Oracle Application Framework

Oracle Application Framework (OAF) is an architecture for creating web based front end pages and J2EE type of applications within the Oracle EBS ERP platform.Oracle Application Framework is the development and deployment platform developed by Oracle to develop the Oracle E-Business Suite “Self-Service” or HTML based Applications.

OA Framework, is based on the industry-standard Model-View-Controller(MVC) design pattern and can be used to develop extensions to existing E-Business Suite functionality.

Oracle Application (OA) enables you to personalize the layout of the user interface and the content it displays to suit your business needs. Oracle’s JDeveloper tool with an Oracle Applications extension is used for development for the Oracle Applications Framework and uses Java & XML languages for the same.

Oracle Application Framework (OA Framework or OAF) is a framework developed by Oracle Corporation for application development within the Oracle E-Business Suite (EBS).

Oracle Application Framework (OAF) provides visual and declarative approaches to Java EE development. The OA framework is also available to customers for personalization’s, customizations and custom-application development.

Oracle Applications Framework is the Oracle Applications development and deployment platform for HTML-based business applications

ORACLE APPS latest version **12.2.7**

80% OAF Pages

20% oracle forms

Features

- – The OA Framework helps to create self Service pages in Oracle EBS.
- – The OA Framework helps in controlling the flow of the application.
- – OA Framework works on the basic Java EE principles.
- – To Learn OAF you must know the Basics of Java like Core Java.
- – User Interface is Very Interactive in OAF
- – OA Framework contains inbuilt security
- – OA Framework is J2EE based but it also supports various standards like HTML, XML, SQL, JSP.

Different Projects

1) Development of OAF pages from scratch (Custom Modules)

2) Customization of OAF PAGES FROM EXISTING Custom application (Custom modules)

3) Extension of OAF PAGES FROM EXISTING STD MODULES (Std modules)

4) Personalization of OAF pages from existing Std modules (Std modules) (without touching the code)

5) Upgradation of OAF pages from one lower version to higher version of oracle apps

OAF Architecture:

Model

a) Model will implement all the DB objects and transactions
b) Model will be implement all the business logic for DB Objects
c) Model will be implement all the BC4J OBJECTS

BC4J – business components for java—directory Structure of all your objects

Path – directory Structure (EO,VO,AM and VL)

BC4J Consist of 3 types

a) Entity object (EO) and Entity Association (AO)

Entity object (EO)

Types of Entity Object:

1) Entity object represents table/view/synonyms and snapshots.

2) All the Entity object naming convention end with EO.

```
Example :  AP_SUPPLIER  –  ApSupplierEO , AP_SUPPLIER_SITES  –
ApSupplierSitesEO
```

3) Performing the DML operations on the page must have to create an entity object.

4) When u create the entity object OAF will create BY DEFAULT two files.

```
 ApSupplierEO.xml     and     ApSupplierEOImpl.java
```

Note: Oracle apps we have custom tables(custom tables provided by client) and std table(provided by oracle)

1) Java based entity object – perform the DML operations on oracle custom tables will go for java based entity object.

```
Example :  Xx_emp_tab ( insert on custome
table )
```

2) PLSQLbased entity object: perform the DML operations on oracle Std tables will go for plsql based entity object.

```
Example :  ap_suppliers ( insert on Std table
)
```

4 Sections

Example: AP_Supplier—SUPPLIER ID, Supplier Name, SupplierType(3 attributes), status, sdate and end date

1) Setter and getter

```
Public number getSupplierId()
{
Return SupplierId
}

Public void setSupplierId()
{
}

Public String getSupplierName()
{
Return SupplierName
}

Public void setSupplierName()
{
}
```

2) Create method Public void create()

```
{
// at the time creation of new record create method will invoke

setSupplierId(“SEQ NAME”);
}
```

3)Remove method

```
Public void remove()
{
// at the delete the record this remove method will
execute

setStuas(“Inactive”);

}
```

4) Validate method Pubic void validate()

```
{
Date dsdate=get sdate();
Date dedate=getedate();

If(dedate.getTime()>dsdate.gettime())
{
}
}
```

Entity Association (AO)

1) Association is relationship between two entity objects having common column (Attribute) between two EO’s

2) All the entity association naming convention end with AO.
Example: SupplierSitesAO

3) When you create entity association OAF will generate 1 file
Example: SupplierSitesAO.xml

4)This association contains SupplierEO.SidSitesEO.sid

5) To perform the DML operations on master and child tables will go for Entity Association
Example: on delete cascade—delete the parent corresponding child also will delete.

On update keys – if u update corresponding child also will update

On update who columns -- (Creation date, created by) --

b) View object (VO)and View links (VL)

1) View objects access all the SQL QUERIES

2) All the view objects naming convention end with VO

```
Example :  EmployeeVO
```

3) When u create the view objects oaf will generate 2 files mandatory and one file is optional

View Objects (VO) :

```
Example :  EmployeeVO.xml (Mandatory) – query
columns will convert into attributes,
attribute Data types, attribute precision
```

Select * from emp;

```
Method 1: Vo.setwhereclause(empid);

Method 2: Vo.setwhereclause(deptid);
```

EmployeeVORowimpl.java(optional) – to process multiple rows

Page1 -- emp full data
Page2—filter the emp data by emp id
Page3 – filter the emp data by dept id

Types of view objects

1)AutomaticVO - always link with EOS(then only u can perform DML operations on the page)

2)ManualVO - Manually u have to write the sql queries(display the DATA AND SEARCH OPERATIONS)

View links (VL):

1) Relationship between two view objects having common column between two VOs
EmpVO – deptno and DeptVO – deptno **Note : Jdeveloper tool have a option to perform join with the help of viewlink**

2) All the view link objects naming convention end with VL .

```
Example :DeptEmpVL
```

3) When u create the view link OAF will generate one xml file

```
DeptEmpVL.xml
```

EmpVO – deptno
DeptVO – deptno

4) Display the master and child data to avoid the duplications

c) Application module (AM)(OADB Transactions, connection open , connection close)

1) DB Connections open and close will be taken care Application module

2) Interface between all the DB objects and transactions with OAF page called as Application module.

3) All the Application module objects naming convention end with AM .

```
Example:  EmployeeAM
```

4) When you create the application module OAF will generate two files.

1) EmployeeAM.xml (It will create the view objects instances in am.xml file)
2) EmployeeAMImpl.java (voimpl and row impl info will be there in am impl.java)

Types of Application modules:

a) Root Application Module: When you attach AM to the main region of the page called as Root application module

b) Nested Application Module: When you attach am to the sub region of the called as Nested application module

```
Note: When you create the page by default one
main region will be creates (default region
for any page), Every region we have property
called AM Definition, If you want to run the
page u need to map the AM to the page.
```

View

1) IS THE OUT PUT OF THE PAGE

2) View is formed with page->region->items(fields)
All the pages naming convention end PG. Example: EMPPG
All the Regions naming convention end with RN Example: EMPRN

3) When u create the page one xml file will be create EMPPG.xml (Region information, field information and mapping AM info will be there)

4) When u create the region one xml file will be createEMPRN.xml (only fields information)

Controller

1.Controller will interact all the web browser activities and navigation

2.Controller will invoke Http get – at the time page open
Http post – button press, link press tab out

3.All the controller objects naming convention end with CO
Ex: EmployeeCO

4. When u create the controller OAF will generate one java file EmployeeCO.java – this java file contains two mandatory methods and one optional method

```
Public void processrequest(OAPageContext, OAWebBean)
{
Http get
}

Public void procesformrequest(OAPageContext x, OAWebBean y)
{
http post
opc
}

Public void Processformdata()
```

Optional method in the controller—I have third-party data (soa service, web service and excel data) to be processed into MY oaf page

Note: In java class methods can pass the class as parameter. Cannot get the methods from the java class directly, create the object(alias) for the class and access the methods from objects.
Packagename.
OAPageContext opc

Two mandatory parameters in controller methods.

OAPageContext – is one of the built in class in OAF

Context values - Org id,User name,User id,Resp name,Resp id.
All the AOL related from EBS will access from OAPageContext java class.

1) To get the apps context values.
Opc.getUserName() (void method or data type method)
Opc.getorgid()
Opc.getrespname()

2) To get the value non db value from the page.
SupplierName
Opc.getParameter(“SupplierName”);

3) Call one page to another page.
Page1 --> page2
Opc.setforwardurl(“page2 url”);

OAWebBean -- is one of the built in class in OAF

OAWebBean:owb
Bean = item = field
Represents the properties of the Items (fields)
25+ item styles are there in OAF
CHECK BOX, RADIO BUTTON, BUTTON, TEXT INPUT ETC

Bc4j file structures:

Model will be implement all the BC4J OBJECTS
bc4j – business components for java—directory Structure of all your objects
Path – directory Structure (EO,VO,AM and VL)

BC4J Consist of 3 types
a) Entity object (EO) and Entity Association (AO)
b) View object (VO)and View links (VL)
c) Application module (AM)(OADB Transactions, connection open , connection close)

AM AND VO -> BC4J XXAAM.oracle.apps.po.xxname.server
[custom top client name].oracle.apps.[application short name].[componenet name].server

Entity Object(EO) -> BC4J
XXAAM.oracle.apps.po.xxname.schema.server

LOV ->BC4J
XXAAM.oracle.apps.po.xxname.LOV.server

Poplist ->BC4J
XXAAM.oracle.apps.po.xxname.poplist.server

Controller & Page
XXAAM.oracle.apps.po.xxname.webui