# ADF BC

## Entity Object

|  |
| --- |
| Ingredients    EO XML metadata file   * Stores the db object * Attributes and validators * View and Entity Accessors * Business Event defintions   EO Definition  Runtime java class representation of the XML metadata file. Based on  oracle.jbo.server.EntityDefImpl. Can be customized.  Enity Object  oracle.jbo.server.EntityImpl  **Entity collection**  oracle.jbo.server.EntityCache  Enity objects are Table or View based |

Resource Bundle

|  |
| --- |
| To get a localized value basd on a key from EO  //Access the ResourceBundleDef defined for the entity object  ResourceBundleDef resourceDef = this.getResourceBundleDef();  //Get the user locale  Locale locale = this.getDBTransaction().getSession().getLocale();  //Get the localized value for the key  String retVal = StringManager.getLocalizedStringFromResourceDef(resourceDef, key, null, locale, null, false);  //Get the user locale  Locale currentLocale = this.getDBTransaction().getSession().getLocale();  ResourceBundle myResources = ResourceBundle.getBundle("MyResources", currentLocale); |

Setting an attribute to sequence value using expression

(new oracle.jbo.server.SequenceImpl("EMPLOYEES\_SEQ",adf.object.getDBTransaction())).getSequenceNumber()

Effective Dated Entity Objects

Similar to keeping audit trail for all changes on this EO

empEntityImpl.setEffectiveDateMode(Row.EFFDT\_UPDATE\_CHANGE\_INSERT\_MODE );

empEntityImpl.setSalary(10000);

getDBTransaction().commit();

Find By Primary Key

//This method finds the employee entity instance for employee id

**private EmployeeEOImpl findEmployeeById(int empId) {**

//Find the entity definition from the implementation class

**EntityDefImpl orderDef = EmployeeEOImpl.getDefinitionObject();**

//Create the Key object

**Key orderKey = EmployeeEOImpl.createPrimaryKey(new Integer(empId));**

//Find and return the desired instance

**return (EmployeeEOImpl)orderDef.findByPrimaryKey(getDBTransaction(), orderKey);**

}

The polymorphic discriminator type is introduced to simulate inheritance hierarchies in entity and view objects

Use an attribute as discriminator

EntityImpl empEOImpl = empBaseEODef.findByPKExtended

(getDBTransaction(), empIdKey, true);

**Refresh on Insert** or **Refresh on Update**

**Change Indicator – Data Consistency**

**Track Change History**

**Need to have columns**

* **Created On**: This attribute is populated with the time stamp of when the entity instance was created.
* **Created By**: This attribute is populated with the name of the user who created the entity instance.
* **Modified On**: This attribute is populated with the time stamp of when the entity instance was modified.
* **Modified By**: This attribute is populated with the name of the user who modified the entity instance.

findByAltKey()

//Find employee entity definition

EntityDefImpl empEODef = EmployeeEOImpl.getDefinitionObject();

String email= "jobinesh@xyz.zom";

//Email has been defined as alt key on entity

Key emailAltKey = new Key(new Object[] { email });

//Find by alt Key 'EmailAltKey' on employee entity definition

EntityImpl employeeEO = empEODef.findByAltKey(getDBTransaction(), "EmailAltKey", emailAltKey, false, true);

Validations

* Attribute Level

public boolean validateName(String name) {

if (name != null && name.trim().length() == 0) {

Return false;

}

return true;

}

* Entity Level  
  public boolean validateXXX(ArrayList ctxList)
* Transaction Level

Select the **Defer execution to Transaction Level** option displayed under the **Validation Execution** tab.

However, for a validation cycle, validations defined at transaction level will be triggered only once for an entity type. In other words, it is not triggered for each modified instance in the entity cache.

#### Configuring Locking Mode for an ADF application

**Per application module level**

jbo.locking.mode

**Application level**

adf-config.xml

Click on the **Business Components** tab on the **adf-config.xml** editor window and set **Locking Mode**

jbo.locking.mode  
**None -** This implies the case where no locking is done when the user updates a row.

**Pessimistic -**  pessimistic locking mode will try to lock an entity instance in advance before allowing any changes by explicitly firing SELECT FOR UPDATE

**Optimistic –** The optimistic locking mode will try to lock the entity before posting modified data to the database, followed by a check for stale data

**Optupdate –** The optupdate locking mode will not issue lock on the row, it will just see if all row attributes that are participating in the update have the latest value from the database table.

//Find the employee entity using ID

EmployeeEOImpl empEO= findEmployeeById(1000) ;

//Modifies the entity name

empEO.setName("John");

//Commit the transaction

getTransaction().commit();

empEO.setName("John");

getTransaction().postChanges();

or

getTransaction().commit();

**Security**

Click on the **General navigation** tab, expand the **Security** section, and select the desired operation—read, update, or remove. These operations are defined by the oracle.adf.share.security.authorization.EntityPermission class and will be used by the ADF security to authenticate the user.

Programmatically creating Entity Instance

//Application Module

public class HRServiceAppModuleImpl extends ApplicationModuleImpl implements HRServiceAppModule {

public EmployeeEOImpl createEmployee() {

//Get the EmployeeEODefImpl which is the

//java representation of EmployeeEO.xml

EmployeeEODefImpl employeeEODefImpl = (EmployeeEODefImpl)EmployeeEOImpl.getDefinitionObject();

//Create the entity instance in the current transaction

EmployeeEOImpl newEmployee = (EmployeeEOImpl)employeeEODefImpl.createInstance2(this.getDBTransaction(), null);

//EmployeeEO uses EmployeeID as PK

//EmployeeId is the PK

newEmployee.setEmployeeId(1000);

newEmployee.setFirstName("Jobinesh");

newEmployee.setLastName("Purushothaman");

newEmployee.setDepartmentId(10);

newEmployee.setEmail("JOBINESH@XYZ.COM");

newEmployee.setHireDate(new Timestamp(System.currentTimeMillis()));

newEmployee.setJobId("IT\_PROG");

try {

//Commit the transaction

getDBTransaction().commit();

} catch (JboException ex) {

//If commit fails, then roll back the entire transaction

getDBTransaction().rollback();

throw ex;

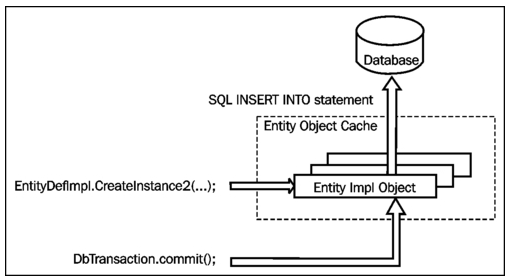
}

return newEmployee;

}

// Other methods go here

}



EO in transaction post cycle

//Get the Entity Definition

EntityDefImpl empEODef = EmpEOImpl.getDefinitionObject();

//Generate Key using empId parameter

Key empIdKey = EmpEOImpl.createPrimaryKey(new Integer(empId));

//Find Emp entity using Key

EmpEOImpl empEO= (EmpEOImpl) empEODef.findByPrimaryKey(getDBTransaction(), empIdKey);

//Update the attribute

empEO.setFirstName("Jobinesh");

//Commit transaction

getDBTransaction().commit();

## Associations

* **One to One**: An entity instance is associated to a single instance of another entity
* **One to Many**: An entity instance is associated to multiple instances on another entity
* **Many to One**: Multiple instances of an entity are associated to one instance of another entity
* **Many to Many**: Multiple instances are participating in the association at both the ends

To programmatically access the destination entities

EntityDefImpl deptEODef = DeptEOImpl.getDefinitionObject();

//Find Creates the Key to find Department

Key deptIdKey = DeptEOImpl.createPrimaryKey(new Integer(deptId));

//Find the Department entity using deptId

DeptEOImpl deptEOImpl = (DeptEOImpl)departmentEODef.findByPrimaryKey(getDBTransaction(), deptIdKey);

//Access Employees for this depertament using association

//accessor getEmpEO() generated on DeptEOImplclass

RowIterator rowIter= DeptEOImpl.getEmpEO();

while(rowIter.hasNext()){

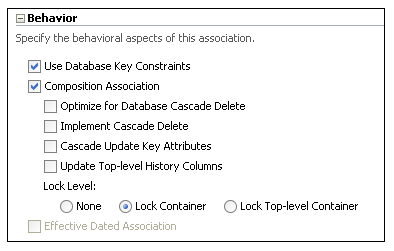
Row row=rowIter.next();

//Row represent Emp entity instance

//Business logic goes here

}

#### Composition association



## View Object

|  |
| --- |
| **Adding CountryName in LocationsVO**   1. We need to fetch CountryName from Countries and display it in Locations 2. Create Association and ViewLink between Locations and Countries on CountryId column 3. In the LocationsVO add CountriesEO as the associated entity. 4. Then add CountriesVO and a child of LocationsVO in AM 5. In the LocationsVO add the CountriesEO as another entityobject 6. Then add the attribute CountryName from that EO |

View Object Ingredients

* View Definition XML file –
  + SQL select query, Bind variables, Attributes, Entity Usage information, LOV, View Accessor, Business Rules
* View Definition - oracle.jbo.server.ViewDefImpl
* View Object - oracle.jbo.server.ViewObjectImpl
* View Criteria
* Bind Variables
* View Accessor
* Row
* Row Set - View object uses row sets to manage the collection of rows from a query.
* Row Set Iterator
* Query Collection
* View Link

You can also specify the **Updatable**, **Reference**, and **Delete Participant** properties for entity usages:

* **Updatable**: This property decides if the entity object attributes can be modified through view object. By default, the first entity object is updatable and subsequent ones are read-only.
* **Reference**: Select this option if you want the information from this entity object to be treated as read-only reference information in this view object. Attribute values are dynamically fetched from the entity cache when a joining key attribute changes.

Creating Row using Name Value Pair

//In application module implementation class

public void createOnPolymorphicVO(){

Row row=null;

NameValuePairs nvp=null;

//Get the polymorphic VO

MarketingEmployeeListVOImpl vo =

(MarketingEmployeeListVOImpl)

getMarketingEmployeeList(); nvp = new NameValuePairs();

nvp.setAttribute("DepartmentId", "90");

//VO delegates the createAndInitRow() call to

//SalesEmployeeExEO as DepartmentId=90

row=vo.createAndInitRow(nvp);

nvp = new NameValuePairs();

nvp.setAttribute("DepartmentId", "80");

//Vo delgates the createAndInitRow() call to

//ExecutiveEmployeeExEO as DepartmentId=80

row=vo.createAndInitRow(nvp);

}

Query Mode for a VO

public void changingQueryModeDynamically() {

//Possible view object modes are

//1. ViewObject.QUERY\_MODE\_SCAN\_DATABASE\_TABLES

//2. ViewObject.QUERY\_MODE\_SCAN\_VIEW\_ROWS

//3. ViewObject.QUERY\_MODE\_SCAN\_ENTITY\_ROWS

//Get the view object

ViewObject vo = findViewObject("EmployeeDetails");

//Fires query against database(default query mode)

vo.executeQuery();

//Rest of the business logic goes here..

//This e.g just prints all rows

//In real life application, you can process the rows here

while (vo.hasNext()) {

Row r = vo.next();

String email = (String)r.getAttribute("Email");

System.out.println("Employee with email -" + email);

}

//Now find all employees whose last name start with 'P'

//This is done against the rows in the cache(no DB hit)

vo.setQueryMode(ViewObject.QUERY\_MODE\_SCAN\_ENTITY\_ROWS);

ViewCriteria vc = vo.createViewCriteria();

ViewCriteriaRow vcr1 = vc.createViewCriteriaRow();

vcr1.setAttribute("LastName", "LIKE 'P%'");

vc.setCriteriaMode(ViewCriteria.CRITERIA\_MODE\_CACHE);

vc.add(vcr1);

vo.applyViewCriteria(vc);

vo.executeQuery();

}

// Defined In custom ApplicationModuleImpl

public void queryUsingMulitpleQueryModes(){

//Find the view object

ViewObject employeeVO = findViewObject("EmployeeDetails");

//Combine both database mode and entity cache query mode

employeeVO.setQueryMode(

ViewObject.QUERY\_MODE\_SCAN\_DATABASE\_TABLES |

ViewObject.QUERY\_MODE\_SCAN\_ENTITY\_ROWS);

//Execute query

employeeVO.executeQuery();

// Business logic to manipulate row goes here

}

Appending a where clause at runtime

//This is in application module implementation class

public void executeEmployeeDetailsVOWithBindVar(){

ViewObject vo = findViewObject("EmployeeDetails");

//Append WHERE cluase

vo.setWhereClause("FirstName = :bindVarFirstName");

//Define the bind variable

vo.defineNamedWhereClauseParam("bindVarFirstName", null,

null);

//Set the bind param value

vo.setNamedWhereClauseParam("bindVarFirstName","Tom");

vo.executeQuery();

}

Bind Variable

//In application module implementation class

public Row findEmployeeForEmpId(Integer empId){

ViewObject employeeVO = findViewObject("Employees");

//Define WHERE clause

String whereClause = "EmployeeEO.EMPLOYEE\_ID = :empId";

employeeVO.setWhereClause(whereClause);

employeeVO.setNamedWhereClauseParam("empId", empId);

employeeVO.executeQuery();

return employeeVO.first();

}

VariableValueManager

//In application module implementation class

public void findEmployeeForEmplId(Integer empId){

ViewObject employeeVO =findViewObject("Employees");

VariableValueManager vm =

employeeVO.ensureVariableManager();

vm.setVariableValue("empId", empId);

employeeVO.executeQuery();

}

setWhereClauseParam()

//In application module implementation class

public Row findEmployeeForEmpId(Integer empId){

//Define WHERE clause with JDB style bind var :NNN

String whereClause = "EmployeeEO.EMPLOYEE\_ID = :1";

ViewObject employeeVO =findViewObject("Employees");

employeeVO.setWhereClause(whereClause);

employeeVO.setWhereClauseParam(1,empId);

employeeVO.executeQuery();

return employeeVO.first();

}

### CHANGING THE QUERY OF A VIEW OBJECT AT RUNTIME

Before you alter the query dynamically, make sure you close all existing row sets that are using the existing query

/\*\*

\* This is a custom method defined in an application module.

\* It sets the query on a view object at runtime

\*/

public void modifyQueryDynamically(ViewObjectImpl voImpl, String newQueryString) {

// FULLSQL\_MODE\_AUGMENTATION flag that tells framework that

// you want to keep the query set through setQuery() for

// the rest of the execution using this view object

voImpl.setFullSqlMode(voImpl.FULLSQL\_MODE\_AUGMENTATION);

closeAllRowsets(voImpl);

//clear the previous WHERE clause and parameters

voImpl.setWhereClause(null);

voImpl.setWhereClauseParams(null);

//Query can be set fully in expert mode

voImpl.setQuery(newQueryString);

}

/\*\*

\* Close all row set when query changes

\* to avoid the orphan Rowset

\*/

public void closeAllRowsets(ViewObject vo) {

RowSet[] rowSets = vo.getRowSets();

if (rowSets == null || rowSets.length == 0) {

return;

}

for (RowSet rowSet : rowSets) {

rowSet.closeRowSet();

}

}

Sorting

ViewObject vo = findViewObject("EmployeeDetails");

//FullName is a transient attribute

vo.setSortBy("FullName desc");

vo.executeQuery();

ViewObject vo = findViewObject("EmployeeDetails");

vo.setOrderByClause("EmployeeEO.FIRST\_NAME ASC");

vo.executeQuery();

In-memory filtering with RowMatch

//Get row to be checked from employee view object

ViewObject vo = findViewObject("EmployeeDetails");

Row row = vo.getCurrentRow();

//Set the condition for in memory filtering

RowMatch rm = new RowMatch("FirstName = 'William'");

// alternatively use rm.setWhereClause( condition );

if (rm.rowQualifies(row)) {

//Row exists with William as first name

}

//Get employee view object

ViewObject vo = findViewObject("EmployeeDetails");

//Set the condition for in memory filtering

RowMatch rm = new RowMatch("FirstName like 'J%'");

vo.setRowMatch(rm);

vo.executeQuery();

//Get employee view object

ViewObject vo = findViewObject("EmployeeDetails");

RowMatch rm = new RowMatch("LastName = :bvLastName");

vo.getVariableManager().setVariableValue("bvLastName", "Grant");

//Get row to be checked from employee view object

ViewObject vo = findViewObject("EmployeeDetails");

Row row = vo.getCurrentRow();

RowMatch rm = new RowMatch("FirstName = :bvFirstName");

ViewObject ownerVO=(ViewObject)row.getStructureDef();

ownerVO.getVariableManager().setVariableValue

("bvFirstName", "William");

Row row=getRow

if(rm.rowQualifies(row)) {

//Row exists with William as last name

}

RowMatch rm = new RowMatch("UPPER(FirstName) = UPPER(:bvFirstName)");

//In view object implementation

@Override

protected boolean rowQualifies(ViewRowImpl vr) {

//Omit the rows from row set whose StatusFlag= 'DELETE'

Object attrValue =vr.getAttribute("StatusFlag");

if (attrValue != null) {

if ("DELETE".equals(attrValue))

return false;

else

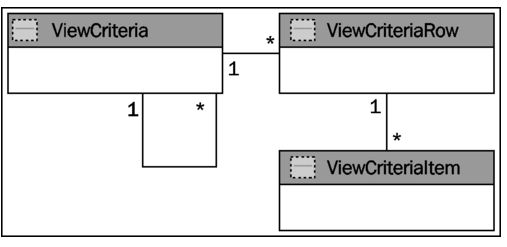
return true;

}

return super.rowQualifies(vr);

}

### VIEW CRITERIA



* **Database**: In this mode view criteria is used for database query. This is the default mode. You can use the ViewCriteria.CRITERIA\_MODE\_QUERY constant to refer this mode programmatically.
* **In-memory**: In this mode, view criteria is used for in-memory filtering. You can use theViewCriteria.CRITERIA\_MODE\_CACHE constant to refer this mode programmatically.
* **Both (database and in-memory)**: In this mode, view criteria is used for a database query first and then for in-memory row filtering. You can combine the ViewCriteria.CRITERIA\_MODE\_QUERYand ViewCriteria.CRITERIA\_MODE\_CACHE constants in order to refer this mode programmatically:

ViewCriteria.CRITERIA\_MODE\_QUERY | ViewCriteria.CRITERIA\_MODE\_CACHE

public void executeEmployeeVOWithDynamicVC(){

//Get the desired view object

ViewObject vo = findViewObject("EmployeeDetails");

//Build View Criteria first

ViewCriteria vc = vo.createViewCriteria();

//Create View Criteria Row

ViewCriteriaRow vcr = vc.createViewCriteriaRow();

//Set the QBE

vcr.setAttribute("FirstName","LIKE 'J%'");

//Optionally override the default criteria mode-Database

vc.setCriteriaMode(ViewCriteria.CRITERIA\_MODE\_QUERY |

ViewCriteria.CRITERIA\_MODE\_CACHE);

//Add View Criteria Row

vc.add(vcr);

//Execute Query, Framework will generate a WHERE clause

//for the above VC as

//SELECT … FROM … WHERE EmployeeEO.LAST\_NAME LIKE

//:vc\_temp\_1

vo.executeQuery();

}

View Criteria with Bind Variable

public ViewCriteria CreateVCForEmpName(){

//Get the VariableValueManager who is responsible for

//managing bind vairbales

VariableValueManager vvm =ensureVariableManager();

//Create View Criteria

ViewCriteria vc =createViewCriteria();

ViewCriteriaRow vcr = vc.createViewCriteriaRow();

//Create View Criteria Item and set operator

ViewCriteriaItem vci = vcr.ensureCriteriaItem("FirstName");

vci.setOperator(JboCompOper.OPER\_LIKE);

vci.setRequired(ViewCriteriaItem.VCITEM\_REQUIRED);

//Define bind variable and init properties

VariableImpl fstNameVar = (VariableImpl)vvm.

addVariable("dynamicBindVarAttribute");

fstNameVar.setJavaType(String.class);

fstNameVar.setMandatory(true);

fstNameVar.setUpdateableFlag(Variable.UPDATEABLE);

fstNameVar.setVariableKind(

Variable.VAR\_KIND\_VIEW\_CRITERIA\_PARAM);

fstNameVar.setProperty(

AttributeHints.ATTRIBUTE\_DISPLAY\_HINT,

AttributeHints.ATTRIBUTE\_DISPLAY\_HINT\_HIDE);

//Add bind variable to VC Item as value

vci.setValue(0, ":dynamicBindVarAttribute");

vci.setIsBindVarValue(0, true);

//Initialize bind variable value

vvm.setVariableValue(fstNameVar, "A%");

//Insert View Criteria Row to VC

vc.insertRow(vcr);

return vc;

}

Remove applied View Criteria

public void unApplyVC(String viewCriteriaCNameToBeRemoved){

//Get Employee view object

ViewObjectImpl vo = (ViewObjectImpl)findViewObject("EmployeeDetails");

vo.removeApplyViewCriteriaName(viewCriteriaCNameToBeRemoved);

}

Persist the changes made on a View Criteria on Run-time

public void alterVCAndSaveState(){

//Get Employee view object

ViewObject vo = findViewObject("EmployeeDetails");

ViewCriteriaManager vcm = vo.getViewCriteriaManager();

//Get an existing VC

ViewCriteria vc = vcm.getViewCriteria

("EmployeeDetailsVC");

//Add new condition to VC

ViewCriteriaRow vcr = vc.createViewCriteriaRow();

vcr.setAttribute("EmployeeId","IN (1000,2000)");

vc.add(vcr);

//Persist the changes

vc.saveState();

}

## View Links

## Application Module

|  |
| --- |
| **Test Client**  Get the fully qualified module name from bc4j.xcfg in AM    **public** **static** **void** main(String[] args) {  String amDef = "com.nttdata.model.am.CoreHRMainAM"; // Fully qualified module name  String config = "CoreHRAMLocal"; // AM Name  ApplicationModule am = Configuration.createRootApplicationModule(amDef, config);  ViewObject vo = am.findViewObject("CountriesVO1");  vo.executeQuery();;  Row row = vo.first();  System.out.println(row.getAttribute(1));  // Work with your appmodule and view object here  Configuration.releaseRootApplicationModule(am, true);  } |

|  |
| --- |
| Runtime Behavior of ADF-BC  Read it from Real World |

## Polymorphic View Objects

## Validations

# ADF Model

# ADF UI