JAVA means DURGA SOFT

JAVA FAQ`S

Very important Questions in EJB

DURGA M.Tech
(Sun certified & Realtime Expert)
Ex. IBM Employee
Trained Lakhs of Students
for last 14 years across INDIA

India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

EJB FAQs

1). What are the Differences between EJB 3.0 and EJB 2.1?

Differences are:
1) EJB 3.0 allows developers to program EJB components as ordinary Java objects with ordinary Java business interfaces rather than as heavy weight components like EJB 2 (home, remote).

2) In EJB 3.0 you can use annotation or deployment descriptors but in EJB 2 you have to use deployment descriptors.

3) EJB 3 introduced persistence API for database access. In EJB 2 you can use Entity bean.

4) EJB 3.0 is much faster the EJB2

2). What are the key features of the EJB technology?

EJB components are server-side components written entirely in the Java programming language
2. EJB components contain business logic only - no system-level programming & services, such as transactions, security, life-cycle, threading, persistence, etc. are automatically managed for the EJB component by the EJB server.
3. EJB architecture is inherently transactional, distributed, portable multi-tier, scalable and secure.
4. EJB components are fully portable across any EJB server and any OS.
5. EJB architecture is wire-protocol neutral–any protocol can be utilized like IIOP,JRMP, HTTP, DCOM,etc.

3). What is the difference between EJB and RMI

Both of them are java solution for distributed computing.
RMI offers remote access to an object running in another JVM and no other services.
But EJB offers far more services than RMI apart from remote method calling. EJB leverages this remote-object feature of RMI and ORB (RMI-IIOP) which can be called by any COBRA client, but also provides other services such as persistence, transaction management, security, resource management, object pooling and messaging.

4).What are the ways for a client application to get an EJB object?

The client has the JNDI name of the EJB object; this name is used to get the EJB object.

2) The client has the JNDI name of the Home object, this is a more usual case; this name is used to get the Home object, then a finder method is invoked on this Home to obtain one or several entity bean objects. The client may also invoke a "create" method on the Home object to create a new EJB object (session or entity).

3) The client has got a handle on an EJB object. A handle is an object that identifies an EJB object; it may be serialized, stored, and used later by a client to obtain a reference to the EJB Object, using the getEJBObject method().

4) The client has got a reference to an EJB object, and some methods defined on the remote interface of this Enterprise Bean return EJB objects.

5). What are the different kinds of enterprise beans?

Stateless session bean- An instance of these non-persistent EJBs provides a service without storing an interaction or conversation state between methods. Any instance can be used for any client.

Stateful session bean- An instance of these non-persistent EJBs maintains state across methods and transactions. Each instance is associated with a particular client.

Entity bean- An instance of these persistent EJBs represents an object view of the data, usually rows in a database. They have a primary key as a unique identifier. Entity bean persistence can be either container-managed or bean-managed.

Message-driven bean- An instance of these EJBs is integrated with the Java Message Service (JMS) to provide the ability for message-driven beans to act as a standard JMS message consumer and perform asynchronous processing between the server and the JMS message producer.

6). What is Entity Bean?

The entity bean is used to represent data in the database. It provides an object-oriented interface to data that would normally be accessed by the JDBC or some other back-end API. More than that, entity beans provide a component model that allows bean developers to focus their attention on the business logic of the bean, while the container takes care of managing persistence,transactions, and access control.

There are two basic kinds of entity beans: container-managed ersistence (CMP) andbean-managed persistence (BMP).

Container-managed persistence beans are the simplest for the bean developer to create and the most difficult for the EJB server to support. This is because all the logic for synchronizing the bean's state with the database is handled automatically by the container. This means that the bean developer doesn't need to write any data access logic, while the EJB server is supposed to take care of all the persistence needs automatically. With CMP, the container manages the persistence of the entity bean. Vendor tools are used to map the entity fields to the database and absolutely no database access code is written in the bean class.

The bean-managed persistence (BMP) enterprise bean manages synchronizing its state with the database as directed by the container. The bean uses a database API to read and write its fields to the database, but the container tells it when to do each synchronization operation and manages the transactions for the bean automatically. Bean-managed persistence gives the bean developer the flexibility to perform persistence operations that are too complicated for the container or to use a data source that is not supported by the container.

7). Why does EJB needs two interfaces(Home and Remote Interface)?

There is a pure division of roles between the two .
Home Interface is the way to communicate with the container which is responsible for creating , locating and removing beans and Remote Interface is the link to the bean that allows access to all methods and members

8).What is an EJB Context?

EJBContext is an interface that is implemented by the container, and it is also a part of the bean-container contract. Entity beans use a subclass of EJBContext called EntityContext. Session beans use a subclass called SessionContext. These EJBContext objects provide the bean class with information about its container, the client using the bean and the bean itself. They also provide other functions. See the API docs and the spec for more details

9). Does the container create a separate instance of the generated EJBHome and EJBObject classes?

The EJB container maintains an instance pool. The container uses these instances for the EJB Home reference irrespective of the client request. While referring the EJB Object classes the container creates a separate instance for each client request. The instance pool maintenance is up to the implementation of the container. If the container provides one, it is available otherwise it is not mandatory for the provider to implement it. Having said that, yes most of the container providers implement the pooling functionality to increase the performance of the application server. The way it is implemented is again up to the implementer.

10). What's difference between httpsession and EJB session bean ?

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 doesn't contains any business logic, but is more of a place where tostoreobjects.

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 doesn't 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 Container's responsibility 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.

11). What are the key benefits of the EJB technology?

1. Rapid application development
2. Broad industry adoption
3. Application portability
4. Protection of IT investment

12). Why do we have a remove method in both EJBHome and EJBObject?

With the EJBHome version of the remove, you are able to delete an entity bean without first instantiating it (you can provide a Primary Key object as a parameter to the remove method). The home version only works for entity beans. On the other hand, the Remote interface version works on an entity bean that you have already instantiated. In addition, the remote version also works on session beans (stateless and stateful) to inform the container of your loss of interest in this bean.

13). What are the services provided by container?

Container services are totally depends upon the "vendor implementation". But more or less most of the vendors suppots the basic services like,

LifeCycle Management - It is Automatic...

Resource Management-Creating and destroying the objects based the current load of requests for better usage of memory.

Session Management - it is used by Developer coded callback methods...

Transaction Management - it is used by configuring deployment descriptor (DD) ...

Security management - it is used by configuring deployment descriptor (DD) ...

The other services, if any will be in advanced versions, and depends on Vendor specific.

14). Is it possible to share an HttpSession between a JSP and EJB? What happens when I change a value in the HttpSession from inside an EJB?

You can pass the HttpSession as parameter to an EJB method, only if all objects in session are serializable. This has to be consider as ?passed-by-value?, that means that its read-only in the EJB. If anything is altered from inside the EJB, it won't be reflected back to the HttpSession of the Servlet Container.The ?pass-by-reference? can be used between EJBs Remote Interfaces, as they are remote references. While it IS possible to pass an HttpSession as a parameter to an EJB object, it is considered to be ?bad practice (1)? in terms of object oriented design. This is because you are creating an unnecessary coupling between back-end objects (ejbs) and front-end objects (HttpSession). Create a higher-level of abstraction for your ejb's api. Rather than passing the whole, fat, HttpSession (which carries with it a bunch of http semantics), create a class that acts as a value object (or structure) that holds all the data you need to pass back and forth between front-end/back-end

15). What is the difference between a Coarse Grained? Entity Bean and a Fine Grained? Entity Bean?

A ?fine grained? entity bean is pretty much directly mapped to one relational table, in third normal form. A ?coarse grained? entity bean is larger and more complex, either because its attributes include values or lists from other tables, or because it ?owns? one or more sets of dependent objects. Note that the coarse grained bean might be mapped to a single table or flat file, but that single table is going to be pretty ugly, with data copied from other tables, repeated field groups, columns that are dependent on non-key fields, etc. Fine grained entities are generally considered a liability in large systems because they will tend to increase the load on several of the EJB server?s subsystems (there will be more objects exported through the distribution layer, more objects participating in transactions, more skeletons in memory, more EJB Objects in memory, etc.)

16). Does Stateful Session bean support instance pooling?

Stateful Session Bean conceptually doesn't have instance pooling.
What is the difference between JavaBean and EJB?
A Java Bean is a software component written in the Java programming language that conforms to the JavaBeans component specification. The JavaBeans APIs became part of the "core" Java APIs as of the 1.1 release of the JDK.

The JavaBeans specification defines a Java-based software component model that adds a number of features to the Java programming language. Some of these features include:

* introspection
* customization
* events
* properties
* persistence

Enterprise JavaBeans (EJBs) are Java-based software components that are built to comply with Java's EJB specification and run inside of an EJB container supplied by a J2EE provider. An EJB container provides distributed application functionality such as transaction support,

17). What are the Interfaces need to create to implement Session Bean with Exmaple?

Session bean class (CartBean)
Home interface (CartHome)
Remote interface (Cart)

Session bean class (CartBean) :

```
public class CartBean implements SessionBean {

String customerName;
String customerId;
Vector contents;

public void ejbCreate(String person)
throws CreateException {

if (person == null) {
throw new CreateException("Null person not allowed.");
}
else {
customerName = person;
}

customerId = "0";
contents = new Vector();
}

public void ejbCreate(String person, String id)
throws CreateException {

if (person == null) {
throw new CreateException("Null person not allowed.");
}
else {
customerName = person;
}

IdVerifier idChecker = new IdVerifier();
if (idChecker.validate(id)) {
customerId = id;
}
else {
throw new CreateException("Invalid id: "+ id);
}

contents = new Vector();
}
```

```
public void addBook(String title) {
contents.addElement(title);
}

public void removeBook(String title) throws BookException {

boolean result = contents.removeElement(title);
if (result == false) {
throw new BookException(title + "not in cart.");
}
}

public Vector getContents() {
return contents;
}

public CartBean() {}
public void ejbRemove() {}
public void ejbActivate() {}
public void ejbPassivate() {}
public void setSessionContext(SessionContext sc) {}

}
```

Home Interface:

```
public interface CartHome extends EJBHome {
Cart create(String person) throws
RemoteException, CreateException;
Cart create(String person, String id) throws
RemoteException, CreateException;
}
```

The signatures of the ejbCreate and create methods are similar, but differ in important ways. The rules for defining the signatures of the create methods of a home interface follow.

The number and types of arguments in a create method must match those of its corresponding ejbCreate method.
The arguments and return type of the create method must be valid RMI types.
A create method returns the remote interface type of the enterprise bean. (But an ejbCreate method returns void.)
The throws clause of the create method must include the java.rmi.RemoteException and the javax.ejb.CreateException

Remote Interface :

```
public interface Cart extends EJBObject {

public void addBook(String title) throws RemoteException;
public void removeBook(String title) throws
BookException, RemoteException;
public Vector getContents() throws RemoteException;
}
```

The method definitions in a remote interface must follow these rules:

Each method in the remote interface must match a method implemented in the enterprise bean class.
The signatures of the methods in the remote interface must be identical to the signatures of the corresponding methods in the enterprise bean class.
The arguments and return values must be valid RMI types.
The throws clause must include the java.rmi.RemoteEx

18). How many EJB Objects are created for a Bean?

For a Session bean - one EJB object for one bean instance. For entity bean it depends, if 2 users are accessing one row at time then one EJB object is used for both the beans other wise for each bean one EJB object.

19). What are the parameters must follow for Session Bean ?

It implements the SessionBean interface.
The class is defined as public.
The class cannot be defined as abstract or final.
It implements one or more ejbCreate methods.
It implements the business methods.
It contains a public constructor with no parameters.
It must not define the finalize method.

20).When you will chose Stateful session bean and Stateless session bean?

Stateful session beans are used when there is converstional state and when there is a need of temporary storage

Stateless session bean are used when there is no conversational state and when session bean has to be used only for database access

21). What is the difference between Stateful session bean and Stateless session bean?

A stateful session beans can keep data between client accesses. wheras a stateless session bean cannot.
2) A stateful seesion bean contain the state of client after seesion is expired. whereas a stateless bwan cnanot.
3) A stateful session beans use the bean of pools for client application n after use them it return the bean in the pool. whereas a stateless session bean cannot.

22). What are the callbacks method in Session Bean ?

```
public void ejbCreate() {}
public void ejbRemove() {}
public void ejbActivate() {}
public void ejbPassivate() {}
public void setSessionContext(SessionContext sc) {}
```

23). How is Stateful Session bean maintain their states with client?

When a client refers to a Stateful Session object reference, all calls are directed to the same object on the EJB container. The container does not require client identity information or any cookie object to use the correct object.
This means that for a client to ensure that calls are directed to the same object on the container, all it has to do is to use same reference for every call.

For example the following holds for all stateful session beans:

```
StatefulHome sfh = ...//get home interface for stateful bean
Stateful bean1 = sfh.create();
Stateful bean2 = sfh.create();
if (bean1.isIdentical(bean1)){} //this is true!
if (bean1.isIdentical(bean2)){} //this is false!

//Note that the second test would evaluate to true for stateless beans
```

Thus, if you're calling a Stateful Session Bean from a servlet, your servlet need to keep the reference to the remote object in the HttpSession object between client calls for you to be able to direct calls to the same object on the container.
Likewise, if you're calling from an application, you only obtain the reference to the bean once and reuse the object throughout the application session.

24). What is the free pool?
The free pool is a data structure the EJB container uses to cache anonymous instances of a given bean type. The free pool improves performance by reusing objects and skipping container callbacks when it can.

25). Without home and remote interfaces cant we implement ejb?

Was just reading about EJB 3.0. I suppose with EJB 3.0, Home interface is absolutely gone and implementing Business Interface is not mandatory. All enterprise beans in EJB 3.0 are just POJO (Plain Old Java Object) with appropriate annotations.

26). When are stateless EJBs passivated?

Stateless ejbs are never passivated. Since stateless ejbs do not have state, there is no need to passivate them. They are put back into the free pool after each method call so they will be available to service other requests.

27). Is method overloading allowed in EJB?

Yes you can overload methods Should synchronization primitives be used on bean methods? - No. The EJB specification specifically states that the enterprise bean is not allowed to use thread primitives. The container is responsible for managing concurrent access to beans at runtime.

28). What is handle and why it is used in EJB?

The handle mechanism allows a client application to maintain a reference to an EJB object. A handle object may be obtained by calling the getHandle() method on the reference to an EJB object. The main interest is that the handle class implements java.io.serializable interface, which means that a handle may be serialized. This allows the client to store the handle, or to pass it to another process. The handle may then be deserialized and used to obtain the reference to the EJB object, by calling the getEJBObject() method.

Handles on session bean objects are valid until the session bean object exists, i.e. their life time is limited to that of the client. Handles on entity bean objects are valid during the complete life time of the entity bean object; this means that such handles may be used by different clients and stored for a long time; the EJB server holding the entity bean objects may be stopped and restarted, the handle will still be valid.

If we consider the entity bean object of the example above (a2), the way to obtain a handle on this object is the following (the handle class is defined in the javax.ejb package):

Handle h = a2.getHandle();The handle object may then be serialized and stored in a file:

```
ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("handlefile"));
out.writeObject(h);
out.close();
```

Then, a client can read the handle, and retrieve the referenced object:

```
ObjectInputStream in = new ObjectInputStream(new FileInputStream("handlefile"));
Handle h = (Handle) in.readObject();
Account a = (Account)PortableRemoteObject.narrow(h.getEJBObject(),
Account.class);
```

The EJB Specification allows the client to obtain a handle for the home interface. This allows the client to store a reference to an entity bean's home interface in stable storage. The client code must use the javax.rmi.PortableRemoteObject.narrow(...) method to convert the result of the getEJBHome() method invoked on a handle to the home interface type

29). Implement Local and Remote Interfaces in EJB?

Remote BeansThe EJB 1.1 specification defines all EJBs as remote objects. This means that every time you make a call to an EJB, you are making a remote call. This means that there is considerable overhead to each EJB call, and hence performance implications. To combat this, server vendors invented a way of circumventing the remote calls to some degree. Oracle's solution with OC4J was the pass-by-reference setting, which determined whether EJB objects were communicated by reference to the object, or whether the whole object had to be passed to the client.

An EJB has a remote interface and a home interface, with the exception of MessageDrivenBeans. The remote interface extends the interface javax.ejb.EJBObject and the home interface extends the interface javax.ejb.EJBHome. The EJB is accessible from any client, in any JVM, provided they have the proper authorization.

For example, the Home and Remote interfaces of an EJB called EMP may look like this.

Remote:

```
public interface Emp extends EJBObject
{
long getEmpno() throws RemoteException;
void setEmpno(long newDeptno) throws RemoteException;
String getEname() throws RemoteException;
void setEname(String newDname) throws RemoteException;
```

Home:

```
public interface DeptHome extends EJBHome
{
public Emp create() throws RemoteException, CreateException;
public Dept findByPrimaryKey(DeptPK primaryKey) throws RemoteException,
FinderException;
```

Note that both the home and the remote interface throw a RemoteException in all of their method definitions. The ejb-jar.xml deployment descriptor for these EJBs would look something like the snippets below:

```
<entity>
<ejb-name>Emp</ejb-name>
<home>ejb.cmplocal.EmpHome</home>
<remote>ejb.cmplocal.Emp</remote>
<ejb-class>ejb.cmplocal.EmpBean</ejb-class>
.
.
.
```

Local BeansThe EJB 2.0 specification standardize a means of making local connections to EJBs with Local Interfaces.

For an EJB to be classed as a local EJB, it must implement the local versions of the home and remote interfaces, javax.ejb.EJBLocalObject for the Home interface, and javax.ejb.EJBLocalHome. For a client to call the Local interface, they must be running in the same JVM as the JVM that the EJB exists in. This means that not only an EJB can call a local EJB , Servlets or JSPs can also call the EJB via it's local interface if they are packaged together as part of same application.

For example, the LocalHome and Local interfaces of an EJB called EMP may look like this.

Local:

```
public interface Emp extends EJBLocalObject
{
long getEmpno();
void setEmpno(long newEmpno);
String getEname();
void setEname(String newEname);
```

LocalHome:

```
public interface EmpHome extends EJBLocalHome
{
public Emp create() throws CreateException;
public Emp findByPrimaryKey(EmpPK primaryKey) throws FinderException;
```

The ejb-jar.xml deployment descriptor for these EJBs would look something like the snippets

below:

```
<entity>
<ejb-name>Emp</ejb-name>
<local-home>ejb.cmplocal.EmpHome</local-home>
<local>ejb.cmplocal.Emp</local>
<ejb-class>ejb.cmplocal.EmpBean</ejb-class>
<cmp-version>2.x</cmp-version>
<abstract-schema-name>Emp</abstract-schema-name>
.
.
.
```

Note that now the local interfaces no longer throw the RemoteException, showing that they are not remotely called methods. Also, the XML contains different elements. There is now a local-home and a local tag. Also we are declaring that this is an EJB 2.x bean, using the cmp-version tag.

Calling Local BeansCalling a local bean from Java code is very simple, and very similar to using a remote bean. The code to call a remote bean is shown below.

```
try
{
Context ctx = new InitialContext();
Object o = ctx.lookup("Emp");
EmpHome empHome = PortableRemoteObject.narrow(o, EmpHome.class)
return empHome.findByDeptno(getDeptno());
}
catch (RemoteException r)
{
System.err.println("Error loading Employees(Remote): " + r.getMessage()); return null;
}
catch (NamingException n)
{
System.err.println("Error loading Employees(Naming): " + n.getMessage());
return null;
}
catch (FinderException f)
{
System.err.println("Error loading Employees(Finder): " + f.getMessage());
return null;
}
```

The code for a local bean is similar, but we no longer have to worry about the PortableRemoteObject, as the bean is no longer remote.

```
try
{
```

```
Context ctx = new InitialContext();
Object o = ctx.lookup("java:comp/env/LocalEmp");
EmpHome empHome = (EmpHome)o;
return empHome.findByDeptno(getDeptno());
}
catch (NamingException n)
{
System.err.println("Error loading Employees(Naming): " + n.getMessage());
return null;
}
catch (FinderException f)
{
System.err.println("Error loading Employees(Finder): " + f.getMessage());
return null;
}
```

As you can see, the local bean has to lookup the EJB slightly differently, even though they are running in the same container. Also, there is no RemoteException thrown by the find or the create methods, so the exception does not have to be caught.
There is one more difference, and that is in the ejb-jar.xml deployment descriptor. For an EJB to look up a local EJB, it must point to the correct location using an <ejb-local-ref> tag. If this is not used, the container will not be able to find the bean. For each EJB that needs to use the local EJB, the XML below must be in the deployment descriptor.

```
<entity>
<ejb-name>Dept</ejb-name>
.
.
.
<ejb-local-ref>
<ejb-ref-name>LocalEmp</ejb-ref-name>
<ejb-ref-type>Entity</ejb-ref-type>
<local-home>ejb.cmplocal.EmpHome</local-home>
<local>ejb.cmplocal.Emp</local>
<ejb-link>Emp</ejb-link>
</ejb-local-ref>
</entity>
```

This example will allow the EJB Dept to call the local EJB Emp using the name LocalEmp. This is required because EJBs can have both local and remote interfaces, and to call the EJB Emp via it's remote interface the EJB Dept would look up the name Emp rather than the local reference Local Home.

30). How can I call one EJB from inside of another EJB?

In case of Remote :

EJBs can be clients of other EJBs. It just works. Really. Use JNDI to locate the Home Interface of the other bean, then acquire an instance reference.

For Example : Context ctx = new InitialContext();

//get Home interface of bean

//narrow -retype

EmpHome lhome = (EmpHome)

javax.rmi.PortableRemoteObject.narrow(ctx.lookup("java:comp/env/LocalEmp"), EmpHome .class);

//get remote interface

EmpIbean = lhome.create();

//now you can call bussiness method on remote interface like

Ibean.doSomething()

Incase of Local : but we no longer have to worry about the PortableRemoteObject, as the bean is no longer remote

Context ctx = new InitialContext();

Object o = ctx.lookup("java:comp/env/LocalEmp");

EmpHome empHome = (EmpHome)o;

31). What is the difference between Message Driven Beans and Stateless Session beans

In several ways, the dynamic creation and allocation of message-driven bean instances mimics the behavior of stateless session EJB instances, which exist only for the duration of a particular method call. However, message-driven beans are different from stateless session EJBs (and other types of EJBs) in several significant ways:

Message-driven beans process multiple JMS messages asynchronously, rather than processing a serialized sequence of method calls.

Message-driven beans have no home or remote interface, and therefore cannot be directly accessed by internal or external clients. Clients interact with message-driven beans only indirectly, by sending a message to a JMS Queue or Topic.

Note: Only the container directly interacts with a message-driven bean by creating bean instances and passing JMS messages to those instances as necessary.

The Container maintains the entire lifecycle of a message-driven bean; instances cannot be created or removed as a result of client requests or other API calls

32). Can you control when passivation occurs?

The developer, according to the specification, cannot directly control when passivation occurs. Although for Stateful Session Beans, the container cannot passivate an instance that is inside a transaction. So using transactions can be a a strategy to control passivation.

The ejbPassivate() method is called during passivation, so the developer has control over what to do during this exercise and can implement the require optimized logic.

Some EJB containers, such as BEA WebLogic, provide the ability to tune the container to minimize passivation calls.

Taken from the WebLogic 6.0 DTD -"The passivation-strategy can be either "default" or "transaction". With the default setting the container will attempt to keep a working set of beans in the cache. With the "transaction" setting, the container will passivate the bean after every transaction (or method call for a non-transactional invocation).

33). How to call any EJB from a servlet/JSP/Java Client?

```
Context ctx = new InitialContext();

//get Home interface of bean
//narrow -retype
BeanHome lhome = (BeanHome)
javax.rmi.PortableRemoteObject.narrow(ctx.lookup("cz.train.Bean"), BeanHome.class);

//get remote interface
Bean lbean = lhome.create();

//now you can call bussiness method on remote interface like
lbean.doSomething()
```

34). Can the primary key in the entity bean be a Java primitive type such as int?

The primary key can't be a primitive type--use the primitive wrapper classes, instead. For example, you can use java.lang.Integer as the primary key class, but not int (it has to be a class, not a primitive)

35). What are the methods of Entity Bean?

An entity bean consists of 4 groups of methods:

1. create methods: To create a new instance of a CMP entity bean, and therefore insert data into the database, the create() method on the bean's home interface must be invoked. They look like this: EntityBeanClass ejbCreateXXX(parameters), where EntityBeanClass is an Entity Bean you are trying to instantiate, ejbCreateXXX(parameters) methods are used for creating Entity Bean instances according to the parameters specified and to some programmer-defined conditions.

A bean's home interface may declare zero or more create() methods, each of which must have corresponding ejbCreate() and ejbPostCreate() methods in the bean class. These creation methods are linked at run time, so that when a create() method is invoked on the home interface, the container delegates the invocation to the corresponding ejbCreate() and ejbPostCreate() methods on the bean class.

2. finder methods: The methods in the home interface that begin with "find" are called the find methods. These are used to query the EJB server for specific entity beans, based on the name of the method and arguments passed. Unfortunately, there is no standard query language defined for find methods, so each vendor will implement the find method differently. In CMP entity beans, the find methods are not implemented with matching methods in the bean class; containers implement them when the bean is deployed in a vendor specific manner. The deployer will use vendor specific tools to tell the container how a particular find method should behave. Some vendors will use object-relational mapping tools to define the behavior of a find method while others will simply require the deployer to enter the appropriate SQL command.

There are two basic kinds of find methods: single-entity and multi-entity. Single-entity find methods return a remote reference to the one specific entity bean that matches the find request. If no entity beans are found, the method throws an ObjectNotFoundException . Every entity bean must define the single-entity find method with the method name findByPrimaryKey(), which takes the bean's primary key type as an argument.

The multi-entity find methods return a collection (Enumeration or Collection type) of entities that match the find request. If no entities are found, the multi-entity find returns an empty collection.

3. remove methods: These methods (you may have up to 2 remove methods, or don't have them at all) allow the client to physically remove Entity beans by specifying either Handle or a Primary Key for the Entity Bean.

4. home methods: These methods are designed and implemented by a developer, and EJB specification doesn't have any requirements for them except the need to throw a RemoteException is each home method.

36). What is the difference between Container-Managed Persistent (CMP) bean and Bean-Managed Persistent(BMP) ?

Container-managed persistence(CMP) beans are the simplest for the bean developer to create and the most difficult for the EJB server to support. This is because all the logic for synchronizing the bean's state with the database is handled automatically by the container. This means that the bean developer doesn't need to write any data access logic, while the EJB server is supposed to take care of all the persistence needs automatically. With CMP, the container manages the persistence of the entity bean. A CMP bean developer doesn't need to worry about JDBC code and transactions, because the Container performs database calls and transaction management instead of the programmer. Vendor tools are used to map the entity fields to the database and absolutely no database access code is written in the bean class. All table mapping is specified in the deployment descriptor. Otherwise, a BMP bean developer takes the load of linking an application and a database on his shoulders.

The bean-managed persistence (BMP) enterprise bean manages synchronizing its state with the database as directed by the container. The bean uses a database API to read and write its fields to the database, but the container tells it when to do each synchronization operation and manages the transactions for the bean automatically. Bean-managed persistence gives the bean developer the flexibility to perform persistence operations that are too complicated for the container or to use a data source that is not supported by the container.BMP beans are not 100% database-independent, because they may contain database-specific code, but CMP beans are unable to perform complicated DML (data manipulation language) statements. EJB 2.0 specification introduced some new ways of querying database (by using the EJB QL - query language).

37).Can Entity Beans have no create() methods?

Yes. In some cases the data is inserted NOT using Java application, so you may only need to retrieve the information, perform its processing, but not create your own information of this kind

38). What is bean managed transaction?
If a developer doesn't want a Container to manage transactions, it's possible to implement all database operations manually by writing the appropriate JDBC code. This often leads to productivity increase, but it makes an Entity Bean incompatible with some databases and it enlarges the amount of code to be written. All transaction management is explicitly performed by a developer.

39). What are transaction isolation levels in EJB?

Transaction_read_uncommitted- Allows a method to read uncommitted data from a DB(fast but not wise).

2. Transaction_read_committed- Guarantees that the data you are getting has been committed.

3. Transaction_repeatable_read - Guarantees that all reads of the database will be the same during the transaction (good for read and update operations).

4. Transaction_serializable- All the transactions for resource are performed serial.

40). What is the difference between ejbCreate() and ejbPostCreate

The purpose of ejbPostCreate() is to perform clean-up database operations after SQL INSERTs (which occur when ejbCreate() is called) when working with CMP entity beans. ejbCreate() is called before database INSERT operations. You need to use ejbPostCreate() to define operations, like set a flag, after INSERT completes successfully.

When working with BMP entity beans, this is not necessary. You have full control over the entire process, so you can place all the necessary logic surrounding your INSERT statement directly in the ejbCreate() method.

Even if you are creating BMP entity beans, the recommendation would still be to include an empty ejbPostCreate() method. Although some application servers will not enforce it, the spec indicates that this placeholder should be there.

41).What is the difference between sessioncontext and entitycontext?

Since Enterprise Beans live in a managed container, the container is free to call your EJB components methods at its leisure.

The container houses the information like current status of bean, security credentials of the user currently accessing the bean in one object is called EJB Context Object.

A context represents a way for beans to perform callbacks and modify their current status

SessionContext is EJB context for session bean

EntityContext is EJB context for entity bean

Message driven context is EJB context for message driven bean

42). What is the difference between ejb Store() and ejb Load()?

ejbStore() will be called before ejbPassivate() and is used to store the object to persistent database.
ejbLoad() will be called before ejbActivate() and is used to retrieve the object from persistence datastore.

43). What is the difference between EAR, JAR and WAR file?
J2EE defines three types of archives:

1. Java Archives (JAR)—A JAR file encapsulates one or more Java

classes, a manifest, and a descriptor. JAR files are the lowest

level of archive. JAR files are used in J2EE for packaging EJBs

and client-side Java Applications.

2. Web Archives (WAR)—WAR files are similar to JAR files, except

that they are specifically for web applications made from

Servlets, JSPs, and supporting classes.

3. Enterprise Archives (EAR)—An EAR file contains all of the

components that make up a particular J2EE application.

44).How to implement an entity bean which the Primary Key is an auto numeric?

The EJB 2 Spec (10.8.3 - Special case: Unknown primary key class) says that in cases where the PrimaryKeys are generated automatically by the underlying database, thebean provider must declare the findByPrimaryKey method to return java.lang.Object and specify the

Primary Key Class as java.lang.Object in the Deployment Descriptor.

When defining the Primary Key for the Enterprise Bean, the Deployer using the Container Provider's tools will typically add additional container-managed fields to the concrete subclass of the entity bean class.

In this case, the Container must generate the Primary Key value when the entity bean instance is created (and before ejbPost Create is invoked on the instance.)

45). Is Decorator an EJB design pattern?

No. Decorator design pattern, is the one which exhibits very low level runtime polymorphism, for the specific and single object (Instance of the class), but not for atleast for a class. It is the stuff to add specific functionality to a single & pointed object and leaves others like it unmodified. It is having close similarities like aspect stuff, but not with EJB stuff.

46). What is lazy loading?

Lazy loading means not creating an object until the first time it is accessed. Lazy loading typically looks like this:

```
public class Example {
private Vector data = null;

public Vector getData() {
if (this.data == null) {
this.data = new Vector();
// Load data into vector ...
}
return this.data;
}
}
```

This technique is most useful when you have large hierarchies of objects (such as a product catalog). You can lazy-load subordinate objects as you navigate down the hierarchy, and thereby only creat objects when you need them.

47). What is Message Driven Bean?

An MDB is essentially a message consumer that can listen to a message destination or a message endpoint and gets activated when a message arrives. By design, MDBs are anonymous in nature and hence cannot be directly invoked by a client. The only way to invoke an MDB is to send a message to the destination or endpoint to which it is listening. As MDBs are stateless in nature and are not related to any specific client, they can be pooled for concurrent processing of messages.

48). What is CMR?

CMR is an acronym for Container Managed Relation-ships.
CMR, represented by the cmr fields in the deployment descriptor, which represents the relationship exists between different entities (entity beans), which are in turn exhibiting the database to the real world. The relationships are one-one, one-many, & many-many.
All the relations/ referential integrities will be managed by container, then the definition's in the deployment descriptor's are called as Container Managed Relationships (CMR)..

49). Can a Session Bean be defined without ejbCreate() method?

The ejbCreate() methods is part of the bean's lifecycle, so, the compiler will not return an error because there is no ejbCreate() method.

However, the J2EE spec is explicit:

the home interface of a Stateless Session Bean must have a single create() method with no arguments,
while the session bean class must contain exactly one ejbCreate() method, also without arguments.
Stateful Session Beans can have arguments (more than one create method)

50). What are the optional clauses in EJB QL?

WHERE and ORDERBY clauses are optional in EJB QL where as SELECT and FROM are required clauses.

51). Can I use session beans and hibernate (instead of entity beans) for persitance?

Yes, we can. It's same as BMP.

52). If session has thrown ApplicaitonException would you use EJBContext. set RollBack Only method?

According to the EJB specification, when the Application Exception is thrown, the EJBContext.set RollBackOnly method is not called.

Typically, an enterprise bean marks a transaction for rollback to protect data integrity before throwing an application exception, because application exceptions do not automatically cause the Container to rollback the transaction.

For example, an Account Transfer bean which debits one account and credits another account could mark a transaction for rollback if it successfully performs the debit operation, but encounters a failure during the credit operation.

53). What is the difference between activation and passivation?

This would be the difference between Activation and Passivation:

While the bean is in the ready stage, the EJB container may decide to deactivate, or passivate, the bean by moving it from memory to secondary storage. (Typically, the EJB container uses a least-recently-used algorithm to select a bean for passivation.) The EJB container invokes the bean's ejbPassivate method immediately before passivating it. If a client invokes a business method on the bean while it is in the passive stage, the EJB container activates the bean, moving it back to the ready stage, and then calls the bean's ejbActivate method.

54). How do you check whether the session is active in Stateful session bean ?

In Stateful session bean session is not itself a separate entity. it is contained in the bean it self. So in order to check tht we need the check whether the Stateful session bean is present or not which is done by just invoking the home interface with the jndi

55). What is the difference between find and select methods in EJB?
A select method can return a persistent field (or a collection thereof) of a related entity bean. A finder method can return only a local or remote interface (or a collection of interfaces).

Because it is not exposed in any of the local or remote interfaces, a select method cannot be invoked by a client. It can be invoked only by the methods implemented within the entity bean class. A select method is usually invoked by either a business or a home method.

A select method is defined in the entity bean class. For bean-managed persistence, a finder method is defined in the entity bean class, but for container-managed persistence it is not.

56). 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.

57). 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.

58). What is the difference between normal Java object and EJB?
Java Object: is a reusable component.
EJB : is a distributed component used to develop business applications. Container provides runtime environment for EJBs.

59). What is abstract schema?
Abstract schema is part of an entity bean's deployment descriptor which defines the bean's persistent fields and their relationship. Abstract schema is specifed for entity beans with container managed persistence. We specify the name of the Abstract schema name in the deployment descriptor. The queries written in EJB QL for the finder methods references this name. The information provided in this Abstract Schema is used by the container for persistence management and relationship management.

60). What is clustering. What are the different algorithms used for clustering?

Clustering is the use of multiple computers and storage devices to create what seems to be a single system. Clustering is often used to increase a system's availability and for load balancing on highly-trafficked Web sites.

Clustering algorithms find groups of items that are similar. For example, clustering could be used by an insurance company to group customers according to income, age, types of policies purchased and prior claims experience. It divides a data set so that records with similar content are in the same group, and groups are as different as possible from each other. Since the categories are unspecified, this is sometimes referred to as unsupervised learning.

Main strategies of clustering:
1. Hierarchical clustering
2. K-clustering (partitioning)
3. Self Organizing Maps (SOM)
4. Hybrids (incremental)

61). Why did I get a Lock Timed Out Exception?

When you get a Lock Timed Out Exception while invoking a stateful session EJB, one of two things has occurred:

* You have <allow-concurrent-calls> set to true in your weblogic-ejb-jar.xml descriptor and your call timed out while waiting to be processed. The timeout used in this case is the value <trans-timeout-seconds> element of the weblogic-ejb-jar.xml descriptor or its default value of 30 seconds.
* You do not have <allow-concurrent-calls> set to true and you attempt to invoke a stateful session bean that is already busy processing another request. In this case, the second method call will not block and a LockTimedOutException will be thrown immediately.

62). 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 set Message Driven Context() followed by ejb Create() 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 on Message() 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 ejb Remove() on the bean instance when it no longer needs the instance. The bean instance can perform clean up operations here.

63). Can an entity bean be a listener for JMS messages?

No. Message driven beans should be used to consume JMS messages.

64). 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)
65). What is IIOP ?
It is Internet Inter Object Resource Broker Protocl
66). Why don't stateful session beans have a pool?

Stateful session beans get instantiated once for each seperate client request and it stores the client information in it, there is no threading concept in EJB hence if there will be an instance pool will exist then there is a possiblity of information leak between different session objects.

therefore there is no concept of instance pooling in stateful session bean.

67). Without using entity beans can we do database transactions?

Without using entity beans we can do database transactions through Springs .Spring can be used to configure declarative transaction management, remote access to your logic using RMI or web services, mailing facilities and various options in persisting your data to a database

68). What is the use of using session facade design pattern in EJB'S?

There are many uses, important one is to reduce network traffic I you are calling many EJB from your Servlet then this is not advised, because it has to make many network trips, so what you do you call a Stateless session bean and this in turn calls other EJB, since they are in same container there is less network calls other thing you can do now is you can convert them to LOCAL EJB which has not network calls. This increases your server bandwidthJ. Problem solver this is good for a highly available system.

69). What is the difference between session and entity beans? When should I use one or the other?

An entity bean represents persistent global data from the database; a session bean represents transient user-specific data that will die when the user disconnects (ends his session). Generally, the session beans implement business methods (e.g. Bank.transferFunds) that call entity beans (e.g. Account.deposit, Account.withdraw)

70). Is it possible to share an HttpSession between a JSP and EJB? What happens when I change a value in the HttpSession from inside an EJB? –

You can pass the HttpSession as parameter to an EJB method, only if all objects in session are serializable.This has to be consider as passed-by-value, that means that it's read-only in the EJB. If anything is altered from inside the EJB, it won't be reflected back to the HttpSession of the Servlet Container.The pass-by-reference can be used between EJBs Remote Interfaces, as they are remote references. While it is possible to pass an HttpSession as a parameter to an EJB object, it is considered to be bad practice in terms of object-oriented design. This is because you are creating an unnecessary coupling between back-end objects (EJBs) and front-end objects (HttpSession). Create a higher-level of abstraction for your EJBs API. Rather than passing the whole, fat, HttpSession (which carries with it a bunch of http semantics), create a class that acts as a value object (or structure) that holds all the data you need to pass back and forth between front-end/back-end. Consider the case where your EJB needs to support a non HTTP-based client. This higher level of abstraction will be flexible enough to support it.

71).What is EJB role in J2EE?

EJB technology is the core of J2EE. It enables developers to write reusable and portable server-side business logic for the J2EE platform.

72).what are Container-Managed Transactional attributes ?

Not Supported

The bean is not involved in a transaction. If the bean invoker calls the bean while involved in a transaction, the invoker's transaction is suspended, the bean executes, and when the bean returns, the invoker's transaction is resumed.

Required

The bean must be involved in a transaction. If the invoker is involved in a transaction, the bean uses the invoker's transaction. If the invoker is not involved in a transaction, the container starts a new transaction for the bean.

Supports

Whatever transactional state that the invoker is involved in is used for the bean. If the invoker has begun a transaction, the invoker's transaction context is used by the bean. If the

invoker is not involved in a transaction, neither is the bean.
RequiresNew

Whether or not the invoker is involved in a transaction, this bean starts a new transaction that exists only for itself. If the invoker calls while involved in a transaction, the invoker's transaction is suspended until the bean completes.

Mandatory

The invoker must be involved in a transaction before invoking this bean. The bean uses the invoker's transaction context.

Never

The bean is not involved in a transaction. Furthermore, the invoker cannot be involved in a transaction when calling the bean. If the invoker is involved in a transaction, a Remote Exception is thrown

73).How is persistence implemented in enterprise beans?
Persistence in EJB is taken care of in two ways, depending on how you implement your beans: container managed persistence (CMP) or bean managed persistence (BMP) For CMP, the EJB container which your beans run under takes care of the persistence of the fields you have declared to be persisted with the database - this declaration is in the deployment descriptor. So, anytime you modify a field in a CMP bean, as soon as the method you have executed is finished, the new data is persisted to the database by the container. For BMP, the EJB bean developer is responsible for defining the persistence routines in the proper places in the bean, for instance, the ejbCreate(), ejbStore(), ejbRemove() methods would be developed by the bean developer to make calls to the database. The container is responsible, in BMP, to call the appropriate method on the bean. So, if the bean is being looked up, when the create() method is called on the Home interface, then the container is responsible for calling the ejbCreate() method in the bean, which should have functionality inside for going to the database and looking up the data.

74). Are we allowed to change the transaction isolation property in middle of a transaction?

No. You cannot change the transaction isolation level in the middle of transaction.

75). For Entity Beans, What happens to an instance field not mapped to any persistent storage, when the bean is passivated?

The specification infers that the container never serializes an instance of an Entity bean (unlike stateful session beans). Thus passivation simply involves moving the bean from the ready to the pooled bin. So what happens to the contents of an instance variable is controlled by the programmer. Remember that when an entity bean is passivated the instance gets logically disassociated from it's remote object. Be careful here, as the functionality of passivation/activation for Stateless Session, Stateful Session and Entity beans is completely different. For entity beans the ejbPassivate method notifies the entity bean that it is being disassociated with a particular entity prior to reuse or for dereference.

76). What is a Message Driven Bean, what functions does a message driven bean have and how do they work in collaboration with JMS?

Message driven beans are the latest addition to the family of component bean types defined by the EJB specification. The original bean types include session beans, which contain business logic and maintain a state associated with client sessions, and entity beans, which map objects to persistent data. Message driven beans will provide asynchrony to EJB based applications by acting as JMS message consumers. A message bean is associated with a JMS topic or queue and receives JMS messages sent by EJB clients or other beans. Unlike entity beans and session beans, message beans do not have home or remote interfaces. Instead, message driven beans are instantiated by the container as required. Like stateless session beans, message beans maintain no client-specific state, allowing the container to optimally manage a pool of message-bean instances. Clients send JMS messages to message beans in exactly the same manner as they would send messages to any other JMS destination. This similarity is a fundamental design goal of the JMS capabilities of the new specification. To receive JMS messages, message driven beans implement the javax.jms.MessageListener interface, which defines a single onMessage() method. When a message arrives, the container ensures that a message bean corresponding to the message topic/queue exists (instantiating it if necessary), and calls its onMessage method passing the client's message as the single argument. The message bean's implementation of this method contains the business logic required to process the message. Note that session beans and entity beans are not allowed to function as message beans.

77).What is the advantage of putting an Entity Bean instance from the Ready State to Pooled state?

The idea of the Pooled State is to allow a container to maintain a pool of entity beans that has been created, but has not been yet synchronized or assigned to an EJBObject. This mean that the instances do represent entity beans, but they can be used only for serving Home methods (create or findBy), since those methods do not relay on the specific values of the bean. All these instances are, in fact, exactly the same, so, they do not have meaningful state. Jon Thorarinsson has also added: It can be looked at it this way: If no client is using an entity bean of a particular type there is no need for cachig it (the data is persisted in the database). Therefore, in such cases, the container will, after some time, move the entity bean from the Ready State to the Pooled state to save memory. Then, to save additional memory, the container may begin moving entity beans from the Pooled State to the Does Not Exist State, because even though the bean's cache has been cleared, the bean still takes up some memory just being in the Pooled State.

78). What is Session Bean?

The entity bean is used to represent data in the database. It provides an object-oriented interface to data that would normally be accessed by the JDBC or some other back-end API. More than that, entity beans provide a component model that allows bean developers to focus their attention on the business logic of the bean, while the container takes care of managing persistence, transactions, and access control.

There are two basic kinds of entity beans: container-managed persistance (CMP) and bean-managed persistence (BMP).

Container-managed persistence beans are the simplest for the bean developer to create and the most difficult for the EJB server to support. This is because all the logic for synchronizing the bean's state with the database is handled automatically by the container. This means that the bean developer doesn't need to write any data access logic, while the EJB server is supposed to take care of all the persistence needs automatically. With CMP, the container manages the persistence of the entity bean. Vendor tools are used to map the entity fields to the database and absolutely no database access code is written in the bean class.

The bean-managed persistence (BMP) enterprise bean manages synchronizing its state with the database as directed by the container. The bean uses a database API to read and write its fields to the database, but the container tells it when to do each synchronization operation and manages the transactions for the bean automatically. Bean-managed persistence gives the bean developer the flexibility to perform persistence operations that are too complicated for the container or to use a data source that is not supported by the container.

79).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)

It depends on the transaction attribute of the session bean also. You have to set the transaction attribute of the session bean either to Required or Requires New.

80).Can i map more than one table in a CMP?
No, you cannot map more than one table to a single CMP Entity Bean. CMP has been, in fact, designed to map a single table.

81). Difference between Session Bean remove() and Entity Bean remove() method?

SessionBean remove() : inform the container of your loss of interest in this bean. Container will remove the instance.

EntityBean remove() : delete an entity bean without first instantiating it. Delete the row of the table using mentioned primary key.

82). Does each stateless session bean have its own EJB Object?
This is container specific as it is responsible for handling the beans. There may be a 1:N or M:N relationship between EJB Object and the session Bean.