**EJB**

**What is the difference between normal Java object and EJB**

**Java Object**:it's a reusable componet

**EJB:**is reusable and deployable component which can be deployed in any container

EJB : is a distributed component used to develop business applications. Container provides runtime environment for EJBs.

EJB is an Java object implemented according EJB specification. Deployability is a feature.

**What is the difference between JavaBean and EJB**

**Java Beans :** is intra-process component,

JavaBeans is particularly well-suited for asynchronous, intra-application communications among software

**EJB** : is an Inter-Process component

# **What is EJB ?**

Enterprise Java Bean is a specification for server-side scalable,transactional and multi-user secure enterprise-level applications. It provides a consistant component architecture for creating distributed n-tier middleware.

Enterprise JavaBeans (EJB) is a technology that based on J2EE platform.

EJBs are server-side components. EJB are used to develop the distributed, transactional and secure applications based on Java technology.

**What is Session Bean. What are the various types of Session Bean**

**SessionBeans**: They are usually associated with one client. Each session bean is created and destroyed by the particular EJB client that is associated with it. These beans do not survive after system shutdown.

These Session Beans are of two types:

**Stateful Session Beans**:They maintain conversational state between subsequest calls by a client

b) Stateful Session Beans : These beans have internal states. They can be stored (getHandle()) and restored (getEJBObject()) across client sessions.Since they can be persistence, they are also called as Persistence Session Beans.

**Stateless Session Bean**:Consider this as a servlet equivalent in EJB. It is just used to service clients regardless of state and does not maintain any state.

a) Stateless Session Beans : These beans do not have internal States. They need not be passivated. They can be pooled into service multiple clients.

**What is the difference between Stateful session bean and Stateless session bean**

**Stateful:**

* Stateful s.Beans have the passivated and Active state which the Stateless bean does not have.
* Stateful beans are also Persistent session beans. They are designed to service business processes that span multiple method requests or transactions.
* Stateful session beans remembers the previous requests and reponses.
* Stateful session beans does not have pooling concept.
* Stateful Session Beans can retain their state on behave of an individual client.
* Stateful Session Beans can be passivated and reuses them for many clients.
* Stateful Session Bean has higher performance over stateless sessiob bean as they are pooled by the application server.

**Stateless:**

* Stateless Session Beans are designed to service business process that last only for a single method call or request.
* Stateless session beans do not remember the previous request and responses.
* Stattless session bean instances are pooled.
* Stateless Session Beans donot maintain states.
* Stateless Session Beans, client specific data has to be pushed to the bean for each method invocation which result in increase of network traffic.

**What is the life cycle of Stateful session bean**

Stateful Session Bean has three states. Does not exists, Method Ready and Passivated states.

Like Stateless beans, when the Stateful Session Bean hasnt been instantiated yet (so it is not an instance in memory) is it in the Does not exists state.

Once a container creates one or more instances of a Stateful Session Bean it sets them in a Method Ready state. In this state it can serve requests from its clients. Like Stateless Session Beans, a new instance is created (Class.newInstance()), the context is passed (setSessionContext()) and finally the bean is created with ejbCreate().

During the life of a Stateful Session Bean, there are periods of inactivity. In these periods, the container can set the bean to the Passivate state. This happens through the ejbPassivate() method. From the Passivate state the bean can be moved back to the Method Ready state, via ejbActivate() method, or can go directly to the Does Not Exists state with ejbRemove().

**What is the life cycle of Stateless session bean**

Stateless session bean has only two states: Does Not Exists and Method Ready Pool.

A bean has not yet instantiated (so it is not an instance in memory) when it is in the Does Not Exists state.

When the EJB container needs one or more beans, it creates and set them in the Method Ready Pool state. This happens through the creation of a new instance (Class.newInstance()), then it is set its context (setSessionContext()) and finally calls the ejbCreate() method.

The ejbRemove() method is called to move a bean from the Method Ready Pool back to Does Not Exists state.

**What are the call back methods in Session bean**

Session bean callback methods differ whether it is Stateless or stateful Session bean. Here they are.

## **Stateless Session Bean :-**

1. setSessionContext()

2. ejbCreate()

3. ejbRemove()

## **Stateful Session Bean :-**

1. setSessionContext()

2. ejbCreate()

3. ejbPassivate()

4. ejbActivate()

5. ejbRemove()

**When you will chose Stateful session bean and Stateless session bean**

**Stateful** session bean is used when we need to maintain the client state . Example of statefull session is Shoping cart site where we need to maintain the client state .

**stateless** session bean will not have a client state it will be in pool.

To maintain the state of the bean we prefer stateful session bean and example is to get mini statement in

ATM we need sessions to be maintained.

**What is Entity Bean. What are the various types of Entity Bean**

Entity bean represents the real data which is stored in the persistent storage like Database or file system. For example, There is a table in Database called Credit\_card. This table contains credit\_card\_no,first\_name, last\_name, ssn as colums and there are 100 rows in the table. Here each row is represented by one instance of the entity bean and it is found by an unique key (primary key) credit\_card\_no.

There are two types of entity beans.

1) Container Managed Persistence(CMP)

2) Bean Managed Presistence(BMP)

**What is the difference between CMP and BMP**

CMP means Container Managed Persistence. When we write CMP bean , we dont need to write any JDBC code to connect to Database. The container will take care of connection our enitty beans fields with database. The Container manages the persistence of the bean. Absolutely no database access code is written inside the bean class.

BMP means Bean Managed Persistence. When we write BMP bean, it is programmer responsibility to write JDBC code to connect to Database.

**What is the lifecycle of Entity Bean**

The following steps describe the life cycle of an entity bean instance

An entity bean instances life starts when the container creates the instance using newInstance and then initialises it using setEntityContext.

The instance enters the pool of available instances. Each entity bean has its own pool. While the instance is in the available pool, the instance is not associated with any particular entity object identity. Any of these pooled instances may be used to execute finder (ejbFind) or home (ejbHome) methods.

An instance transitions from the pooled state to the ready state when the container selects that instance to service a client call to an entity object. There are two possible transitions from the pooled to the ready state: through the creation of an entity (ejbCreate and ejbPostCreate) or through the activation of an entity (ejbActivate).

When an entity bean instance is in the ready state, the instance is associated with a specific entity object identity. While the instance is in the ready state, the container can synchronize the instance with its representation in the underlying data source whenever it determines the need to using ejbLoad and ejbStore methods. Business methods can also be invoked zero or more times on an instance. An ejbSelect method can be called by a business method, ejbLoad or ejbStore method.

The container can choose to passivate an entity bean instance within a transaction. To passivate an instance, the container first invokes the ejbStore method to allow the instance to prepare itself for the synchronization of the database state with the instance’s state, and then the container invokes the ejbPassivate method to return the instance to the pooled state.

There are three possible transitions from the ready to the pooled state: through the ejbPassivate method, through the ejbRemove method (when the entity is removed), and because of a transaction rollback for ejbCreate, ejbPostCreate,or ejbRemove.

The container can remove an instance in the pool by calling the unsetEntityContext() method on the instance.

**What are the call back methods in Entity bean**

**Entity Bean:**

1. setEntityContext()

2. ejbCreate()

3. ejbPostCreate()

4. ejbActivate()

5. ejbPassivate()

6. ejbRemove()

7. unsetEntityContext()

**When you will chose CMP and BMP**

## **BMP**

- Bean managed persistence

- Developer has to write persistence code for ejbLoad(),ejbStore() for entity beans

- Should follow this approach only when its bare necessity to write your own persistence logic.Usually container managed persistence is quite sufficient and less error prone.

**CMP**

- Container managed persistence

- Developer maps the bean fields with the database fields in the deployment descriptors.

- Developer need not provide persistence logic (JDBC) within the bean class.

- Containiner manages the bean field to DB field synchronization.

The point is only ENTITY beans can have theier pesristence mechanism as CMP or BMP. Session beans, which usually contains workflow or business logic should never have persistence code.Incase you choose to write persistence within your session bean, its usefull to note that the persistence is managed by the container BMP.Session beans cannot be CMP and its not possibel to provide field mapping for session bean.

BMPs are much harder to develop and maintain than CMPs.All other things are being equal,choose CMPs over BMPs for pure maintainability.

There are limitations in the types of the data sources that may be supported for CMPs by a container provide.Support for non JDBC type data sources,such as CICS,are not supported by the current CMP mapping and invocation schema.Therefore accessing these would require a BMP.

Complex queries might not be possible with the basic EJBQL for CMPs.So prefer BMPs for complex queries.

If relations between entity beans are established then CMPs may be necessary.CMR has ability to define manage relationships between entity beans.

Container will tyr to optimize the SQL code for the CMPs,so they may be scalable entity beans than the BMPs.

BMPs may be inappropriate for larger and more performance sesitive applications.

**What are advantages and disadvantages of CMP and BMP**

**CMP:** Container managed persistence

Advantages:

1)Easy to develop and maintain.

2)Relationships can be maintained between different entities.

3)Optimization of SQL code will be done.

4)Larger and more performance applications can be done.

Disadvantages:

1)Will not support for some nonJDBC data sources,i.e,CICS.

2)Complex queries cannot be developed with EJBQL.

**BMP::** Bean managed persistence

Advantages:

1)Support for nonJDBC data sources.

2)Complex queries can be build.

Disadvantages:

1)Hard to develop and maintain.

2)We cannot maintain the relationships between different entities.

3)Optimization of SQL code cannot be done by the container, because bean it self contains the code.

4)Not appropriate for larger and complex applications.

**What is difference between EJB 1.1 and EJB 2.0**

EJB 2.0 adds the local beans, which are accessible only from within the JVM where beans are running in.

In EJB 1.1, we had to implement remote client views for all these beans, even if we had no remote clients.

**What is Message Driven Bean?**

Message Driven Bean (MDB) is an enterprise bean which runs inside the EJB container and it acts as Listener for the JMS asynchronous message . It does not have Home and Remote interface as Session or Entity bean. It is called by container when container receives JMS asynchronous message. MDB has to implement MessageListener which has a method onMessage(Message msg). When the container calls the MDB it passes the message to onMesage () method and then MDB process that message.

**What is the life cycle of MDB?**

The lifetime of an MDB instance is controlled by the container. Only two states exist: Does not exist and Ready , as illustrated in the following figure:

The life of an MDB instance starts when the container invokes newInstance() on the MDB class to create a new instance. Next, the container calls setMessageDrivenContext () followed by ejbCreate () on the instance. The bean then enters the Ready state and is ready to consume messages.

When a message arrives for the bean, the container invokes the onMessage() method of one of the available instances, passing a Message object in argument. Message s can be consumed and processed concurrently by using multiple instances of the same type.   
 The container invokes ejbRemove() on the bean instance when it no longer needs the instance. The bean instance can perform clean up operations here.

**What is local interface. How values will be passed**

If Client and EJB classes are in the same machine ( Same JVM) then we can use Local linterface instead of Remote interface. Since Client and EJB are in same JVM, values are passed by referance.

**What is the difference between local interface and remote interface**

We can describe the following common rules for choosing whether to use remote client view or local client view:

When you will potentially use a distributed environment (if your enterprise bean should be independent of its deployment place), you should obviously choose remote client view.

Use remote client view when you need to be sure that parameters passed between your EJB and the client (and/or other enterprise beans) should be passed "by value" instead of "by reference." With pass-by-value, the bean will have its own copy of the data, completely separated from the copy of the data at the client. With local client view, you can do pass-by-reference, which means your bean, as well as the client, will work directly with one copy of the data. Any changes made by the bean will be seen by the client and vice versa. Pass-by-reference eliminates time/system expenses for copying data variables, which provides a performance advantage.

If you create an entity bean, you need to remember that it is usually used with a local client view. If your entity bean needs to provide access to a client outside of the existing JVM (i.e., a remote client), you typically use a session bean with a remote client view. This is the so-called Session Facade pattern, the goal of which is that the session bean provides the remote client access to the entity bean.

If you want to use container-managed relationship (CMR) in your enterprise bean, you must expose local interfaces, and thus use local client view. This is mentioned in the EJB specification.

Enterprise beans that are tightly coupled logically are good candidates for using local client view. In other words, if one enterprise bean is always associated with another, it is perfectly appropriate to co-locate them (i.e., deploy them both in one JVM) and organize them through a local interface.

**What is EJB Query Language**

EJB QL is somewat similar to SQL. But ejb ql is used to retrieve data from bean objects where as sql is used to retrieve data from tables.

**What is ACID?**

**ACID** is releated to transactions. It is an acronyam of Atomic, Consistent, Isolation and Durable. Transaction must following the above four properties to be a better one

**Atomic**: It means a transaction must execute all or nothing at all.

**Consistent**: Consistency is a transactional characteristic that must be enforced by both the transactional system and the application developer

**Isolation**: Transaation must be allowed to run itselft without the interference of the other process or transactions.

**Durable**: Durablity means that all the data changes that made by the transaction must be written in some type of physical storage before the transaction is successfully completed. This ensures that transactions are not lost even if the system crashes.

**What are the various isolation levels in a transaction and differences between them?**

There are three isolation levels in Transaction. They are

1. Dirty reads 2.Non repeatable reads 3. Phantom reads.

**Dirrty Reads:** If transaction A updates a record in database followed by the transaction B reading the record then the transaction A performs a rollback on its update operation, the result that transaction B had read is invalid as it has been rolled back by transaction A.

**NonRepeatable Reads** :If transaction A reads a record, followed by transaction B updating the same record, then transaction A reads the same record a second time, transaction A has read two different values for the same record.

**Phantom Reads :**If transaction A performs a query on the database with a particular search criteria (WHERE clause), followed by transaction B creating new records that satisfy the search criteria, followed by transaction A repeating its query, transaction A sees new, phantom records in the results of the second query.

**What are the various transaction attributes and differences between them**

There are six transaction attributes that are supported in EJB.

1. Required - T1---T1

0---T1

2. RequiresNew – T1---T2

0---T1

3. Mandatory - T1---T1

0---Error

4. Supports - T1---T1

0---0

5. NotSupported - T1---0

0---0

6. Never - T1---Error

0---0

**What is the difference between activation and passivation**

Activation and Passivation is applicable for only Stateful session bean and Entity bean.

When Bean instance is not used for a while by client then EJB Container removes it from memory and puts it in secondary storage (often disk) so that the memory can be reused. This is called Passivation.

When Client calls the bean instance again then Container takes the passivated bean from secondary storage and puts it in memory to serve the client request. This is called Activation.

**What is Instance Pooling?**

Pooling of instances.

in stateless session beans and Entity Beans server maintains a pool of instances. whenever server got a request from client, it takes one instance from the pool and serves the client request.

**What is the difference between HTTPSession and Stateful Session Bean**

From a logical point of view, a Servlet/JSP session is similar to an EJB session. Using a session, in fact, a client can connect to a server and maintain his state.

But, is important to understand, that the session is maintained in different ways and, in theory, for different scopes.

A session in a Servlet, is maintained by the Servlet Container through the HttpSession object, that is acquired through the request object. You cannot really instantiate a new HttpSession object, and it does not contains any business logic, but is more of a place where to store objects.

A session in EJB is maintained using the SessionBeans. You design beans that can contain business logic, and that can be used by the clients. You have two different session beans: Stateful and Stateless. The first one is somehow connected with a single client. It maintains the state for that client, can be used only by that client and when the client "dies" then the session bean is "lost".

A Stateless Session Bean does not maintain any state and there is no guarantee that the same client will use the same stateless bean, even for two calls one after the other. The lifecycle of a Stateless Session EJB is slightly different from the one of a Stateful Session EJB. Is EJB Containers responsability to take care of knowing exactly how to track each session and redirect the request from a client to the correct instance of a Session Bean. The way this is done is vendor dependant, and is part of the contract.

**What is the difference between find and select methods in EJB**

select method is not there in EJBs

A select method is similar to a finder method for Entity Beans, they both use EJB-QL to define the semantics of the method.

They differ in that an ejbSelect method(s) are not exposed to the client and the ejbSelect method(s) can return values that are defined as cmp-types or cmr-types.

**What are the optional clauses in EJB QL**

Three optional clauses are available in EJB Ql.

1. SELECT

2. FROM

3. WHERE

The EJB QL must always contain SELECT and FROM clauses. The WHERE clause is optional.

The FROM clause provides declarations for the identification variables based on abstract schema name, for navigating through the schema. The SELECT clause uses these identification variables to define the return type of the query, and the WHERE clause defines the conditional query.

**What is handle in EJB**

To get hold the session state of the Stateful Session bean.

A handle is an abstraction of a network reference to an EJB object. A handle is intended to be used as a "robust" persistent reference to an EJB object.

**What is the difference between JNDI context, Initial context, session context and ejb context**

**JNDI Context** Provides a mechanism to lookup resources on the network

**Initial Context** constructor provides the initial context.

**Session Context** has all the information a session bean would require from the container

**Entity Context** has all the information that an Entity bean would need from a container

**Ejb Context** contains the information that is common to both the session and entity bean.

**What is the difference between sessioncontext and entitycontext**

Session Context Contains information that a Session Bean would require from the container

Entity Context contains the information that an Entity Bean would require from a container

**What is the difference between EAR, JAR and WAR file**

In J2EE application modules are packaged as EAR, JAR and WAR based on their functionality

## **JAR: Java Archive File**

EJB modules which contains enterprise java beans class files and EJB deployment descriptor are packed as JAR files with .jar extenstion

## **WAR : Web Archive File**

Web modules which contains Servlet class files,JSP FIles,supporting files, GIF and HTML files are packaged as JAR file with .war( web achive) extension

## **EAR : Enterprise File**

All above files(.jar and .war) are packaged as JAR file with .ear ( enterprise archive) extension and deployed into Application Server.

**What is deployment descriptor**

Deployment Descriptor is a XML document with .xml extenion. It basically descripes the deployment settings of an application or module or the component. At runtime J2EE server reads the deployment descriptor and understands it and then acts upon the component or module based the information mentioned in descriptor.

For example EJB module has a deployment descriptor ejb-jar.xml where we mention whether it is session or entity or mesage driven bean and where is the home, remore and Bean classes are located and what type of transaction etc. In a simple word, without deployment descritor the Container ( EJB/Servlet/JSP container) will not know what to do with that module.

Deployment Descriptor is a file located in the WEB-INF directory that controls the behaviour of Servlets and JSP.

The file is called Web.xml and contains

xmlHeader

Web.xml DOCTYPE Sevlet name

Web-appelements ------ Servlet Class

Init-parm

**Servlet Configuration :**

<web-app>

<Servlet>

<Servlet-name>Admin</Servlet-name>

<Servlet-Class>com.ds.AdminServlet</Servlet-class>

</Servlet>

*<init-param>*

*<param-value> </param-value>*

*<param-name> admin.com</param-name>*

*</init-param>*

<Servlet-mapping>

<Servlet-name>Admin</Servlet-name>

<url-pattern>/Admin</url-pattern>

</Servlet-mapping>

</web-app>

**EJB Deployment descriptor :**

Ejb-jar.xml

META-INF

Weblogic-ejb-jar.xml

<ejb-jar>

<enterprise-bean>

</Session>

<ejb-name>Statefulfinacialcalcu</ejb-name>

<home>fincal.stateful.fincalc</home>

<remote> fincal.stateful.fincalc </remote>

<ejb-Class> fincal.stateful.fincalcEJB <ejb-Class>

<session-type> Stateful </session-type>

<transaction-type> Container </transaction-type>

</Session>

</enterprise-bean>

<assembly-descriptor>

<container-transaction>

<method>

<ejb-name> Statefulfinacialcalcu </ejb-name>

<method-name> \* </method-name>

</method>

<transaction-attribute> supports </transaction-attribute>

</container-transaction>

<assembly-descriptor>

<ejb-jar>

**weblogic-ejb-jar.xml**

<weblogic-ejb-jar>

<weblogic-enterprise-bean>

<ejb-name> Statefulfinacialcalcu </ejb-name>

<jndi-name> statefulfinacalc </jndi-name>

</weblogic-enterprise-bean>

</weblogic-ejb-jar>

**What is CMR**

CMR - Container Managed Relationships allows the developer to declare various types of relationships between the entity beans

**What is the difference between CMP 1.1 and CMP 2.0**

CMR and sub classing of the CMP bean by the container

**What is the difference between optimistic locking and pessimistic locking**

*Optimistic locking* assumes that no one would read or change the data while changes are being by a bean

*Pessimistic locking* would rather lock down the data so that no one can access it

**What is lazy loading**

Lazy loading is a characteristic of an application when the actual loading and instantiation of a class is delayed until the point just before the instance is actually used. The goal is to only dedicate memory resources when necessary by only loading and instantiating an object at the point when it is absolutely needed.

Tools such as Eclipse have popularized the lazy-loading approach as they use the facility to control the load and initialization of heavyweight plug-ins. This gives the double bonus of speeding up the initial load time for the application, as not all plug-ins are loaded straightaway; ensuring efficiency as only the plug-ins that are used are loaded at all.

**Is Decorator an EJB design pattern**

No , If i throw a custom ApplicationException from a business method in Entity bean which is participating in a transaction, would the transaction be rolled back by container. Does container rolls back transaction only in case of SystemExceptions

Yes, the rollback will occur For declarative transactions, container will rollback on systemException. Container has no way to know whether a speicific application exception is serious enough to roll back the participated transaction. Use setRollbackOnly() to doom the transaction.

**What are simple rules that a Primary key class has to follow**

Implement the equals and hashcode methods.

**What is abstract schema**

CMP uses abstract schema to map to the physical database

**What is re-entrant. Is session beans reentrant. Is entity beans reentrant**

Re-entrant means where Bean A calls methods of Bean B and then Bean B turns around and calls methods of Bean A. The above all within a single thread of control. This is also called as loopback.

Entity beans are the only one bean that is reentrant. Neither Session bean nor Message Driven Bean are reentrant. When Entity bean, we have to declare in the deployment descriptor whether it is reentrant ( true or false).

Why an onMessage call in Message-driven bean is always a seperate transaction

The MDB is stateless and inherently each message is unique with respect to the MDB.

Each message needs to be processed independently. Hence the need for separate transactions

**Does Stateful Session bean support instance pooling**

All Beans support Instance Pooling

**statefull** session bean does not maintain instance pooling,

**stateless** session beans and **entity beans** can maintain instance pooling

**Why does EJB needs two interfaces(Home and Remote Interface)**

Home is to provide Lookup from JNDI while Remote is to provide Object Instantiated

**Can I invoke Runtime.gc() in an EJB**

No

**Can a Session Bean be defined without ejbCreate() method**

No

**Why are ejbActivate() and ejb Passivate() included for stateless session bean even though they are never required as it is nonconversational bean**

To have a consistent interface, so that there is no different interface that you need to implement for Stateful Session Bean and Stateless Session Bean. Both Stateless and Stateful Session Bean implement javax.ejb.SessionBean and this would not be possible if stateless session bean is to remove ejbActivate and ejbPassivate from the interface. You could argue that the two (stateful and stateless) are so different that they should have their own interface but Sun did not think so. They made both session beans implement the same interface and provided deployment descriptor to denote which one is it that you are deploying.

With EJB 1.1 specs, why is unsetSessionContext() not provided in Session Beans, like unsetEntityContext() in Entity Beans

This was the answer Provided by some one... According to Mohan this one is not correct. Please see Mohan's reply below and more in the comments section.

ejbRemove() is called for session beans every time the container destroyes the bean. So you can use this method to do the stuff you typically would do in unsetEntityContext(). For entity beans ejbRemove() is only called if the user explicitly deletes the bean. I think that is the reason why the engineers at SUN invented the unsetEntityContext() for this kind of bean.

**What is the difference between ejbStore() and ejbLoad()**

When the EJB container needs to synchronize the instance variables of an entity bean with the corresponding values stored in a database, it invokes the ejbLoad and ejbStore methods. The ejbLoad method refreshes the instance variables from the database, and the ejbStore method writes the variables to the database. The client cannot call ejbLoad and ejbStore.

**What is the difference between ejbCreate() and ejbPostCreate()**

Session and Message Driven Bean will have only ejbCreate() method and no ejbPostCreate () method. Entity bean will have both ejbCreate() and ejbPostCreate() methods.

The ejbPostCreate method returns void, and it has the same input parameters as the ejbCreate method. If we want to set a relationship field to initialize the bean instance, we should do so in the ejbPostCreate method. we cannot set a relationship field in the ejbCreate method.

The ejbPostCreate() allows the bean to do any post-create processing before it begins serving client requests. For every ejbCreate() there must be a matching (matching arguments) ejbPostCreate() method.

**Is stateless Sessiob bean create() method contains any parameters**

No. This method must not contain any input parameters and cannot be overloaded as well.

**How can i retrieve from inside my Bean(Stateless session and Entity CMP) the user name which i am serving (the user name of user just logged in my web application)**

Inside an EJB you may retrieve the "Caller" name, that is the login id by invoking: sessionContext.getCallerIdentity().getName() where sessionContext is the instance of "SessionContext" (setSessionContext) passed to the Session Bean, or the instance of "EntityContext" (setEntityContext) passed to the Entity Bean.

**What is EJB architecture(components)**

EJB Architecture consists of :

a) EJB Server

b) EJB containers that run on these servers,

c) Home Objects, Remote EJB Objects and Enterprise Beans that run within these containers,

d) EJB Clients and

e) Auxillary systems like JNDI (Java Naming and Directory Interface), JTS(Java Transaction Service) and security services.

If my session bean with single method insert record into 2 entity beans, how can I know that the process is done in same transaction (the attributes for these beans are Required)

If your method in the session bean is already running under a transaction the calls to any other bean which have been deployed with trans-attribute 'Required' will be executed within the same transaction context.

So if your session bean is using container-managed transactions and your method is deployed with 'Required', 'RequiresNew' or 'Mandatory', you can safely assume that the calls to your entity beans are handled under same transaction. If you're not running in a transaction, a separate transaction will be set up for each call to your entity beans.

If your session bean is using bean-managed transactions, you can ensure that the calls are handled in the same transaction by :

javax.transaction.UserTransaction tran= null;

try{

tran=ctx.getUserTransaction();

tran.begin();

myBeanHome1.create(....);

myBeanHome2.create(...);

tran.commit();

}catch(...){}

You may want to check if you're already running in a transaction by calling tran.getStatus().

**Is there a way to get the original exception object from inside a nested or wrapped Exception (for example an EJBException or RemoteException)**

Absolutely yes, but the way to do that depends on the Exception, since there are no standards for that. Some examples: ·When you have an javax.ejb.EJBException, you can use the getCausedByException() that returns a java.lang.Exception. ·A java.rmi.RemoteException there is a public field called detail of type java.lang.Throwable ·With a java.sql.SQLException you need to use the method getNextException() to get the chained java.sql.SQLException. ·When you have an java.lang.reflect.InvocationtargetException, you can get the thrown target java.lang.Throwable using the getTargetException() method.

**Can undefined primary keys are possible with Entity beans?If so, what type is defined?**

Yes,undefined primary keys are possible with Entity Beans.The type is defined as java.lang.Object.

**When two entity beans are said to be identical?Which method is used to compare identical or not?**

Two Entity Beans are said to be Identical,if they have the same home inteface and their primary keys are the same.To test for this ,you must use the component inteface's isIdentical() method.

**Why CMP beans are abstract classes?**

We have to provide abstract data to object mapping that maps the fields in our bean to a batabase, and abstract methods methods that corelate these fields.

**Is instance pooling necessary for entity beans?**

One of the fundamental concepts of Entity Beans is that they are the pooled objects.Instance pooling is the service of the container that allows the container to reuse bean instances,as opposed to creating new ones every time a request for a bean is made.This is a perfomance optimizatio done by the container.

**What is the difference b/w sendRedirect() and <jsp: forward>?**

sendredirect will happen on clint side & request , rsponse will be newly created, for forward action it is server side action & request, response is passed & not modified or destroyed.

**How the abstract classes in CMP are converted into concrete classes?**

EJB2.0 allows developer to create only abstract classes and at the time of deployment the container creates concrete classes of the abstract. It is easy for container to read abstract classes and appropriately generate concrete classes.