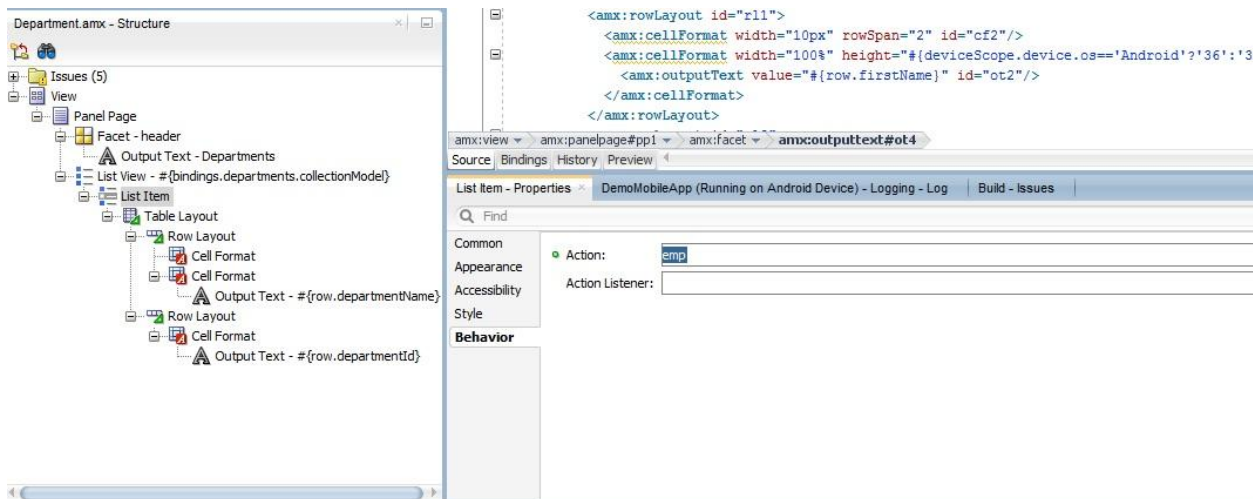




Click **OK** and observe the **List View** created in **Structure** Window as shown below.



Set **Action** property of **List Item** to **emp** to navigate to **Employees** page on selecting a department.



When we use any Data Control for page content, a corresponding **pageDef** (page definition) will be created in **<<Default Package>>.pageDefs** consisting of associated bindings. For our **Department.amx** page, the pageDef will be like shown below.

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uiModel" version="12.1.3.2.1" id="DepartmentPageDef"
    Package="com.mycompany.mobile.pageDefs">
    <parameters/>
    <executables>
        <variableIterator id="variables"/>
        <iterator Binds="root" RangeSize="25" DataControl="Department" id="DepartmentIterator"/>
        <accessorIterator MasterBinding="DepartmentIterator" Binds="departments" RangeSize="25" DataControl="Department"
            BeanClass="com.mycompany.mobile.entity.Department" id="departmentsIterator"/>
    </executables>
    <bindings>
        <tree IteratorBinding="departmentsIterator" id="departments">
            <nodeDefinition DefName="com.mycompany.mobile.entity.Department" Name="departments0">
                <AttrNames>
                    <Item Value="departmentName"/>
                    <Item Value="departmentId"/>
                </AttrNames>
            </nodeDefinition>
        </tree>
    </bindings>
</pageDefinition>
```

This also creates **DataBindings.cpx** in Default Package and updates **adfm.xml** file in **META-INF** directory as below. **DataBindings.cpx** file will have mapping between pages and Data Controls.

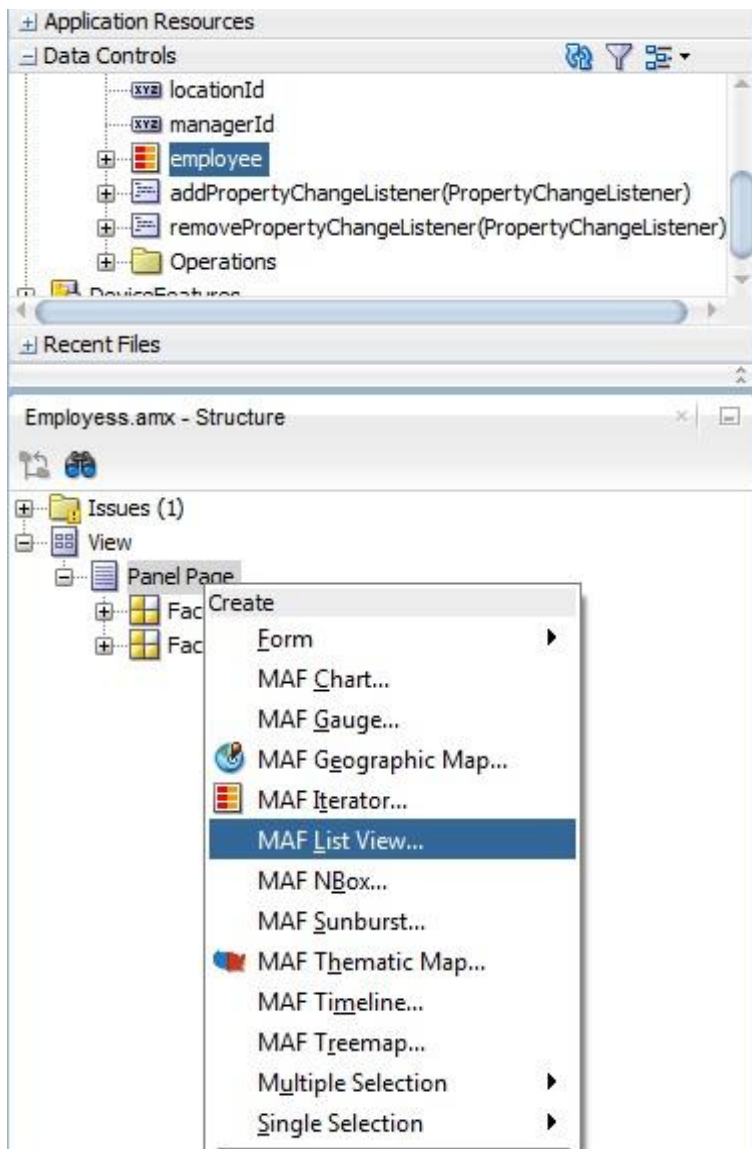
adfm.xml

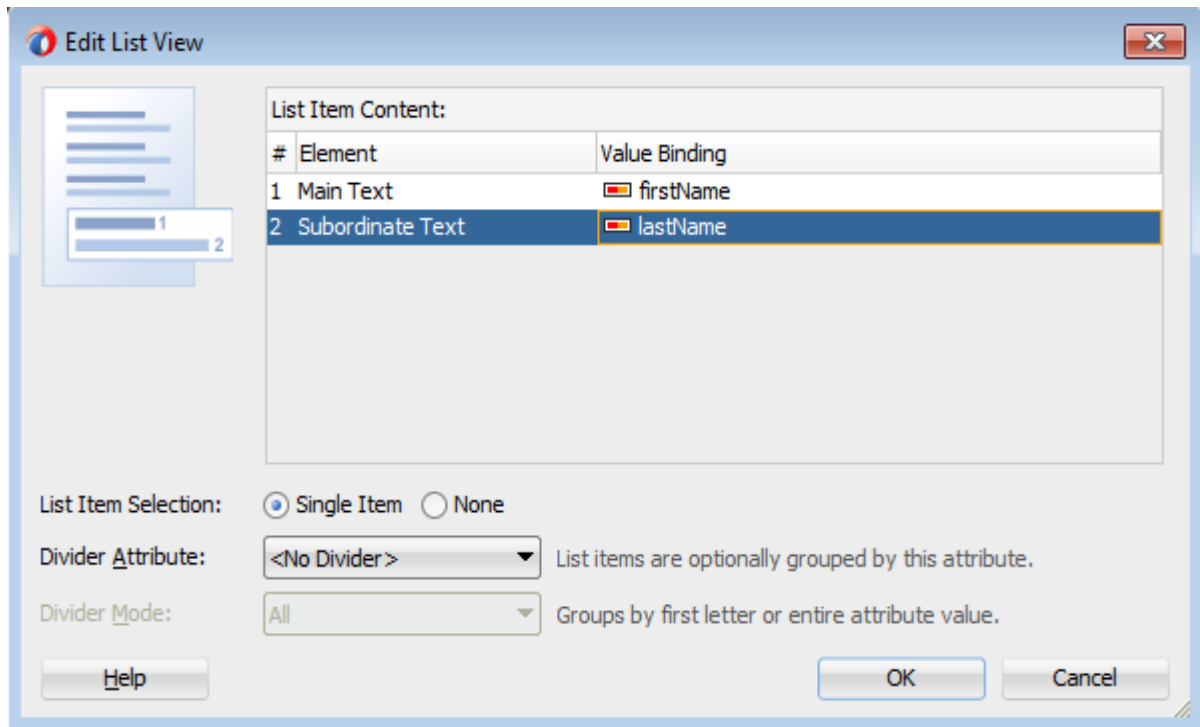
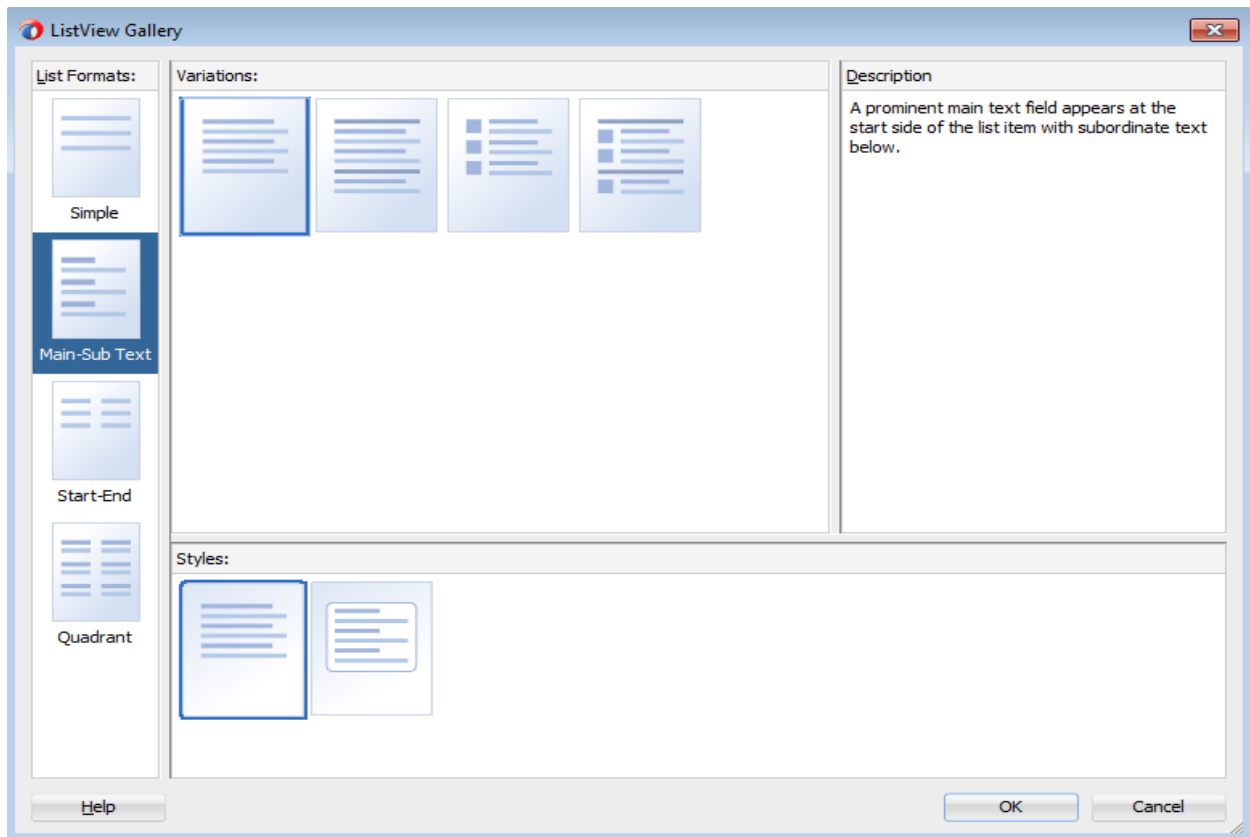
```
<?xml version="1.0" encoding="UTF-8" ?>
<MetadataDirectory xmlns="http://xmlns.oracle.com/adfm/metainf" version="11.1.1.0.0">
    <DataControlRegistry path="com/mycompany/mobile/DataControls.dcx"/>
    <DataBindingRegistry path="com/mycompany/mobile/DataBindings.cpx"/>
</MetadataDirectory>
```

DataBindings.cpx

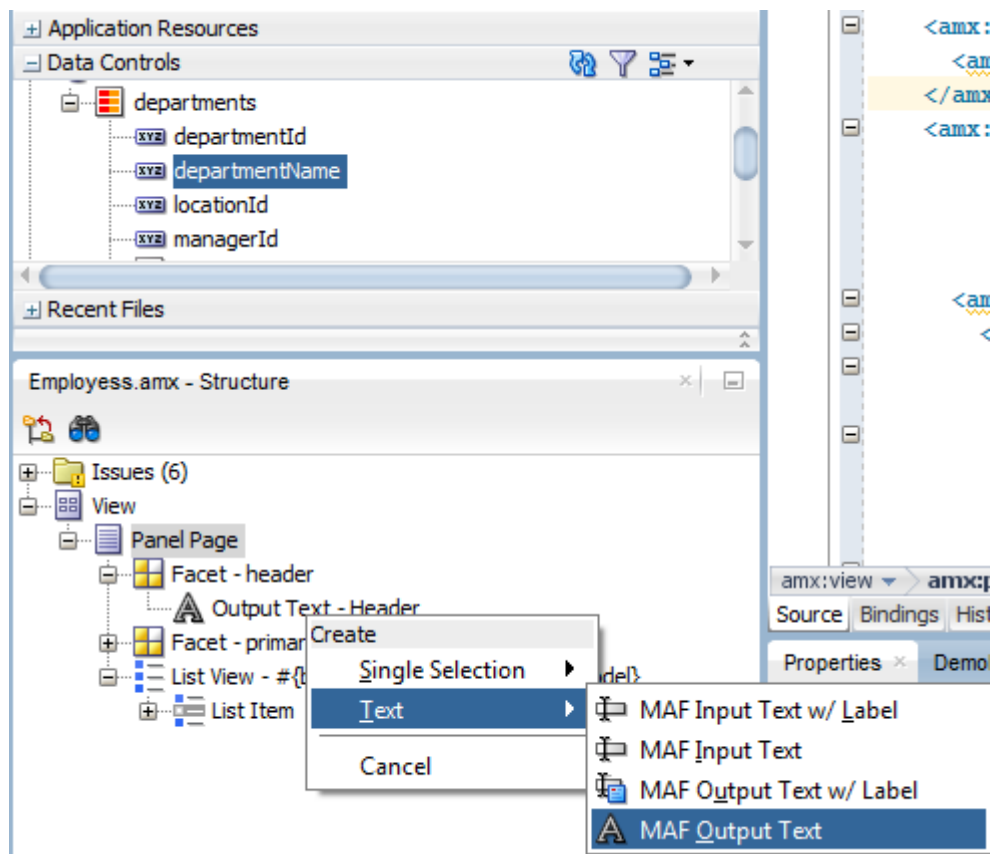
```
<?xml version="1.0" encoding="UTF-8" ?>
<Application xmlns="http://xmlns.oracle.com/adfm/application" version="12.1.3.2.1" id="DataBindings"
    SeparateXMLFiles="false" Package="com.mycompany.mobile" ClientType="Generic">
    <pageMap>
        <page path="/Department.amx" usageId="com_mycompany_mobile_DepartmentPageDef"/>
    </pageMap>
    <pageDefinitionUsages>
        <page id="com_mycompany_mobile_DepartmentPageDef" path="com.mycompany.mobile.pageDefs.DepartmentPageDef"/>
    </pageDefinitionUsages>
    <dataControlUsages>
        <dc id="Department" path="com.mycompany.mobile.Department"/>
    </dataControlUsages>
</Application>
```

Similarly, drag **employee** collection (child resource) on to **Employees.amx** as shown in below.

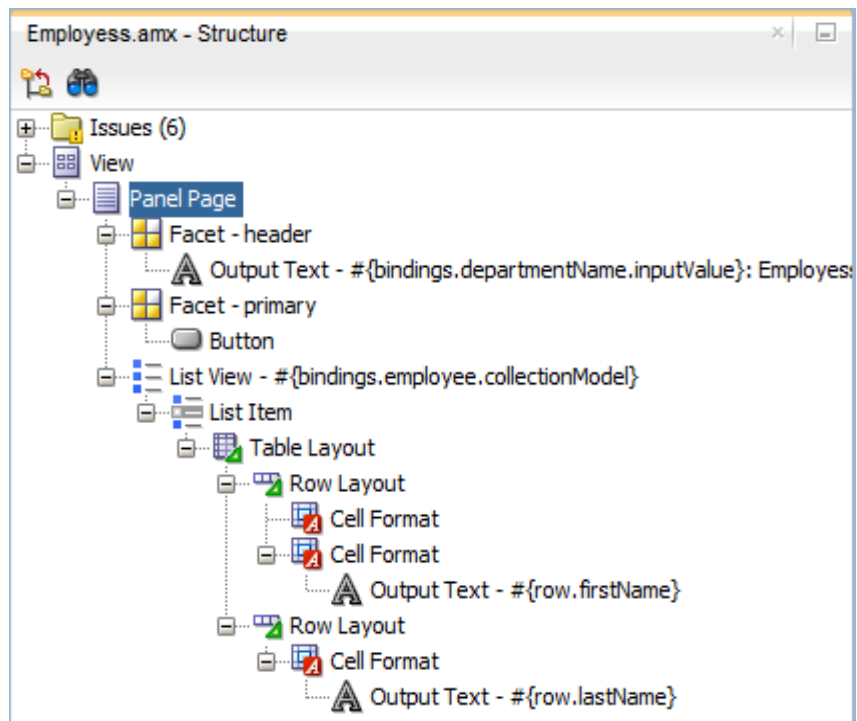




Now drag **departmentName** from **departments** collection into header facet. This creates new output text field so delete the existing one having label as **Header**.



Now the page looks like below in Structure Window.



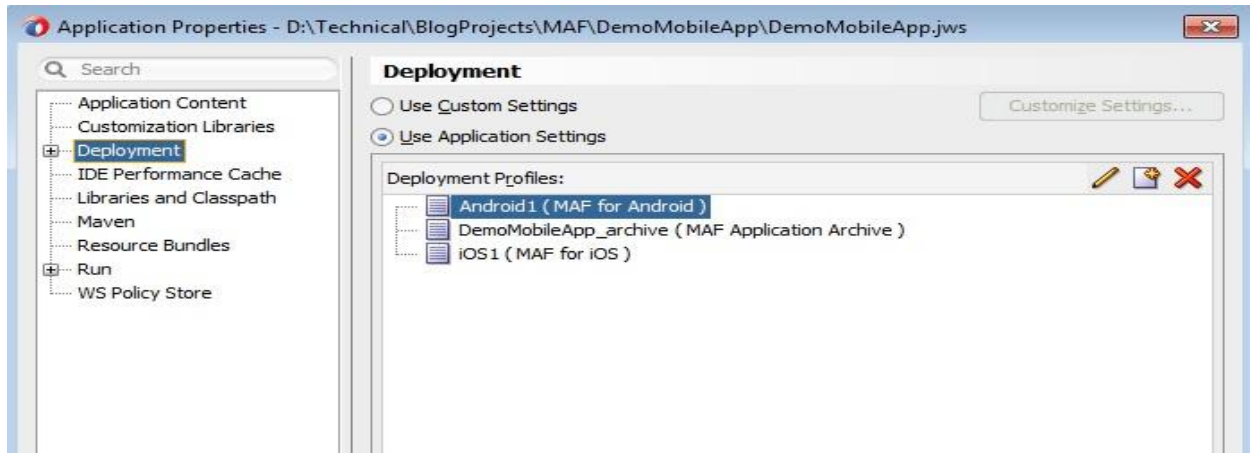
We can see corresponding pageDef as shown below. Observe that 2 different accessor iterators are created in **executables** section as we used attributes belonging to two different collections (department and employees) in this page.

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel" version="12.1.3.2.1" id="EmployeePageDef"
    Package="com.mycompany.mobile.pageDefs">
    <parameters/>
    <executables>
        <variableIterator id="variables"/>
        <iterator Binds="root" RangeSize="25" DataControl="Department" id="DepartmentIterator"/>
        <accessorIterator MasterBinding="DepartmentIterator" Binds="departments" RangeSize="25" DataControl="Department"
            BeanClass="com.mycompany.mobile.entity.Department" id="departmentsIterator"/>
        <accessorIterator MasterBinding="departmentsIterator" Binds="employee" RangeSize="25" DataControl="Department"
            BeanClass="com.mycompany.mobile.entity.Employee" id="employeeIterator"/>
    </executables>
    <bindings>
        <tree IterBinding="employeeIterator" id="employee">
            <nodeDefinition DefName="com.mycompany.mobile.entity.Employee" Name="employee0">
                <AttrNames>
                    <Item Value="firstName"/>
                    <Item Value="lastName"/>
                </AttrNames>
            </nodeDefinition>
        </tree>
        <attributeValues IterBinding="departmentsIterator" id="departmentName">
            <AttrNames>
                <Item Value="departmentName"/>
            </AttrNames>
        </attributeValues>
    </bindings>
</pageDefinition>
```

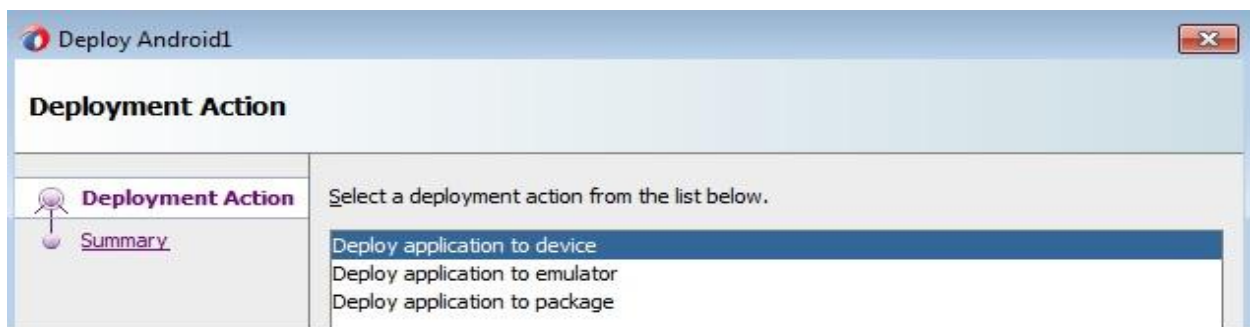
With this, we are done with the creation of UI and let us proceed with deployment and testing.

Deployment

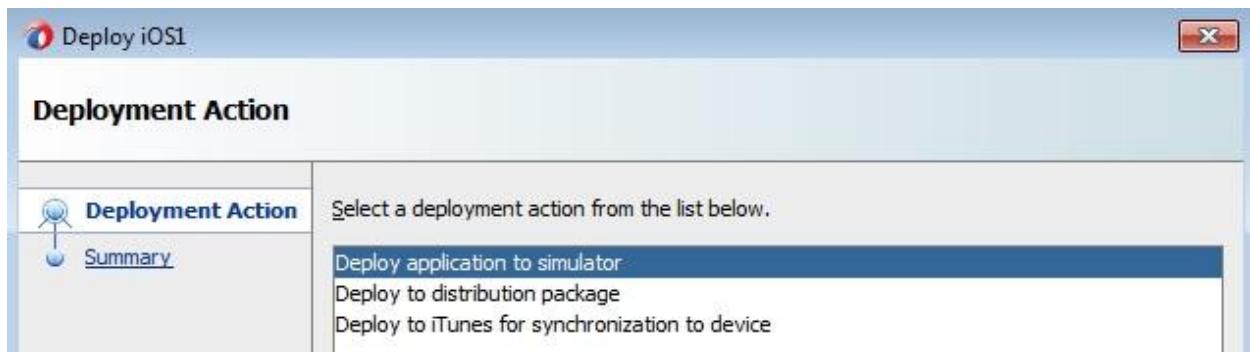
When we create MAF application 3 deployment profiles are created automatically catering to Android, iOS and MAA (MAF Application Archive) that can be observed in **Application Properties -> Deployment**.



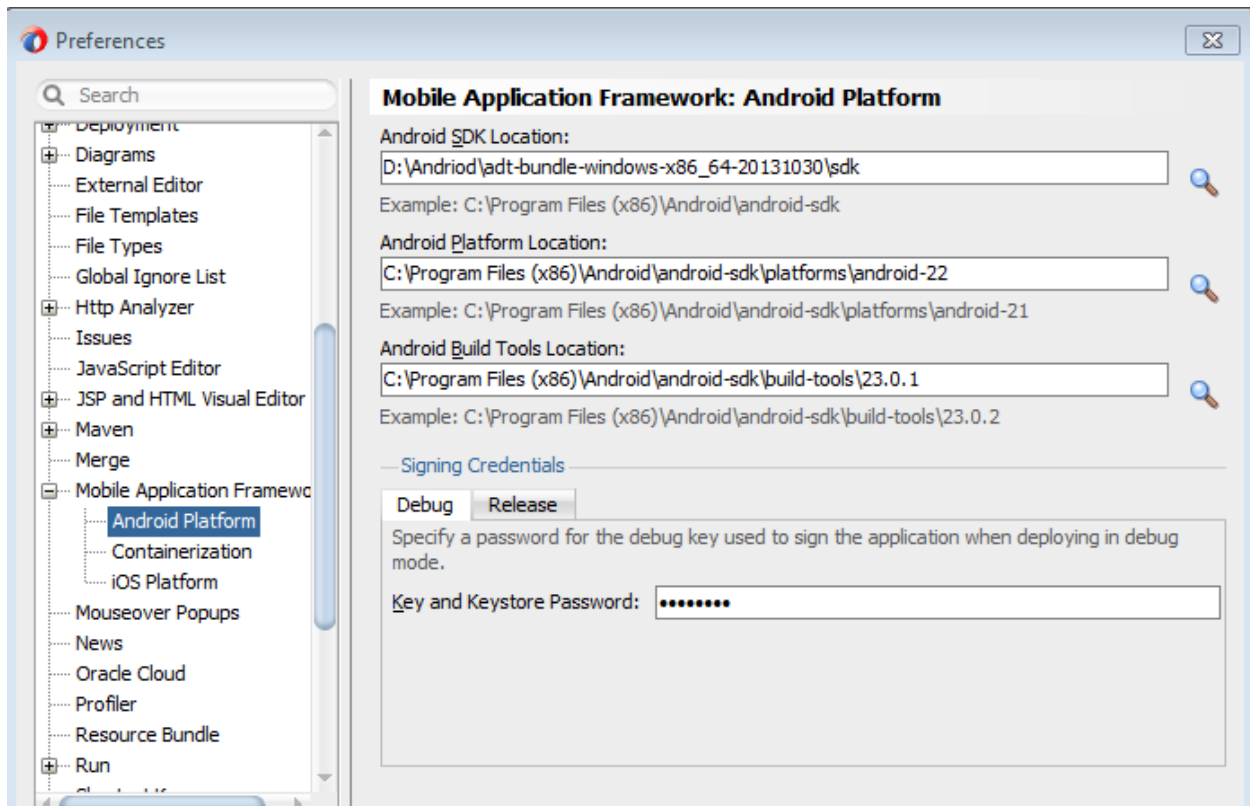
Android deployment profile enables us to deploy MAF application to Device, Emulator and package (apk) as shown below. While deploying to the device, make sure that **USB debugging** is enabled in **Developer Options** and **Allow installation of apps from other sources** is set in **Security settings** of your device.



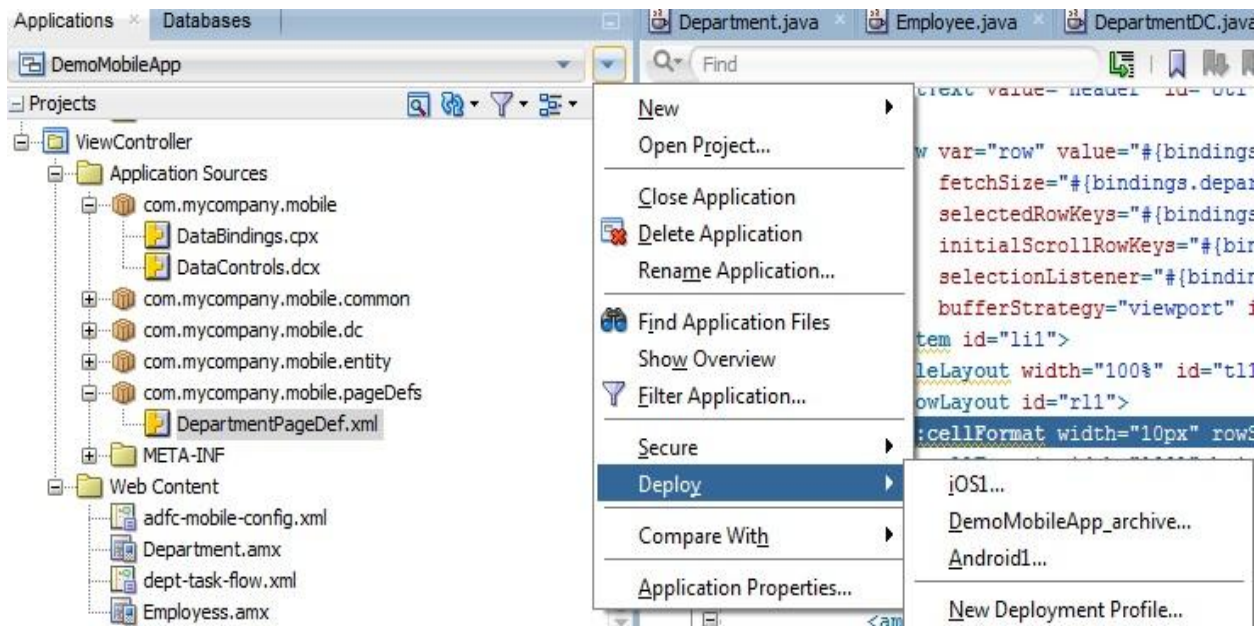
Similarly, iOS deployment profile enables us to deploy MAF application to Simulator, package (ipa) and iTunes for Synchronization as shown below. For this, we need to use only Mac machine and also we need to have **Developer Profile** from Apple. So we will use android device to show this deployment.

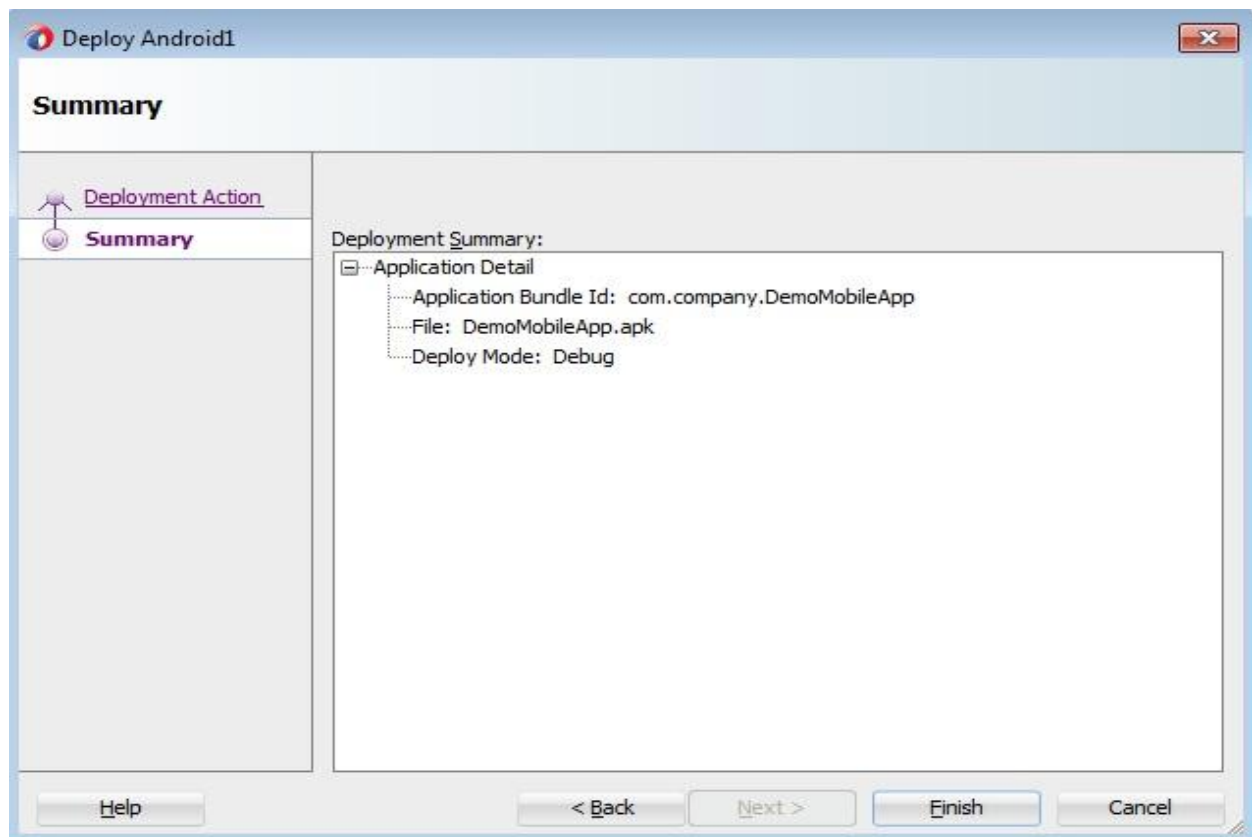
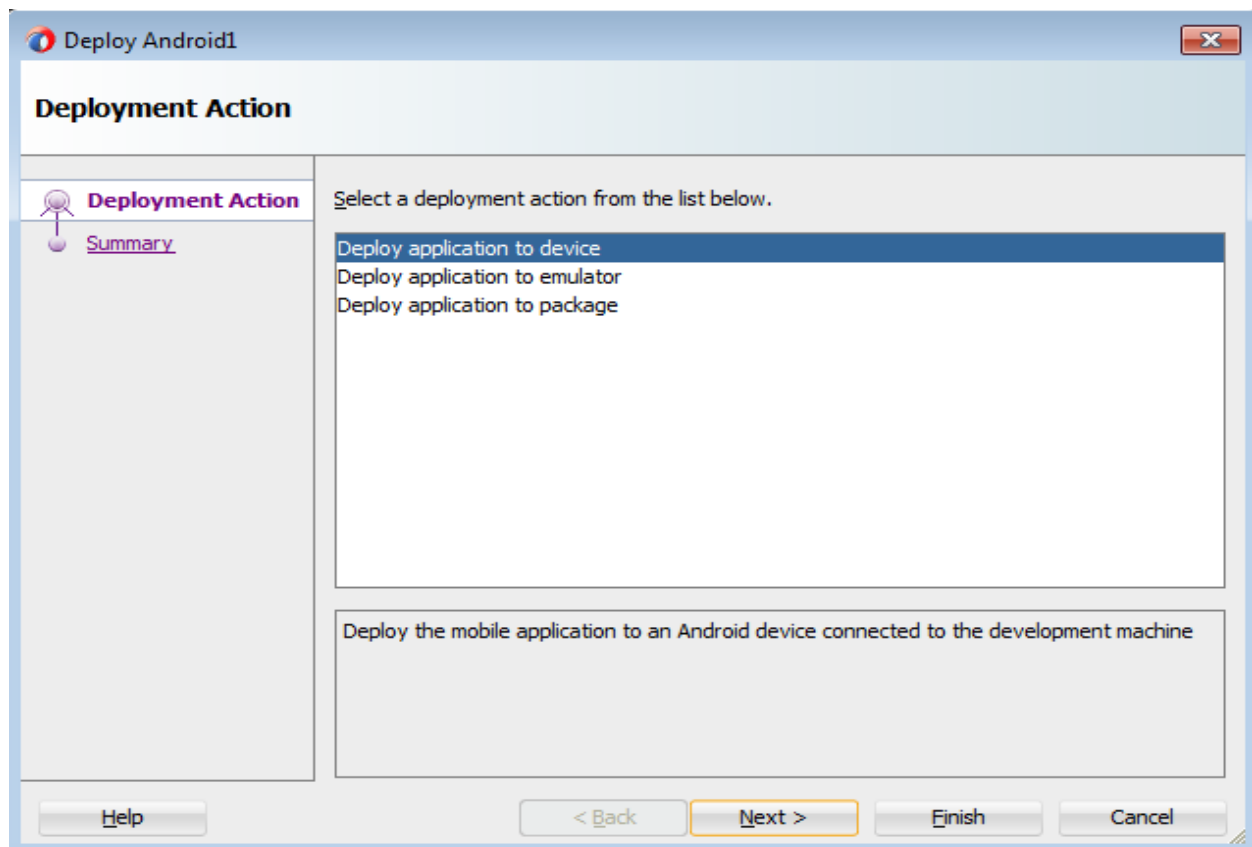


Before starting deployment, make sure that Android SDK location is properly set in **Tools -> Preferences -> Mobile Application Framework-> Android Platform** as shown below. For more information on deployment options and setup refer to MAF documentation.



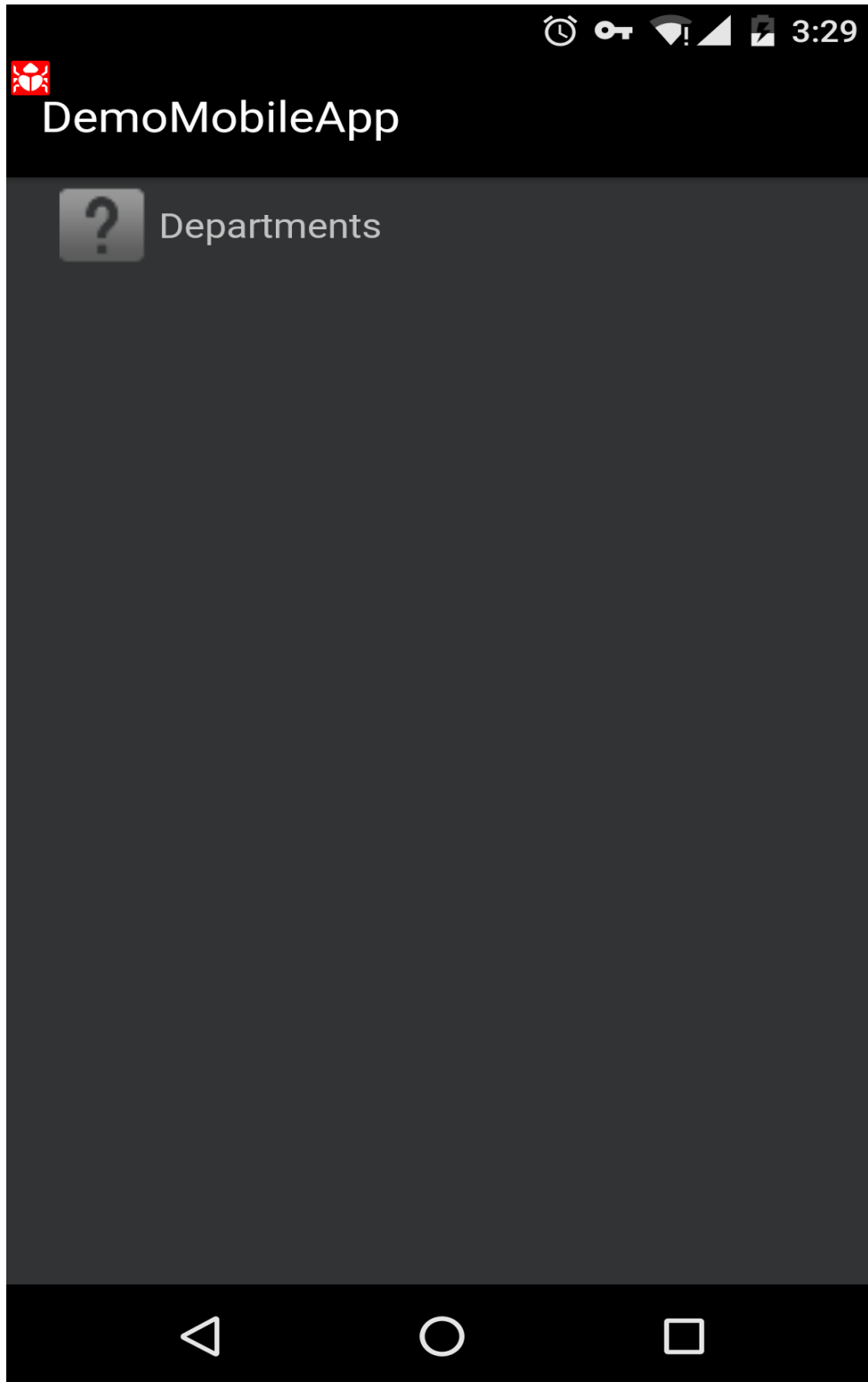
Use the following screenshots and deploy the application to device.





Testing

Once the application is deployed, you can view the springboard and the actual pages as shown below.





Departments

Administration1aaaa1E

10

Marketing

20

Purchasing

30

Human Resources

40

Shipping

50

IT

60

Public Relations

70

Sales

80



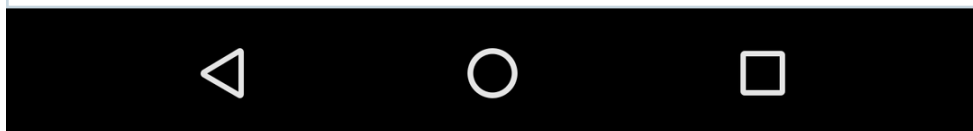
Marketing: Employess

Michael

Hartstein

Pat

Fay



Downloads

[MAF Application](#) (Modify the host and port in connections.xml before testing)

[ADF BC REST Application](#)