Oracle ADF 11g

Contents

4. ADF Task flows

4.1-1 Introduction to ADF TFs.
4.1-2 TF activities.
4.1-3 TF Features.
4.1-4 TF Template.
4.1-5 TF Data control scopes.
4.1-6 TF Transaction options.
4.1-7 Run TF and Run TF as a dialog.
4.1-8 Dynamic Region.
4.1-9 Task Flow Initializers/Finalizers.
4.2-1 Managed beans in TF.
4.2-2 Train components in TF.
4.2-3 TF Input parameters.
4.2-4 BTF vs UBTF

PART –II (not completed yet)

5. ADF faces components
6. Security
7. MDS
8. Deployment

1.1-1 Introduction to Oracle ADF

ADF stands for "Application Development Framework". With ADF we can implement an end to end, Secure, High performance application (of any type) in quick time with the help of several Built-In components implemented (in Core Java and XML). Oracle ADF is not a freeware. The clients should have WebLogic server license to deploy the ADF applications on it.

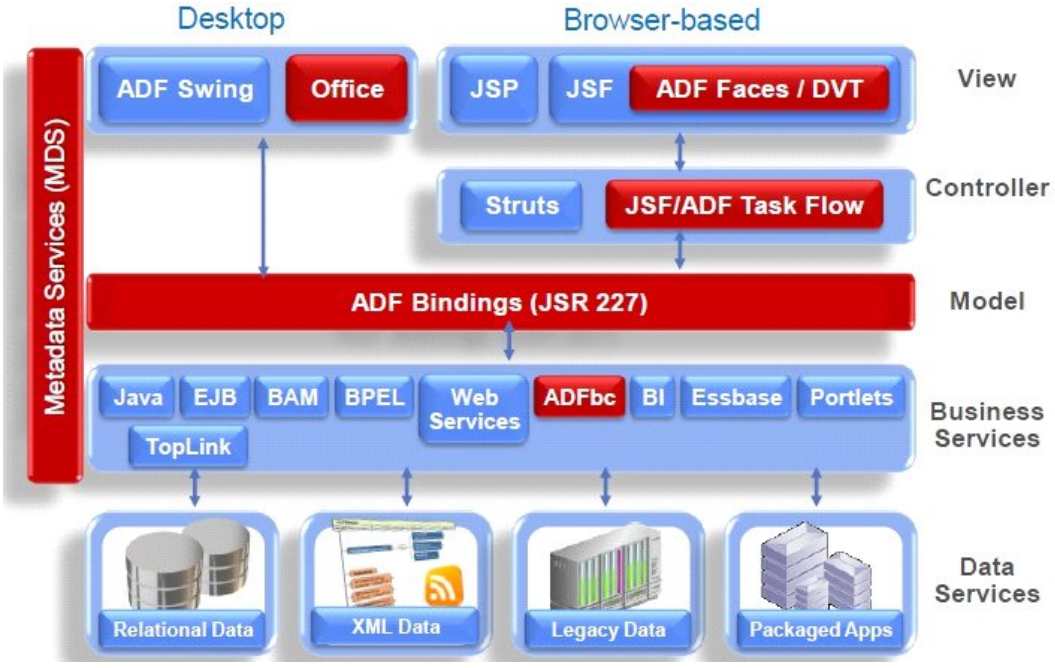
1.1-2 ADF history

OAF: Oracle starts with a framework called “Oracle Application Framework (OAF)” for providing a sophisticated user interface for all its customers. OAF have lot of performance issues because of technology stacks used for developing this framework. OAF has limited features with the name of “BC4J” instead of Business components. And there are no Task flows in OAF to support the GUI based navigation model. OAF completely depends upon Java coding and some configuration files to provide the navigation model (controller layer). Compare to Java/J2EE and OAF has very less coding. With OAF we have few performance issues as the developers add considerable amount of code.

ADF 10g:-To resolve the issues in OAF, Oracle implemented a framework called “ADF 10g”. In ADF 10g we have to include java code instead of ADF BC to form a Business Layer. ADF 10g contains ADF task flows to support the GUI based navigation model. Still ADF 10g have lot of performance issues because of lot of java code needs to be written for providing a business layer.

1.1-3 ADF architecture

Oracle ADF 11g:- Oracle came up with a framework called ADF11g and it is resolved all the performance issues identified in OAF and ADF 10g. Oracle ADF provides ADF Business components [1] (ADF BC) to form a Business Layer. So that our ADF application can interact with any relational Database and perform the basic Data Base operations (CURD Operations). Oracle ADF provides ADF Bindings [2] to form a Model Layer so that our ADF Application has a sophisticated interface to provide the effective communication between top layers (View-Controller Layer) and lower layer (Business layer).



Oracle ADF provides ADF Task Flows [3] to form a Controller Layer. So that our ADF application can have an effective Navigation Model. ADF Task flows are also called as ADF page flows

or ADF Controller. Oracle ADF provides ADF Faces components [4] to provide an interactive user interface (Web pages). Oracle ADF provides Declarative Security [5] Model and Declarative Deployment [6]. Oracle ADF provides an environment called to hold the runtime changes permanently with the help of metadata services (MDS [7]).

Oracle ADF is a technology in a Fusion Middleware Stack (FMWs). ADF interacts with Universal Content Management (UCM) server for maintaining various types of content (text, image, audio and video content) and interacts with WebCenter Portal for Personalization's, Customizations, Declarative Navigation Model, Security, Social Networking and Productivity Services. ADF interacts with SOA for supporting various types of clients. ADF interacts with BPM for supporting the BPM System by providing required information.

Prerequisites to learn ADF: Knowledge/Experience on Core Java and SQL queries is mandatory. Exposure on PI/SQL is an added advantage.

IDE: ADF applications are implemented either in Jdeveloper IDE or Eclipse IDE.

1.1-4 ADF features

- ADF applications are Secure as the ADF is directly implemented on Java and inherits all the features of it.
- ADF promotes Declarative development. The Java/XML codes automatically generated with respect to front end changes (drag/drop).
- ADF supports Rapid application development due to the incredible number of declarative components available and are scattered across all the layers.
- ADF adds value to Java. ADF is not an advanced Java or extension to Java. It is a simple framework implemented on Java platform.
- ADF enables developers to focus much on client requirements and not low level things like coding, complexity and team size.
- ADF follows a best design pattern called MVC to develop ADF applications. The basic Model layer is classified into two: Business layer (ADF BC) and Model layer (ADF bindings).

  The Controller layer (ADF task flows) and View layer (ADF faces components) are same.

2. ADF Business Components

2.1-1 Introduction to ADF BC

Oracle ADF provides ADF BC to form a Business Layer so that our ADF application can perform below operations declaratively:

- Establish a connection to Data Source.

- Create a new record into the Database.
- Perform delete and update operations after retrieve.
- Perform only retrieve (Read only) operation.
- Provide Transaction Support (Commit/Rollback).
- Close the Connections.

ADF BC is responsible for performing the all above operations with the help of below Business components or Business Objects.

- EO (Entity Object) :-

  EO is created on top of a single Table/View/Synonym to support the updatable view object to perform all the CURD operations.

- UVO (Updatable View Object) :-

  It is a query created on one or more Entity Objects to perform the basic CURD operations.

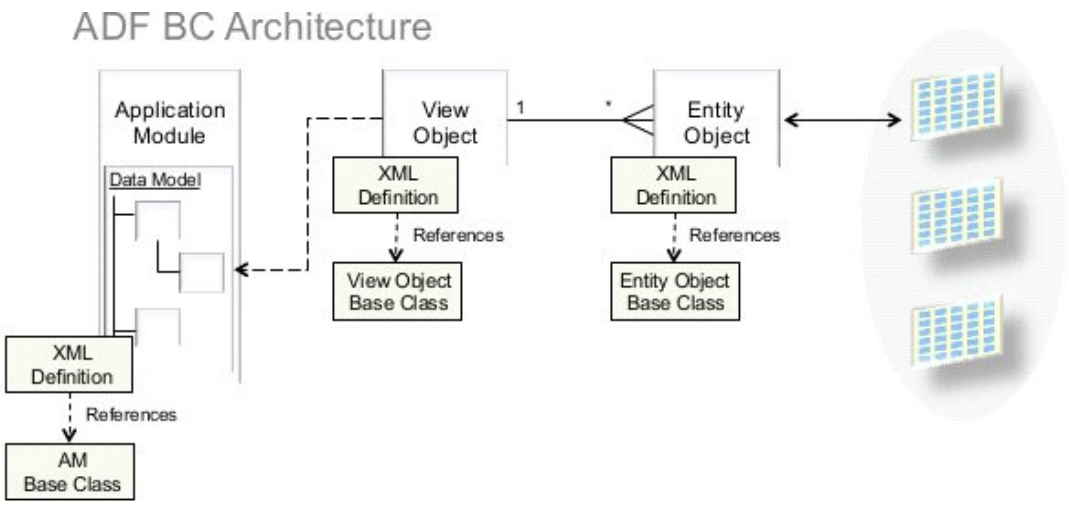
- RVO (Read-only View Object) :-

  It is a query created directly on one or more Database tables with the help of joins to perform only retrieve operations (Select or Read-only). We can't perform DML operations with the help of RVO.

- AM (Application Module) :-

  AM contains VO instances which contains the actual data and also supports the Transaction Management on the Database data.

  VO provides the structure whereas VO instance represents the actual data associated to the structure.



2.1-2 Application Architecture

In ADF, we generally call the ADF implementations as ADF Applications. By default each ADF Application contains two projects they are:

- Model Project:-

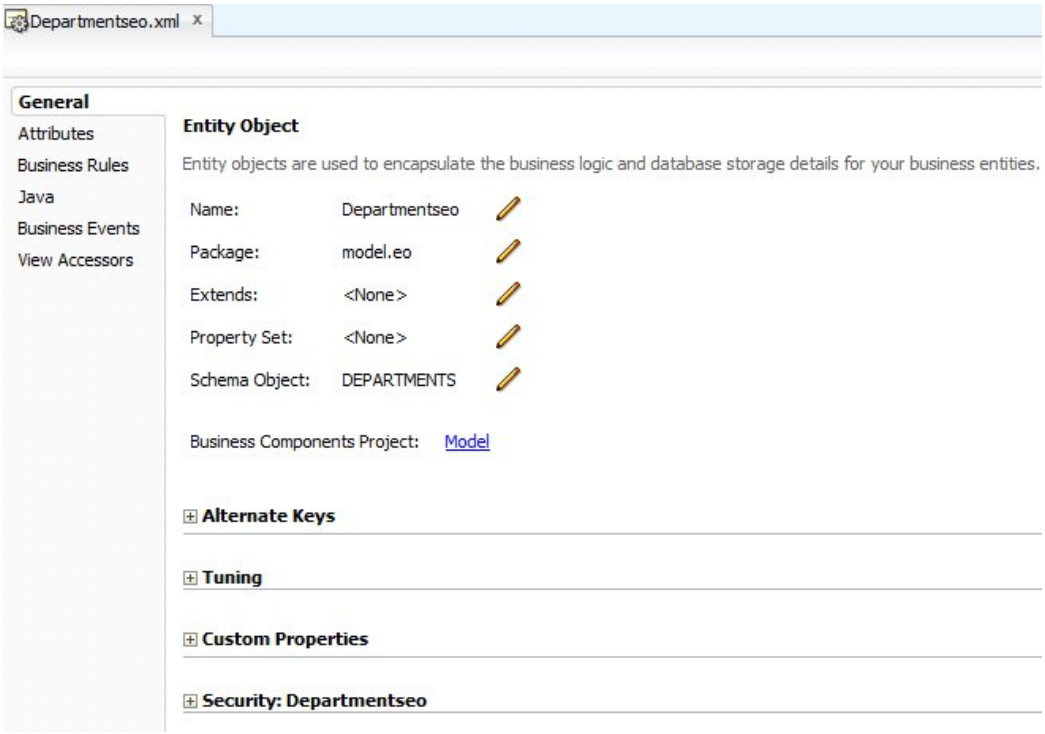
  The Model Project contains all the ADF BC implementations (Business Layer) and required Java classes (impl classes).

- View Controller Project:-

  The Model layer, Controller layer, View layer implementations (ADF bindings, ADF task flows, and ADF faces components) and required Java classes (Managed Bean) are exists in View Controller Project.

2.1-3 Entity Objet (EO)

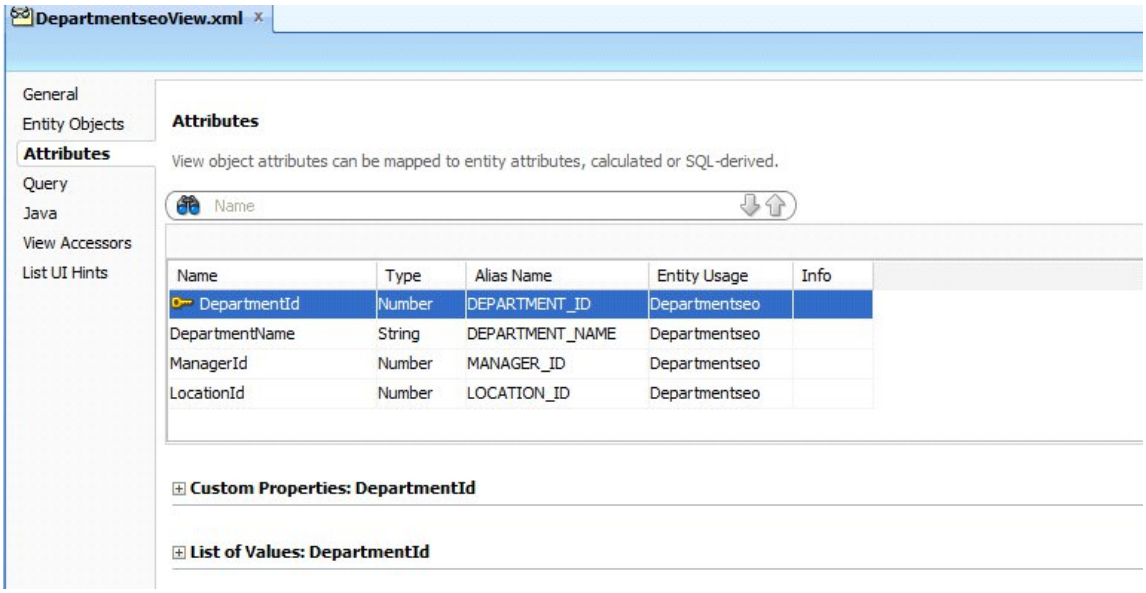
EO is an object created directly on a single DB table or a view. EO contains attributes and each attribute represents an associated column. One of the attribute should be Key-Attribute. The Key-Attribute is an attribute which is created on the Primary Key Column. If there is no Primary Key Column then an attribute named Row-Id will be created automatically and it is treated as Key-Attribute. All the column constraints will be appeared as Built-In Business Rules (can't able to delete) in the EO Business Rules section. According to Business requirement, we can create our own (Custom) Business Rules.



EO usually has one implementation class called EOImpl java class. This EOImpl java class represents a single row at a particular point of time. The purpose of EO is to perform all the CURD operations with the help of Entity Based VO (UVO) EO works behind the screen as we can't able to configure directly in the Application Module.

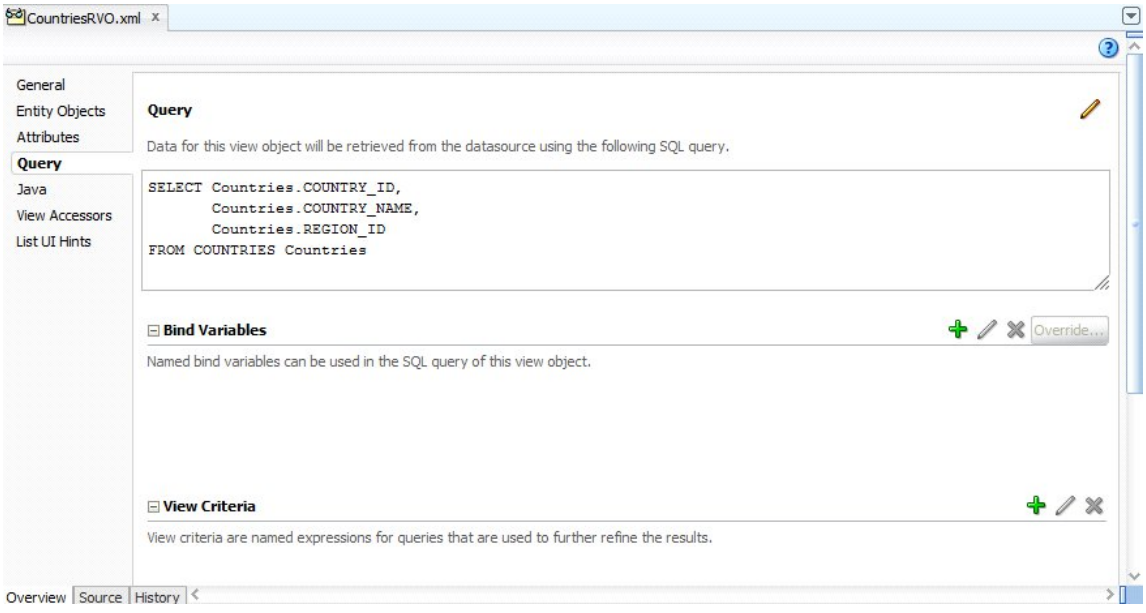
2.1-4 Updatable View Object (Updatable VO)

UVO is a query which is represents a Row-set UVO are created on one or more EO's with the help of SQL joins with the help of UVO we can perform all the CURD operations. Each VO contains VOImpl java class which represents a Row-set .This UVO contains attributes and is carried from EO.



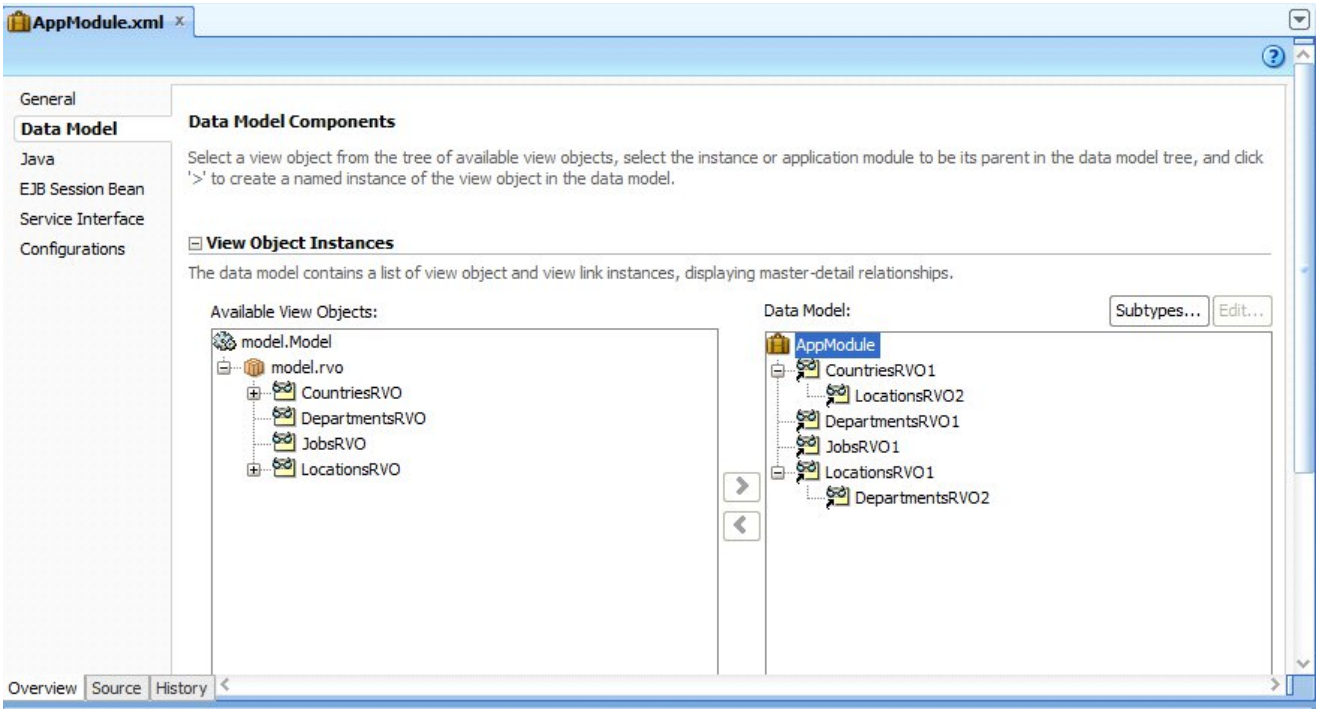
2.1-5 Read only View Object (Read only VO)

RVO is a query created on one or more DB tables with the help of SQL joins. We can't perform any DML operations with RVO. It is only for retrieving the records. Each VO contains the Built-in operations to support the declarative development. Some of the Built-in operations are create, First, Last, Next, Previous, Delete etc.



2.1-6 Application Module (AM)

AM is a container of VO instances (Read-only VO or Updatable VO). The VO instances actually represent the data. AM provides a test browser to test all the configured VO instances in the Data Module section. AM provides Transaction Context for all its configured updatable VO instances. Each AM contains an AM data control created under Data Controls section and AM data control is the outcome of entire ADF BC.



2.1-7 Types of VO Attributes

Based on the below types of the parameters we can easily identify the type of the attribute in any type of view.

Condition 1:- Calculated Attribute

- DB column mapping exist.
- The Content should be there in the SQL query.
- If the attribute is exist in the RVO.

Condition 2:- Persistence Attribute

- DB column mapping exist.
- The Content should be there in the SQL query.

- If the attribute is exist in the UVO.

Condition 3:- Calculated Attribute

- DB column mapping doesn't exist.
- The Content should be there in the SQL query.
- If the attribute is exist in the RVO.

Condition 4:- Calculated Attribute

- DB column mapping doesn't exist.
- The Content should be there in the SQL query.
- If the attribute is exist in the UVO.

Condition 5:- Transient Attribute

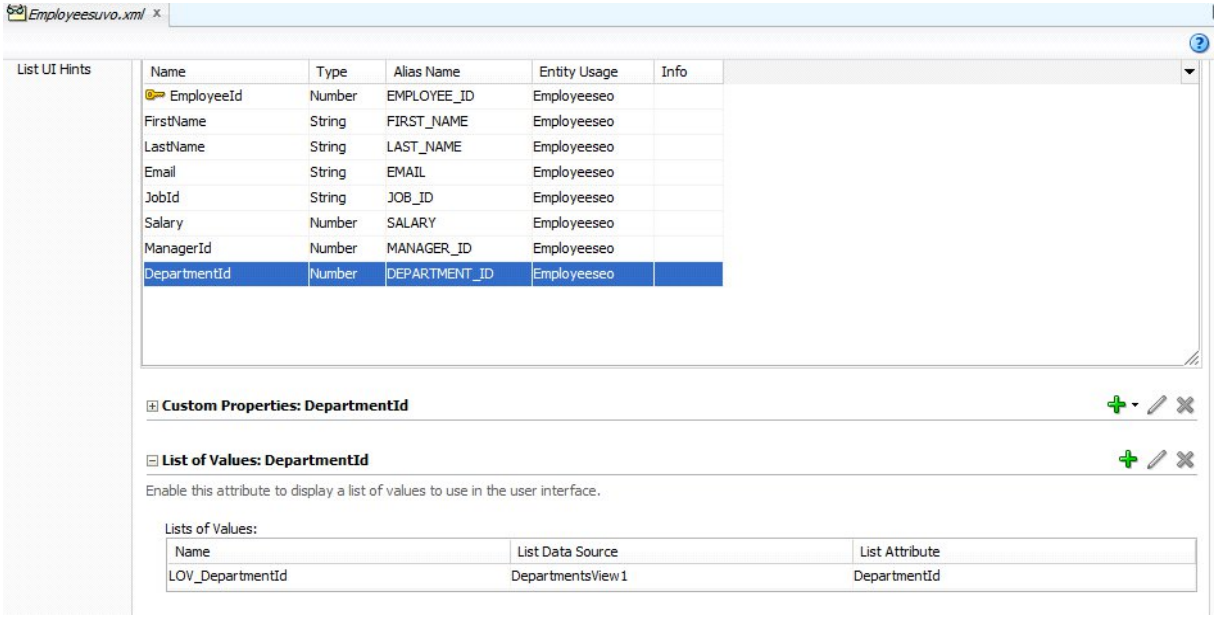
- DB column mapping exist/doesn't exist.
- The Content is not there in the SQL query.
- If the attribute is exist in the RVO or UVO.

If the attribute is existed in query the value will be evaluated during query processing and all the values should be available ready to display. In case of transient attribute there is no content in the query but the values will be provided with the help of the Expression. Expression assigned to the expression property. In case there is no large amount of data and we are showing small amount of data to customers then it is good to go with Transient Attribute.

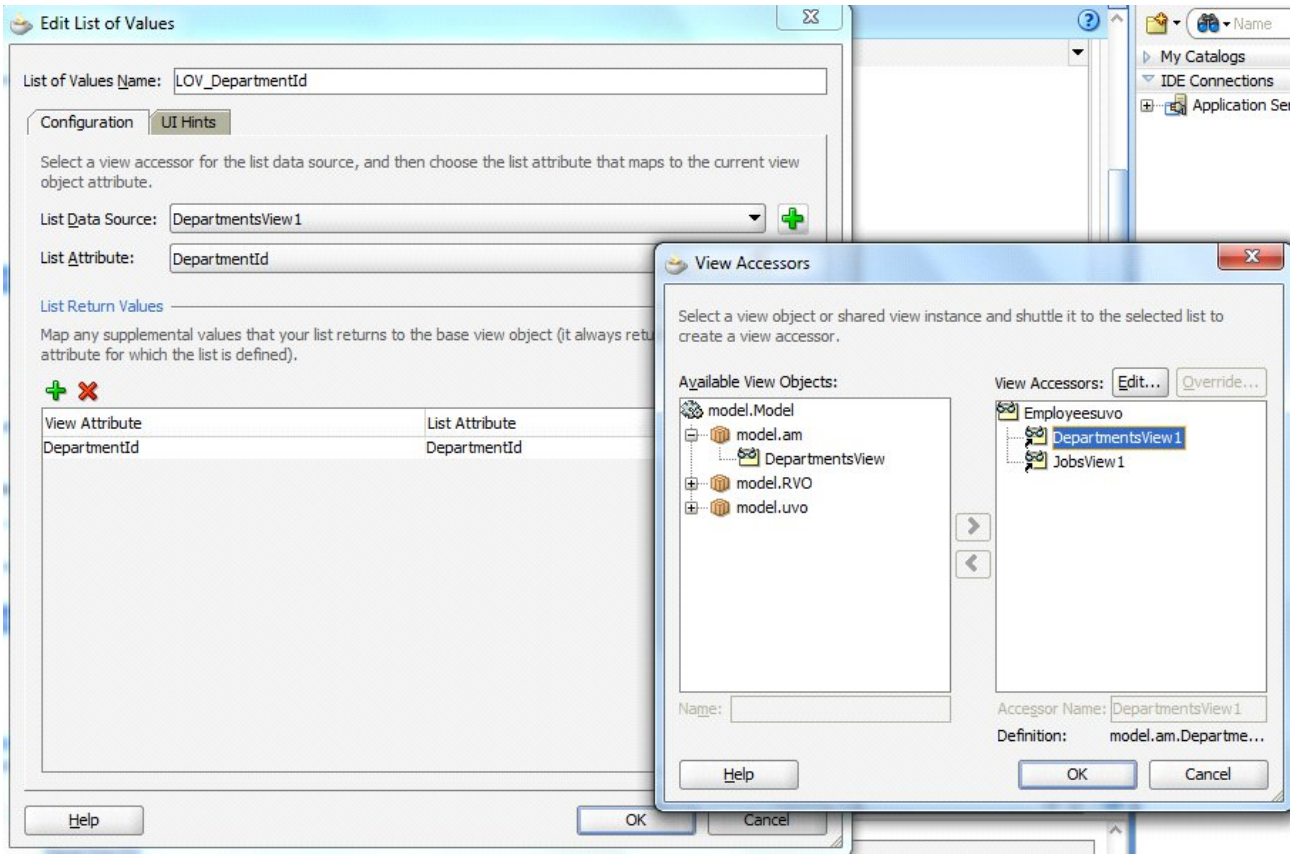
2.1-8 LOV (List of Values) and View Accessor

To make the web pages effective while filling the data for foreign key attributes we should depend upon a concept called List Of Values. In general we are going to create LOV for foreign key attributes in UVO. The VO which provides the list (List Data source) would be key attribute VO. List attributes represents actual content needs to be stored as a data. List UI hints represents the actual content needs to be shown to the customer.

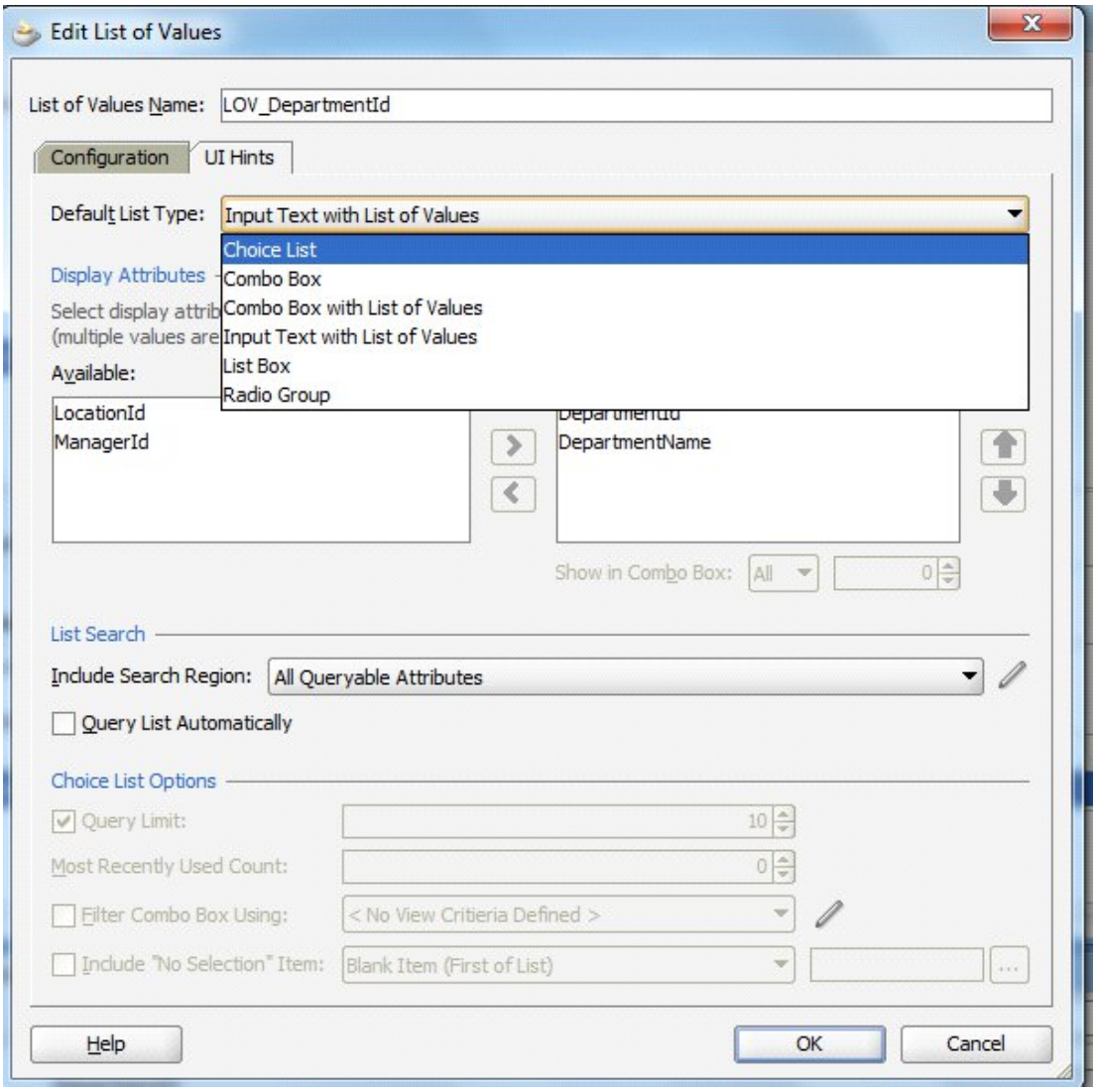
Step-1: Create Lov on DepartmentId attribute of Employee Updatable VO.



Step-2: Configure List data source and List attribute.



Step-3: Configure List UI hints.



NOTE :

There is no need of separate VO instances for list Data source configured in the AM data module.

The View accessor will automatically access the required data as a list. View accessor automatically created when we are creating a LOV.

2.1-9 View Criteria

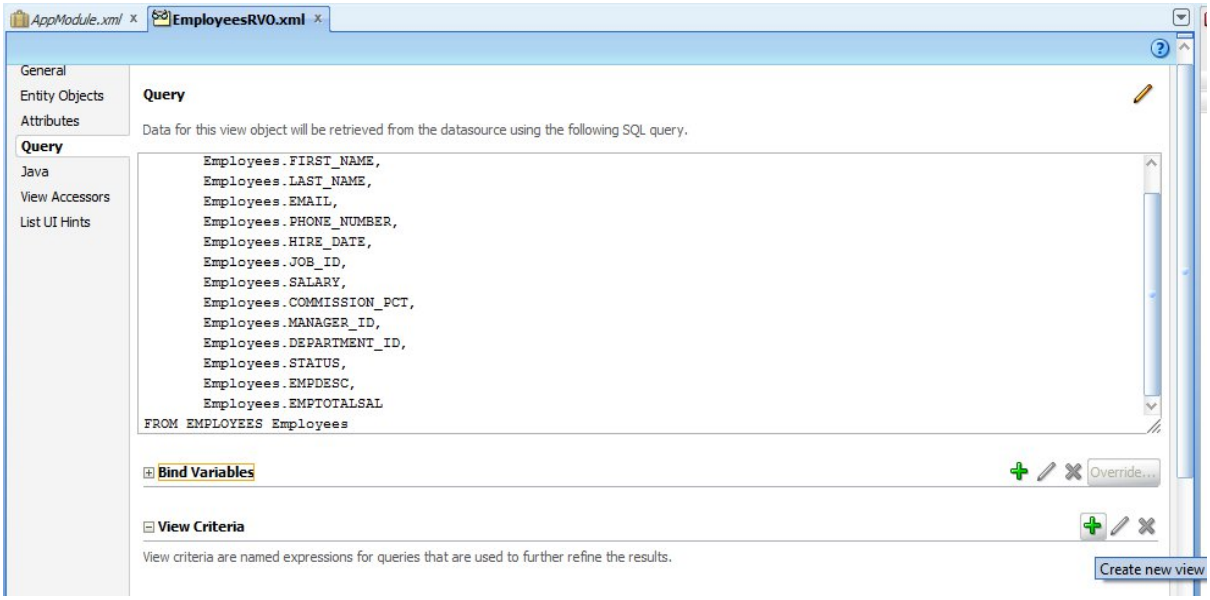
View Criteria is a “Where” clause which makes the VO independent. On a VO (either UVO or RVO) we can have any number of View Criteria’s. Based on a Business requirement we are going to pull the appropriate VO/View Criteria and apply the View criteria programmatically.

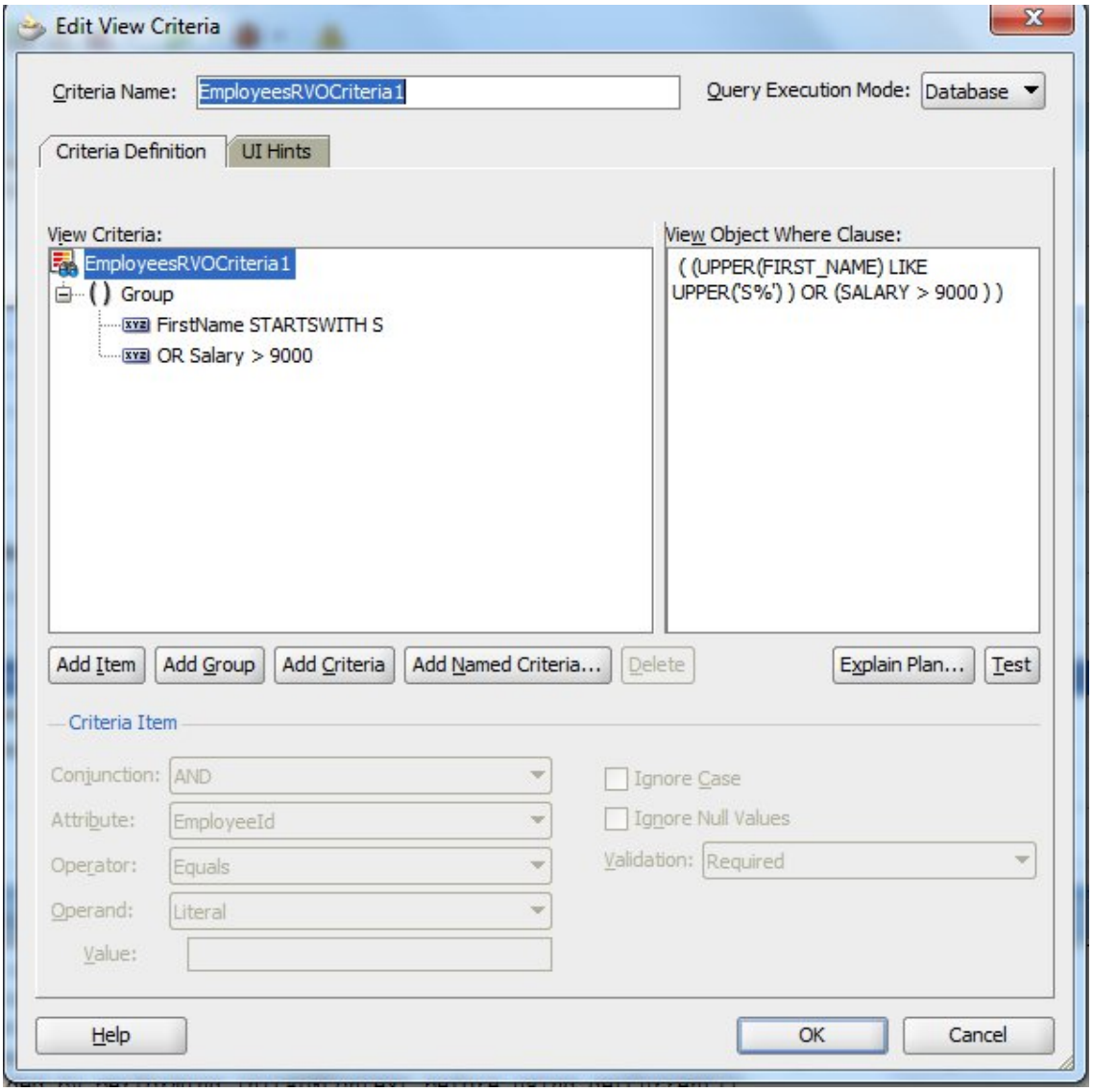
View Criteria is specified into two types they are

- Static View Criteria.
- Dynamic View Criteria.

- Static View Criteria:-

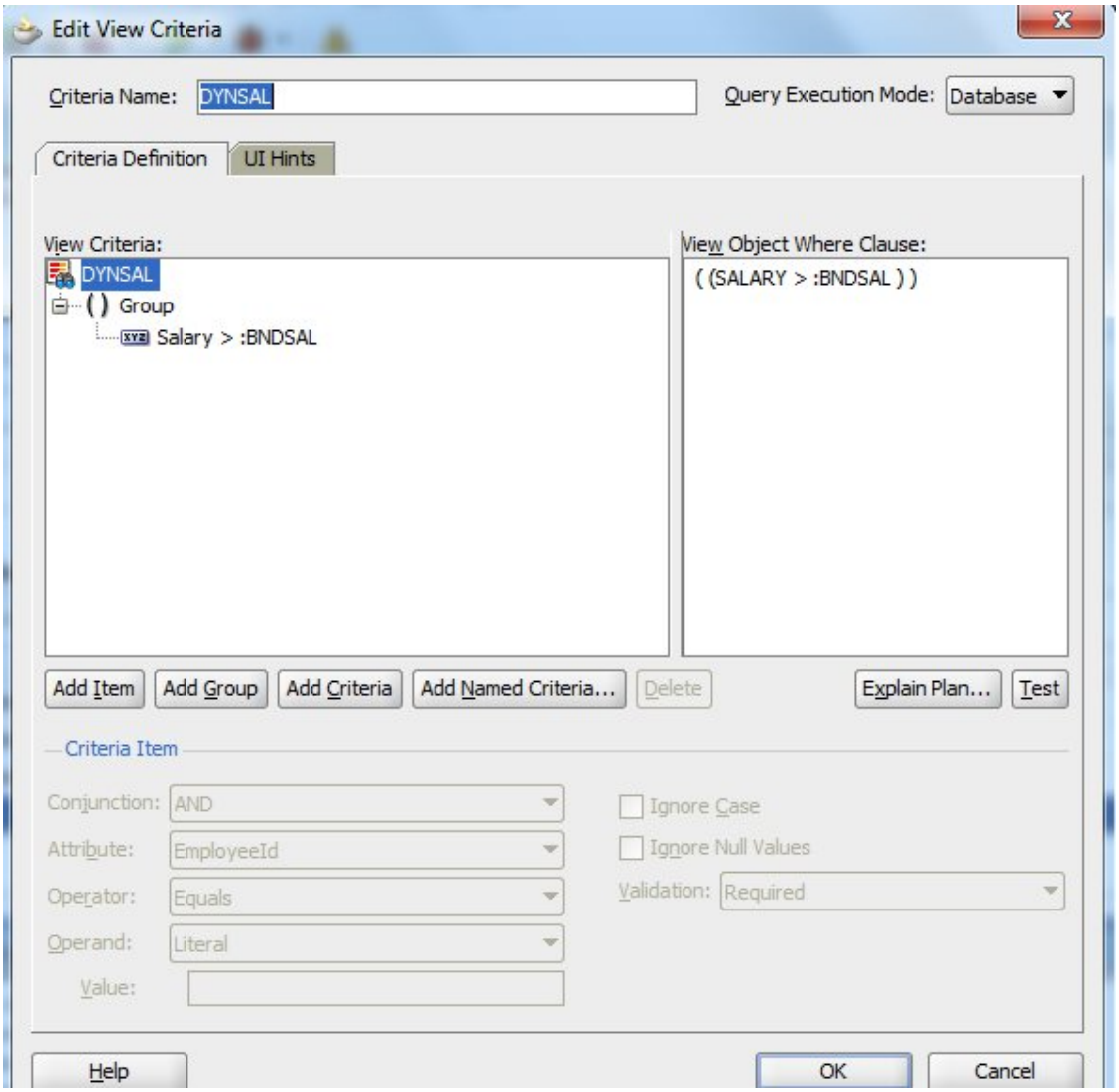
Static View Criteria contains literals as part of group items. These Static Criteria’s are not reusable and we need to create several static view criteria’s for the same condition for different constant values. The maintenance of static view criteria are pretty difficult.





- Dynamic View Criteria:-

  The Dynamic View Criteria is View Criteria which contains bind variables or dynamic variables (instead of literals) in group Items.

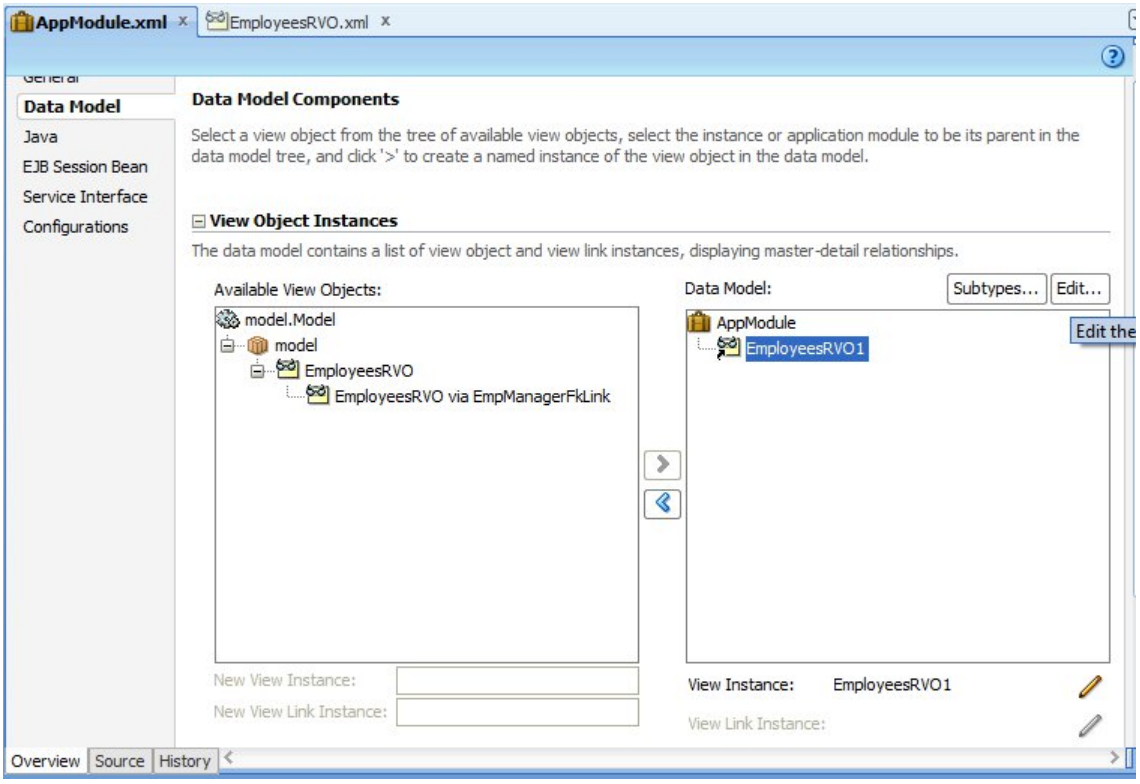


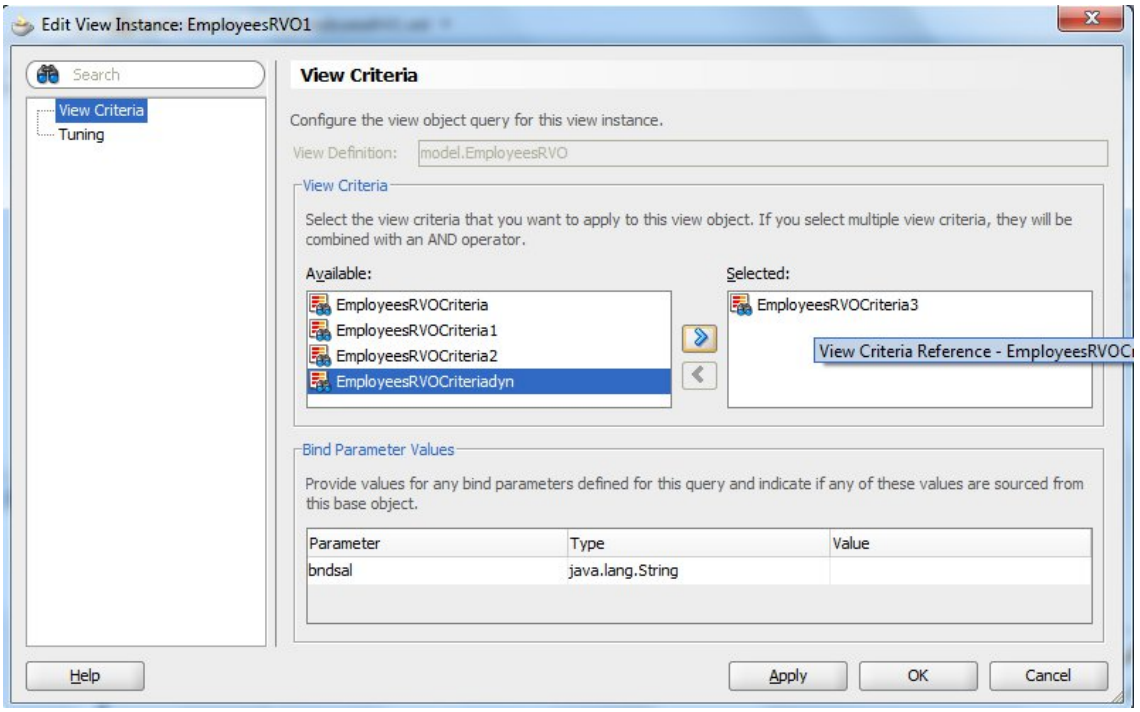
The bind variable is a variable for which we are going to provide the value at runtime. The dynamic view criteria reduces the required no of static view criteria's on the same condition. We can set a validation option either as required or optional for a given bind variable. If we provide validation is required then we must provide the value for that bind variable. If we try to give null values and test the criteria leads to a validation error.

NOTE:-

ViewObjectIMPL, Row, RowSetIterator, ViewCriteria etc.. are the framework provided java classes which are stored in either Oracle.Jbo or Oracle.Jbo.Server packages.

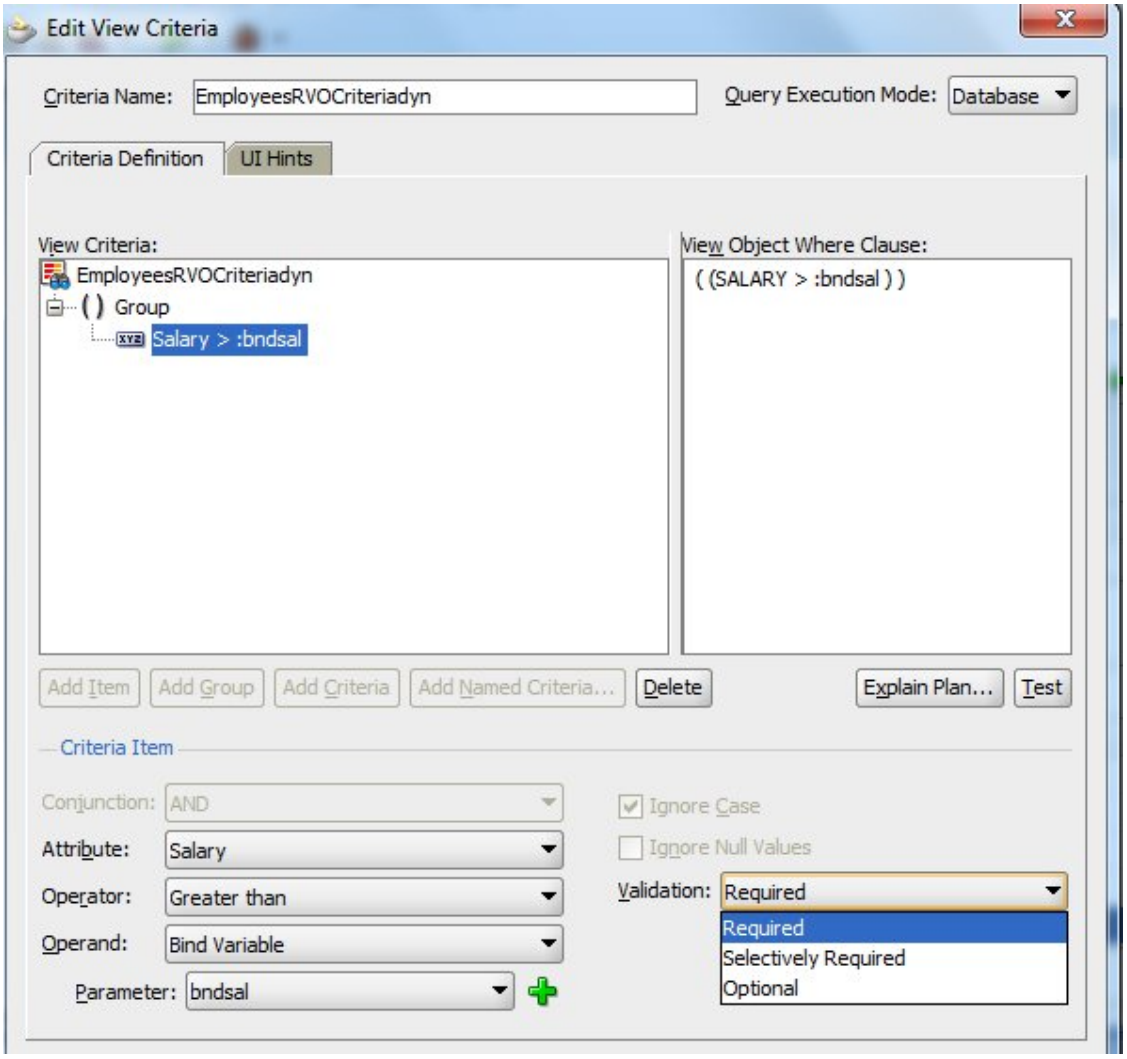
Test the VCs at design time:



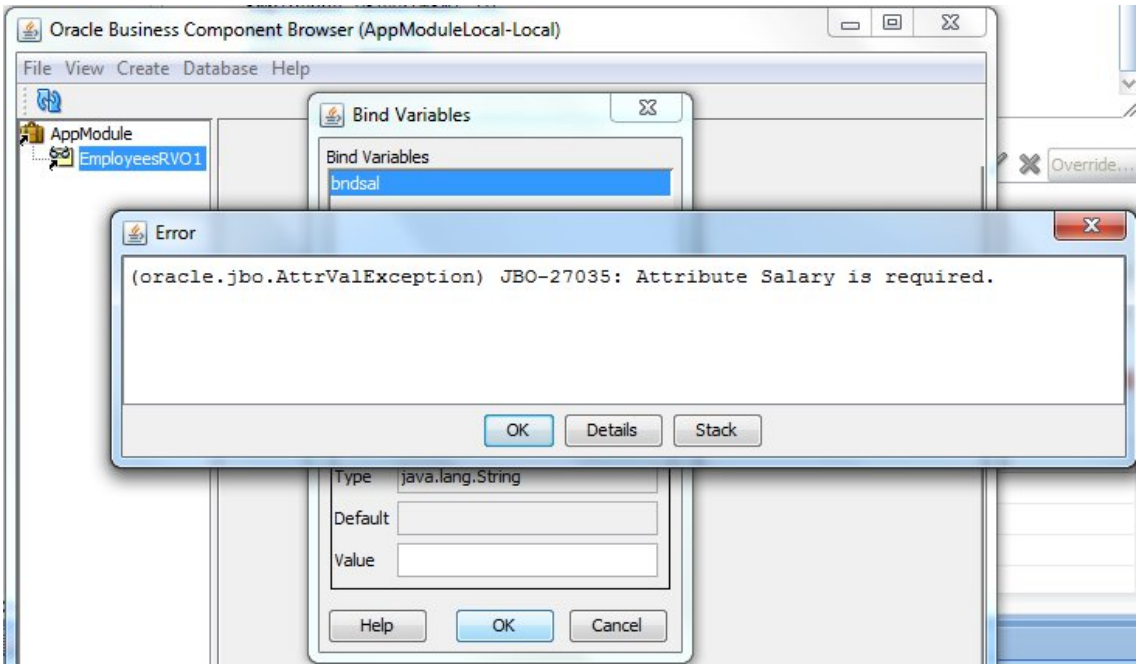


Click on Apply ,OK and Run AM.

Set Validation Required for a Group Item.



Result after providing null value.



2.2.1 Custom methods in AM:-

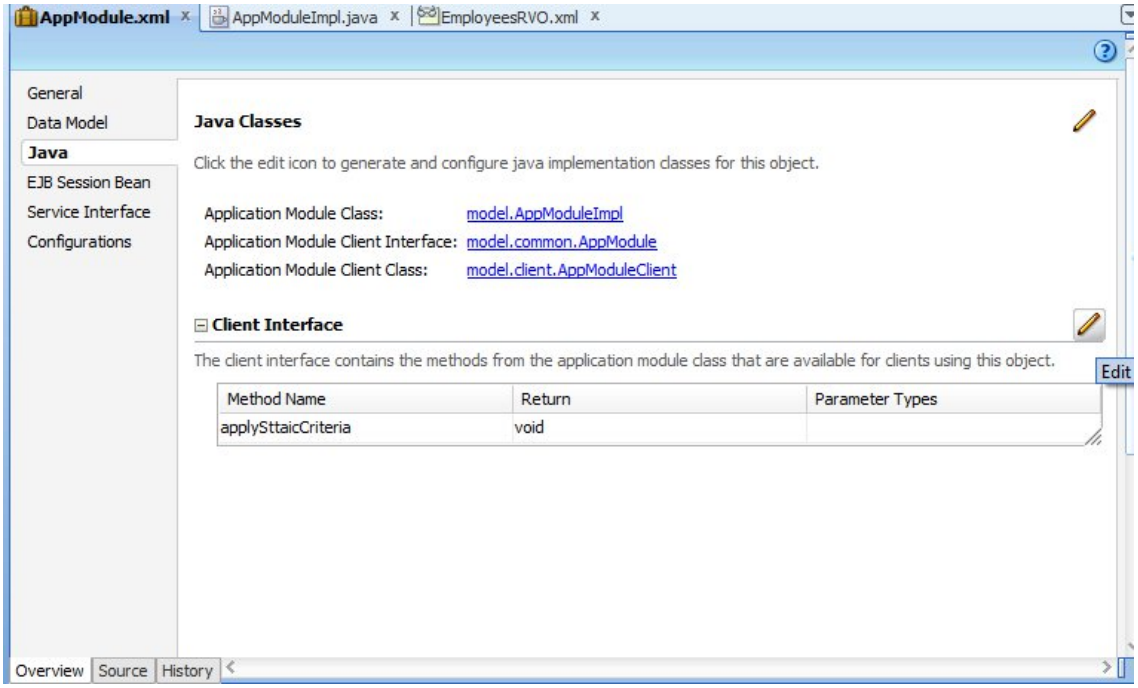
It is a container of VO Instances/View links and other AM Instances. We can configure these instances in Data Model section of AM. When we create an AM, a Data Control with the same name will be created automatically under the Data Controls section. Data Controls section is outcome of the entire Business Layer (ADF BC etc.). AM supports Transaction Context with the help of its Built-In methods (Commit & Rollback). AM provides a Test Browser called BC Test Browser/AM Tester to test all the configured VO Instances and their Built-In methods and AM Built-In methods and custom methods.

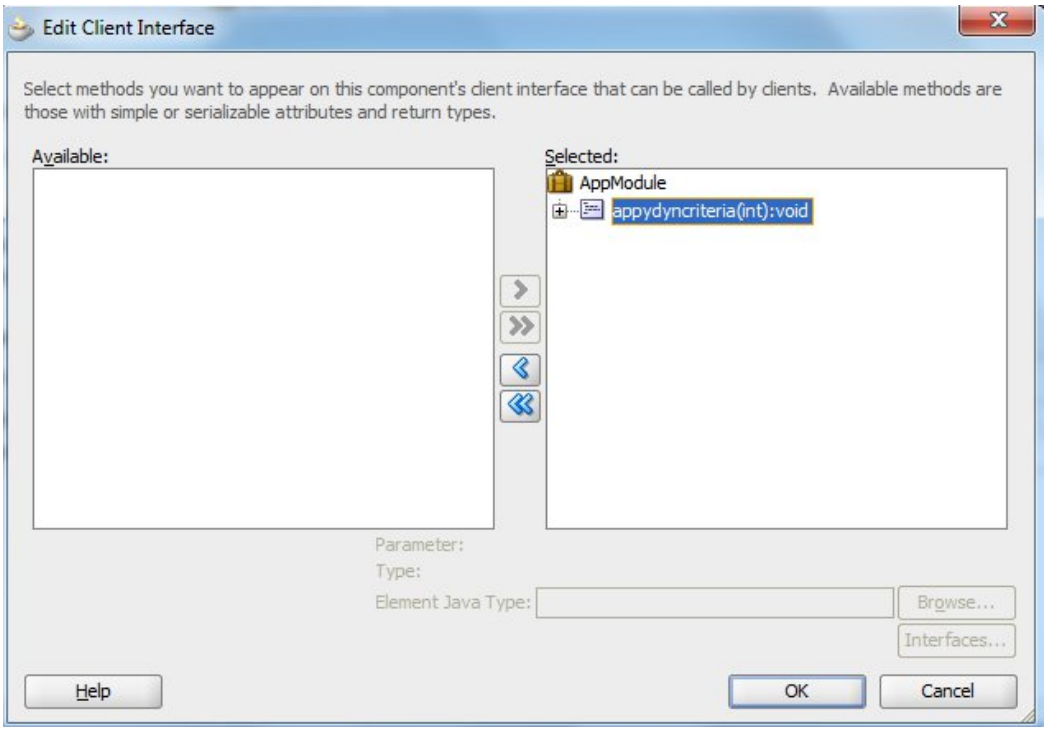
Custom method in AmImpl: applying Dynamic ViewCriteria

```
public void appydyncriteria(int sal) {
    //get vo instance.
    ViewObjectImpl evol = this.getEmployeesRVO1();
    //get the vc
    ViewCriteria vc = evol.getViewCriteria("DYNSAL");
    //set the bind variable value in case if vc is dynamic othrwise skip the below step.
    evol.setNamedWhereClauseParam("BNDSAL", sal);
    // apply vc on vo
    evol.applyViewCriteria(vc);
    //re execute query.
    evol.executeQuery();

}
```

Expose the custom method as Client Interface:-





AM should contain a java class called AMImpl.java. By default it contains all getters of the configured VO Instances. If we want to create any custom methods we can write in the AMImpl.java class and expose those custom methods as client interface. So that the methods are visible or accessible in Data Control section. AM maintains the DB connection information in a configuration file called BC4J.xcfg. An AM contains another AM Instance as its child. Then this process is called Nested AM. In this case parent AM provides Transaction Context (Commit & Rollback) for all its configured VO Instances as well as child AM VO Instances.

3.1-1 Introduction to ADF Bindings

Oracle ADF provides ADF Bindings to form a Model layer. So that the top layers like View, Controller layers can interact with BC layer with the help of this interface. The web pages usually interact with the Business layer with the help of ADF Bindings. ADF Bindings are classified into two parts

- Binding Context
- Binding Container

Each web page should have a page definition file which contains various types of bindings. These bindings are tightly coupled with ADF Faces components to provide the required data.

3.1-2 Types of Bindings

- Tree Binding:-

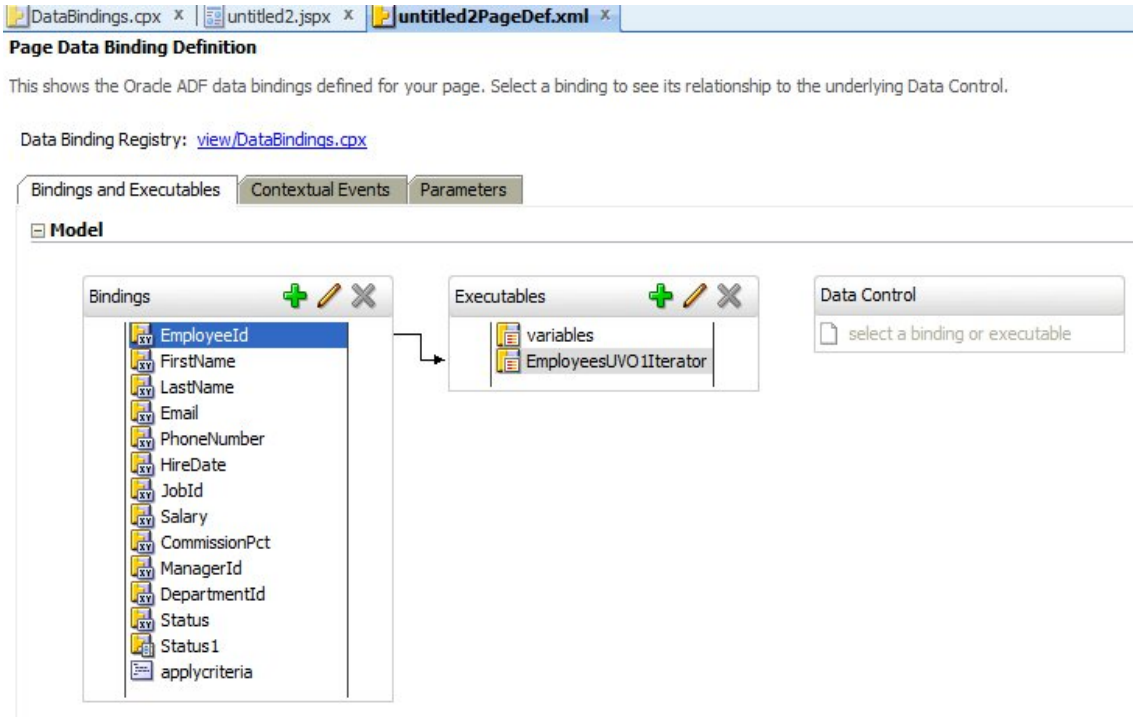
  When we drag and drop the particular VO Instance (from Data Control Section) on to the web page as a table component then Tree Binding will be automatically created in the Binding section of a page definition file.

- Attribute Binding:-

  When we drag and drop a particular VO Instance on to the web page as an ADF Form then Attribute Binding will be created for each attribute in the VO Instance.

- Action Binding :-

  When we drag and drop a built-in method from a VO Instance (Create, first, next, last etc.) and built-in methods of AM (Commit & Rollback) then Action Binding will be created in the binding section of Binding Container.



- Method Action:-

  In general we are going to write our custom methods in AM Impl java class. These methods are exposed as client interface and we can access these from Data Control section. If we drag and drop these on to the page either button or input parameter form then Method Action Bindings will be created in the Page Definition.

- List Binding:-

  When an attribute is have List of Values configured and it is drag and drop on to the web page as select one choice component then List Binding will be created.

- List of Values:-

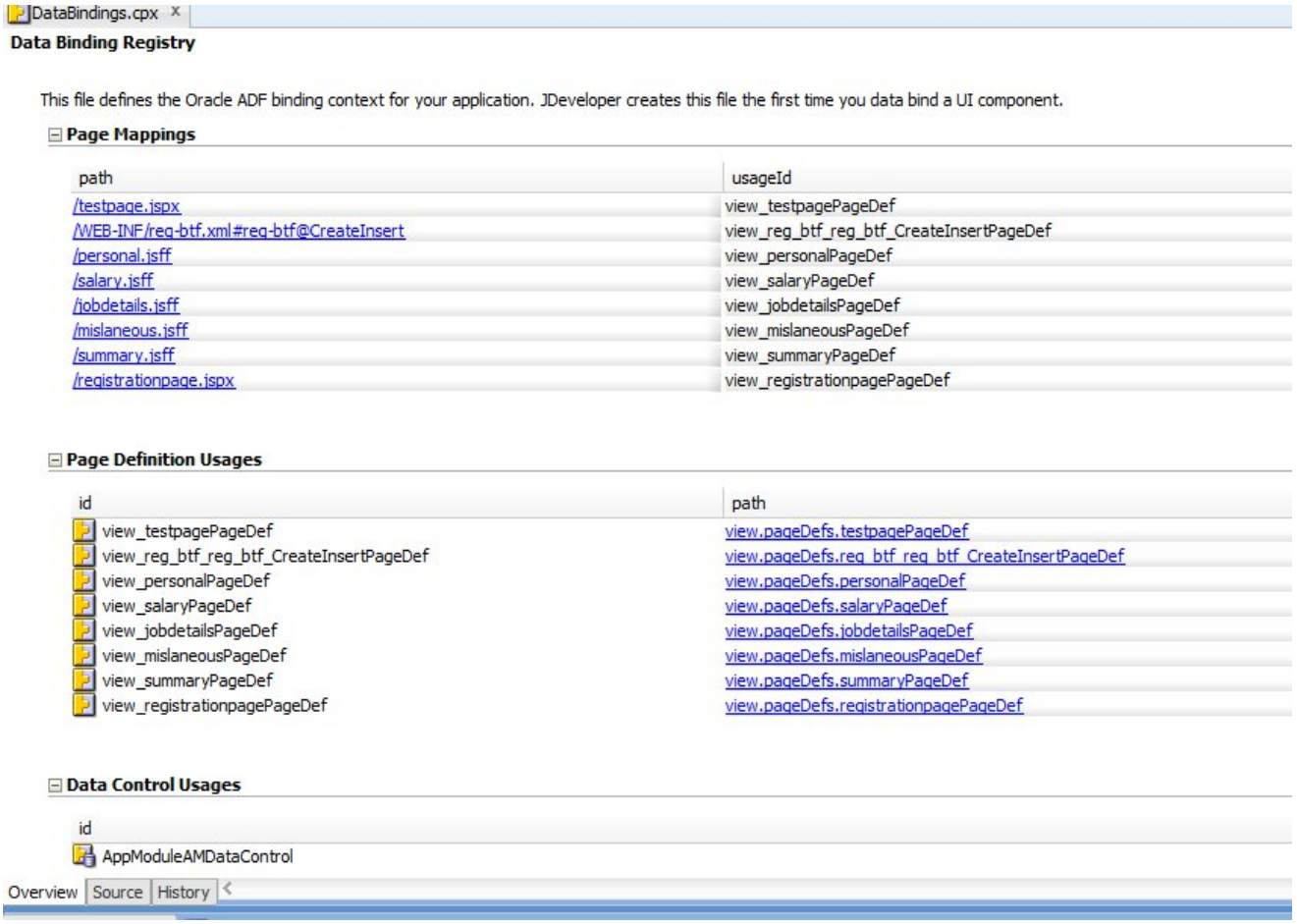
  If we drag and drop the LOV attribute on to the web page as input text with the LOV. Then LOV Bindings will be created in the Binding section of the web page.

- Boolean Binding:-

  If we drag and drop the attribute on to the web page by choosing either select Boolean check box or Boolean radio button then Boolean Binding will be created. At that time we should provide the values for the selected state and unselected states.

3.1-3 Binding Context.

Each page or a page fragment should contain only one page definition file. The Page definition file is also referred as Binding Container. The page/page fragment and its Binding Container mapping along with Data control usage ids will be stored under a configuration file called “DataBindings.cpx”. It is also referred as Binding Context.



The Binding Context will be registered by a configuration file called ADFM.xml. Simply ADF Meta Data configuration file. This file will be loaded by a servlet called ADF Binding Filter Servlet. This servlet is started by a disrupter called web.xml. Once the Binding Container of a running page identified then the contents (Iterators, Query panels) will be executed automatically as these are under executable section of a page definition file. The iterators are the mediators which interacts with which will interact with the configured VO Instances under DC and provide the required data to various types of bindings eg: Tree binding requires a Row set, Attribute binding requires single value List binding requires list of data as column, Boolean binding requires a status value for a checked state and unchecked state. The input for the Model layer is Data Control Items outcome for the Model Layer is the Binding Container.

4. ADF Task Flows

4.1-1 Introduction to ADF TFs

Oracle ADF provides ADF Task flows to form a navigation model for the entire ADF application. ADF Task flows are also called as ADF Page flows or ADF Controller. ADF Task flows are GUI based and we simply develop with the help of various activities and control flows. ADF Task flows are categorized into two types they are

- Bounded Task Flow
- Unbounded Task Flow

4.1-2 TF activities

The ADF Task flow activities are listed below

- View Activity:-

  View Activity refers a web page or page fragment to hold various types of layouts, components etc. A Bounded Task Flow accepts either pages or Page fragments as its View Activity but not accepts a combination. An Unbounded Task Flow Only accepts Pages as its View activity.

- Method call Activity:-

  This activity is used to invoke a custom methods (at ADF BC) or a managed bean method (at VL).

- Task Flow Call Activity:-

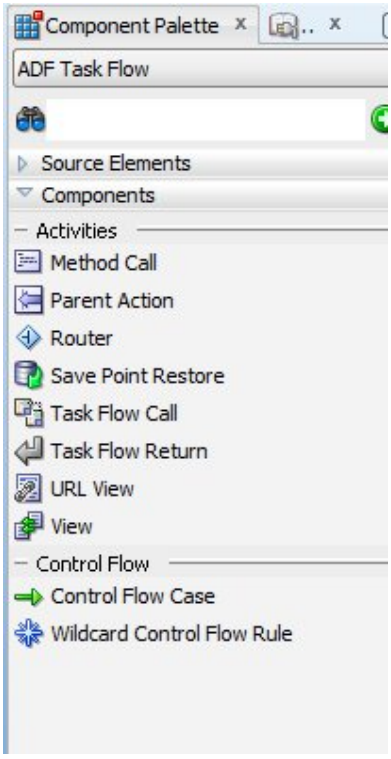
  By using this activity a task flow can call another TF (Child TF) .A Bounded TF called by either Bounded or Unbounded Task Flow. We can't call an Unbounded TF from either Bounded or Unbounded TF.

- Task Flow Return Activity:-

  By using this activity the child TF returns the control to the parent TF.

- Router Activity:-

  By this activity we can validate several expressions and based on the outcome appropriate route (in a TF) will be taken.



- URL View Activity:-

  This activity can make the navigation to either external pages like Google, Facebook or any internal pages (in the Application). For this activity the accurate URL should be provided.

- Parent Action Activity:-

  This activity return some action to its Parent TF and appropriate navigation takes place.

4.1-3 ADF Task Flow Features

- TF have a support of input parameters so that we can provide a communication between two TFs.
- TFs are GUI based.
- TF's are reusable. A Bound TF can be called by either Bounded or Unbounded TF.
- TF have support of Data Control Scopes. The scopes are of two types.

  a) Shared DC Scope
  b) Isolated DC Scope

  These scopes plays vital role when the parent TF and child TF are dealing with same updatable VO Instance and have uncommitted data during navigation. At that time the child TF has to decide from where the data needs to be collected. If child TF configured as Shared DC Scope then the data usually taken from the parent or

calling TF. If DC Scope is configured as Isolated then the data would be taken from the DB tables.

- TF have support of Transaction options to specify the end points for the basic Transaction Context (Commit & Rollback).
- The possible Transaction options are
    - Always begin a new Transaction(ABN)
    - Always use Existing Transaction(AUE)
    - Use Existing Transaction if possible(UEI)
    - No controller Transaction(NCT)
- ADF TF supports security mechanism by enabling ADF TF as resources

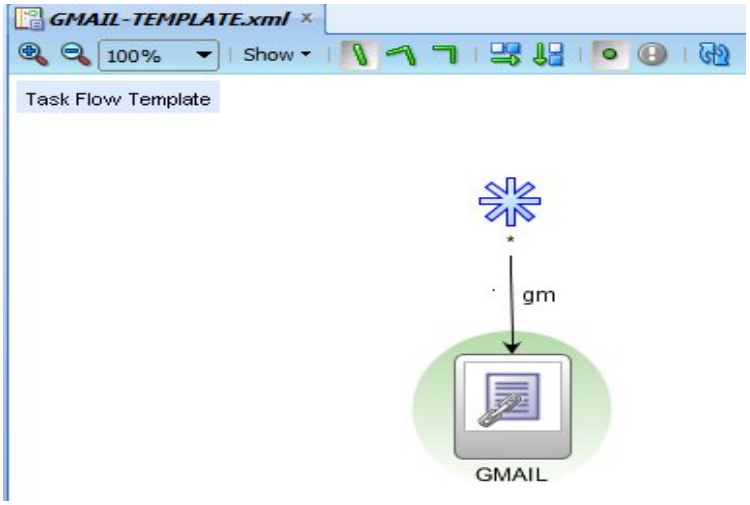
General Process:-

So far we have drag and drop the component on to the web page and we are simply running pages. In general we are going to create page fragments as view activity in the bounded TF and this page fragments contains actual layouts and components. We are going to drag and drop the bounded TF on to the web page as a Region component and we simply run that page.

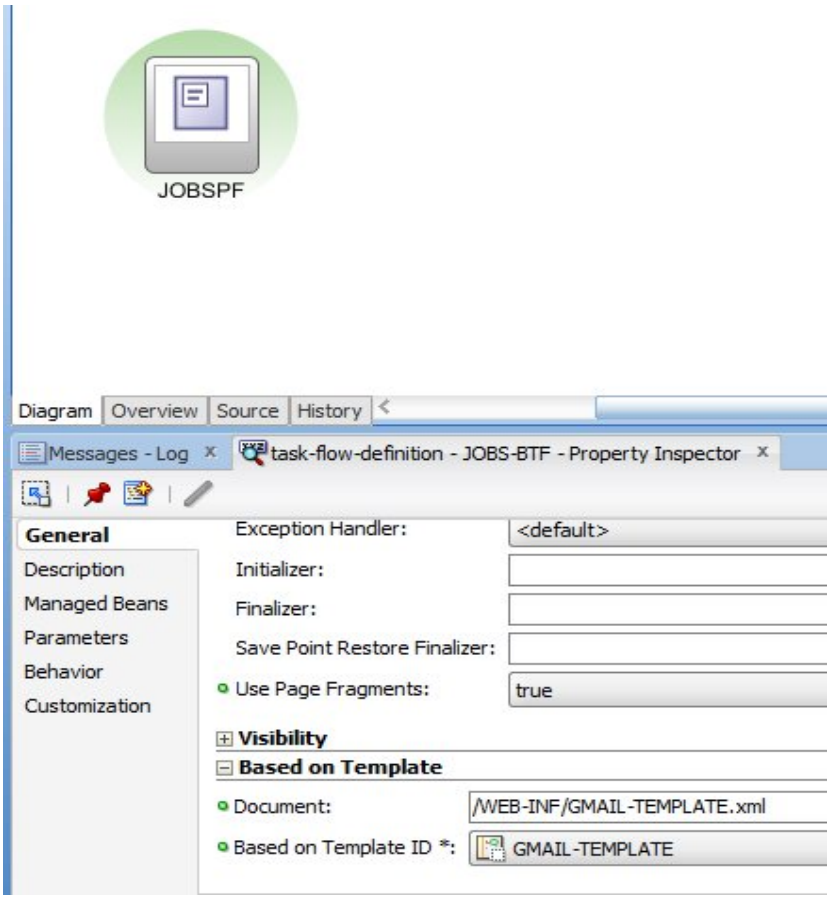
4.1-4 TF Template

TF template is a reusable component which maintains the common navigations across the Task Flows. We can implement those navigations with the help of Wild card (*) Control flow.

Implementation of TF template: Screen I



Extend the TF template in a TF: Screen II

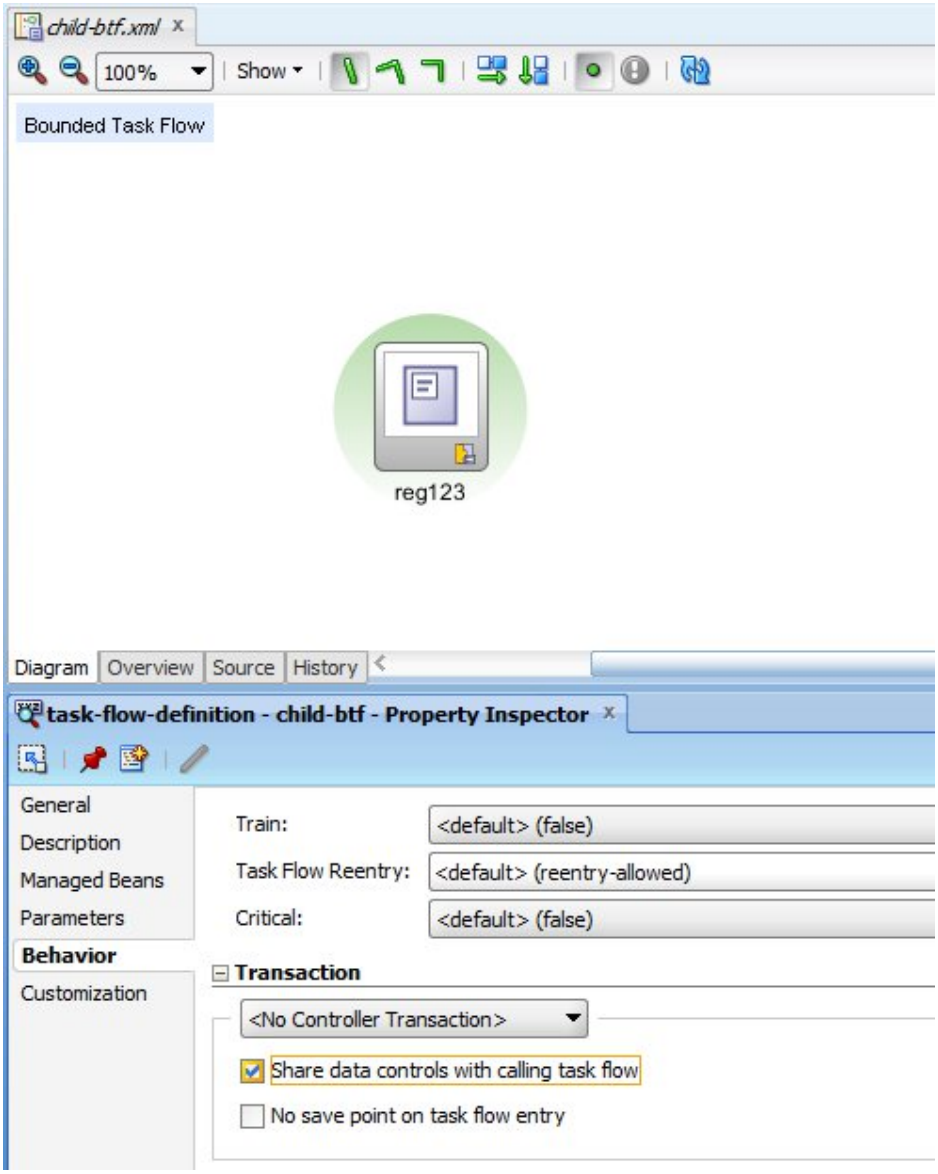


4.1-5 TF Data control scopes

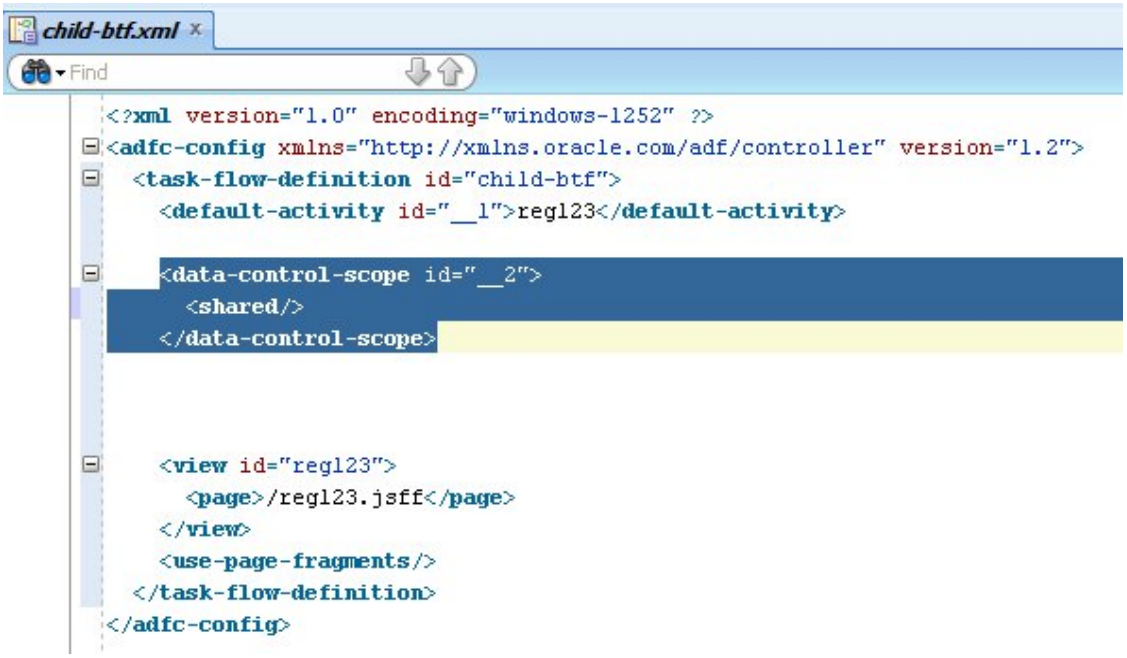
The Data Control Scopes at the TFs are categorized into two types: Shared and Isolated.

Data control scope guides the TF on from where to pick the data. The parent/child TF working on the same updatable VO instances and the navigation happens to the Child TF with uncommitted data from the Parent TF. If the data control scope at Child TF configured Shared then it considers the Parent/Calling TF changes. If the data control scope configured as isolated then the Child TF ignores the parent TF changes and it will take the fresh data from the DB.

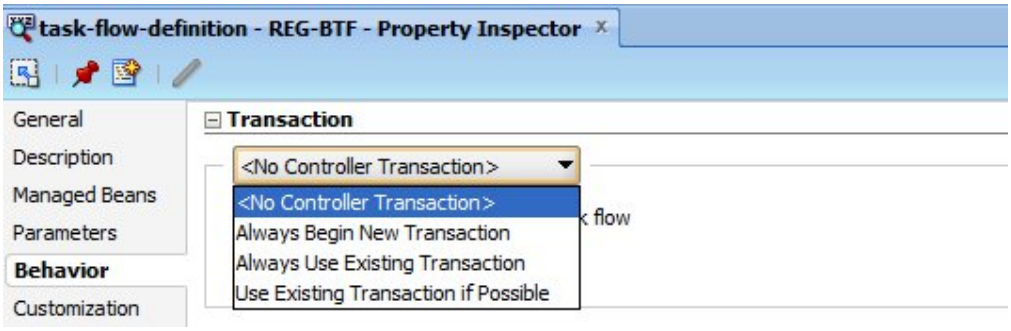
Configuration of Data control scope: Screen-I



Configuration of Data control scope: Screen-II

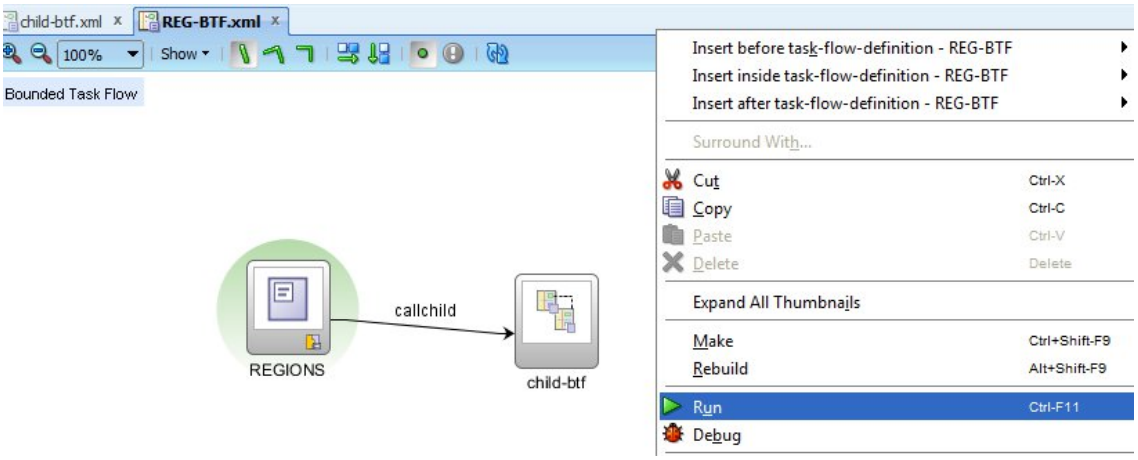


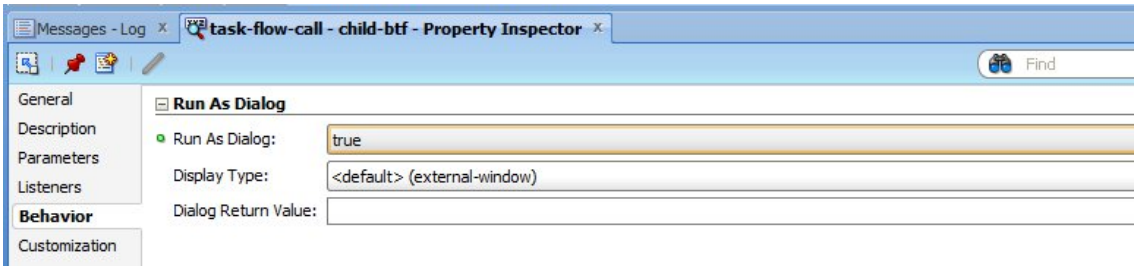
4.1-6 TF Transaction options



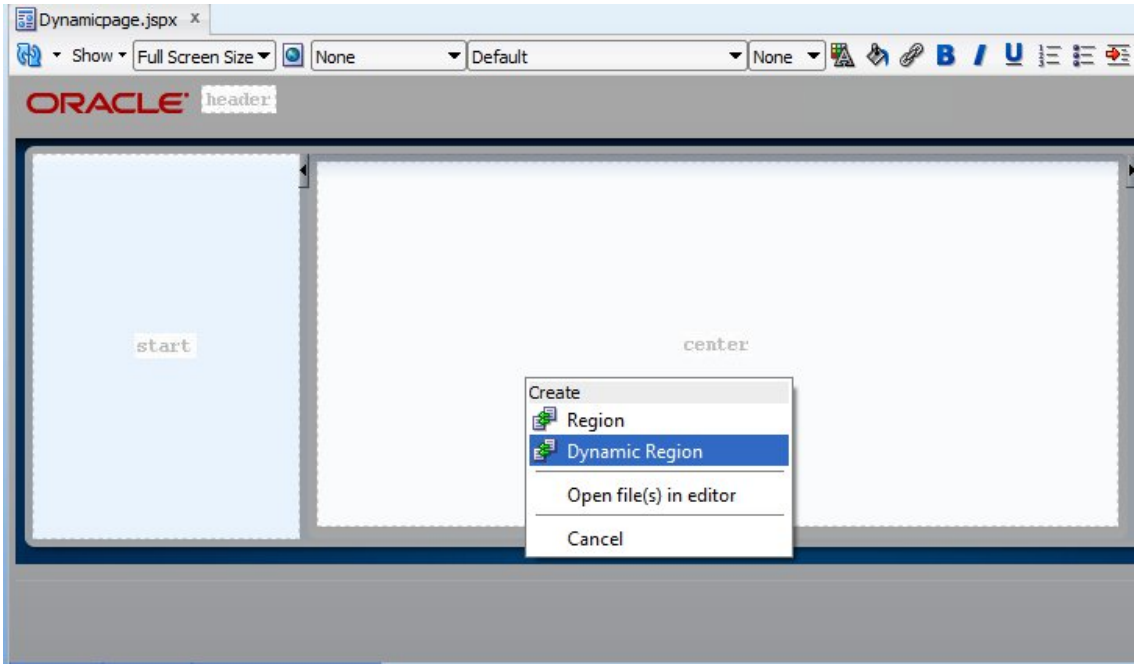
Transaction option at Parent TF	Transaction option at ChildTF	Data Scope at C TF		Result			
NCT	NCT	SHARED		BOTH			
NCT	NCT	ISOLATED		CHILD			
NCT	ABN	SHARED		BOTH			
NCT	ABN	ISOLATED		CHILD			
NCT	AUE	SHARED		ERROR			
NCT	UET	SHARED		BOTH			
NCT	UET	ISOLATED		CHILD			
							NCT-No Controller Transaction
ABN	NCT	SHARED		BOTH			ABN-Always Begin a New Transaction
ABN	NCT	ISOLATED		CHILD			AUE-Always Use Existing Transaction
ABN	ABN	SHARED		ERROR			UEI-Use Existing Transaction If possible
ABN	ABN	ISOLATED		CHILD			
ABN	AUE	SHARED		BOTH			
ABN	UET	SHARED		BOTH			
ABN	UET	ISOLATED		CHILD			
AUE	NCT	SHARED		BLANK			
AUE	NCT	ISOLATED		BLANK			
AUE	ABN	SHARED		BLANK			
AUE	ABN	ISOLATED		BLANK			
AUE	AUE	SHARED		BLANK			
AUE	UET	SHARED		BLANK			
AUE	UET	ISOLATED		BLANK			
UET	NCT	SHARED		BOTH			
UET	NCT	ISOLATED		CHILD			

4.1-7 Run TF and Run TF as a dialog.

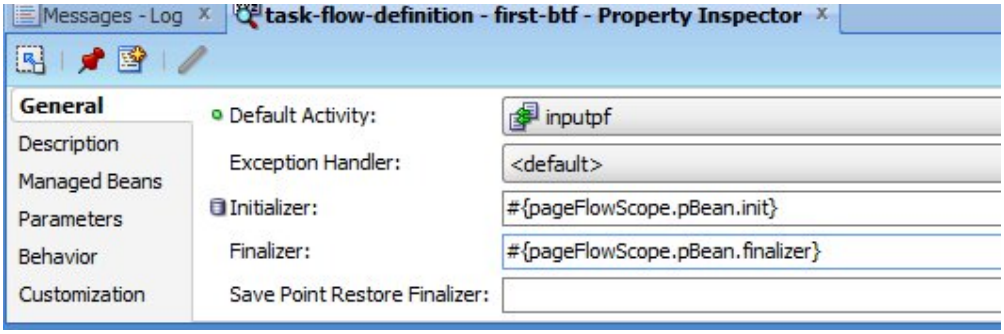




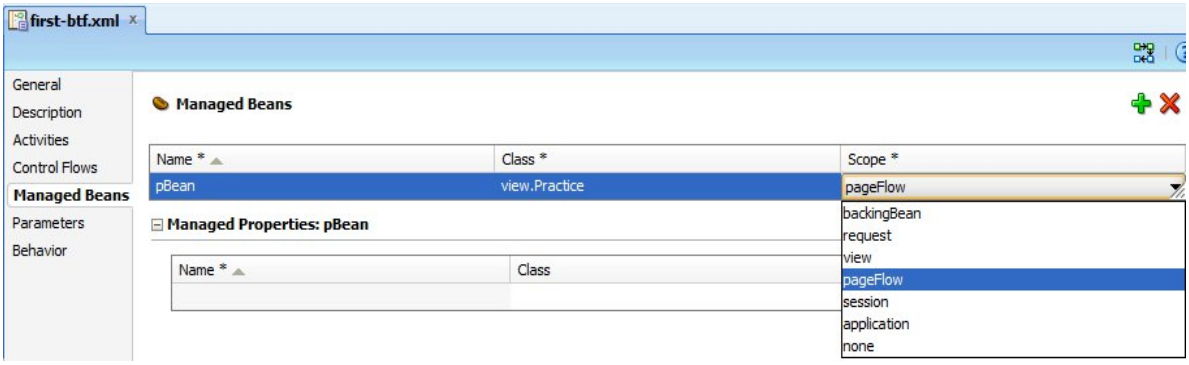
4.1-8 Dynamic Region



4.1-9 Task Flow Initializers/Finalizers

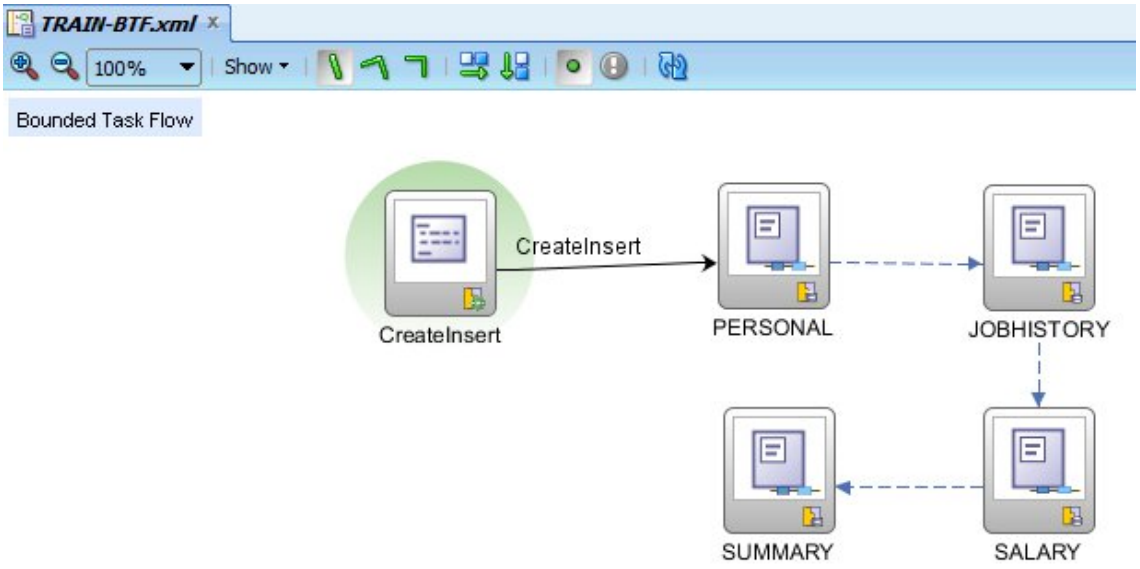


4.2-1 Managed beans in TF



4.2-2 TF Train Components

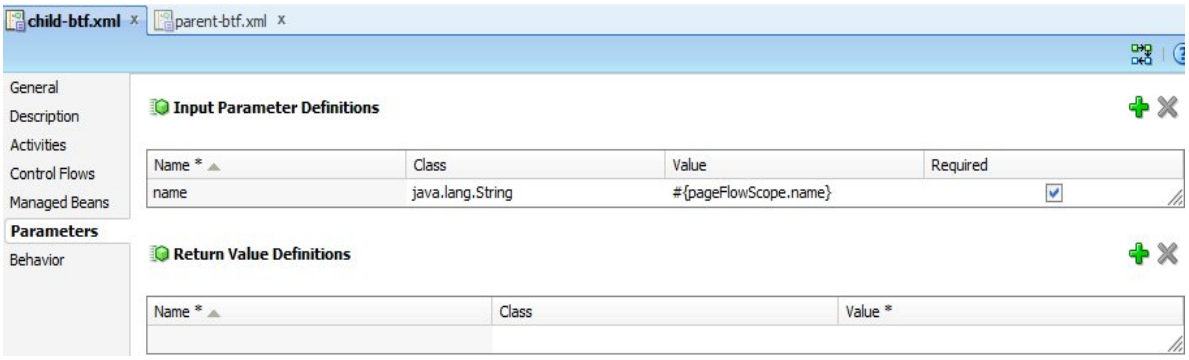
ADF TFs are supported with the Train components to implement the registration page efficiently.



4.2-3 TF Input parameters

We can provide the communication between the two TFs with the help of TF input parameters. Usually the parameters are configured at Child TF. The TF call activity at parent TF provides the data for this input parameter. If the Required option is checked then the parent TF should provide the data for the input parameters. If it is unchecked, it is optional to send data.

Screen-I:- Configure input parameter at child TF



Screen-II:- Configure TF Call activity to pass the value to the child TF input parameter.