

## EJB Interview Questions and Answers

### [What is cold backup, hot backup, warm backup recovery?](#)

Answer :: Cold backup means all these files must be backed up at the same time, before the database is restarted. Hot backup (official name is 'online backup' ) is a backup taken of each tablespace while the database is running and is being accessed by the users .... [Read More](#)

### [What is software architecture of EJB?](#)

Answer :: Session and Entity EJBs consist of 4 and 5 parts respectively:

1. A remote interface (a client interacts with it),
2. A home interface (used for creating objects and for declaring business methods),
3. A bean object (an object, which actually performs business logic and EJB-specific operations).
4. A deployment descriptor (an XML file containing all information required for maintaining the EJB) or a set of deployment descriptors (if you are using some container-specific features).
5. A Primary Key class - is only Entity bean specific. .... [Read More](#)

### [What are the callback methods in Entity beans?](#)

Answer :: <span class="Apple-style-span" style="font-family: Verdana, Arial, Helvetica, sans-serif; font-size: small;">>The bean class defines create methods that match methods in the home interface and business methods that match methods in the remote interface. The bean class also implements a set of callback methods that allow the container to notify the bean of events in its life cycle. The callback methods are defined in the javax.ejb.EntityBean interface that is implemented by all entity beans. The EntityBean interface has the following definition. Notice that the bean class implements these methods.

```
<table width="100%" border="0">
```

```
<tbody>
```

```
<tr>
```

```
<td colspan="2"><font size="2" face="Verdana, Arial, Helvetica, sans-serif">public interface
```

```
javax.ejb.EntityBean {</font></td>
```

```
</tr>
```

```
<tr>
```

```
<td width="10%"> </td>
```

```
<td width="90%"><font size="2" face="Verdana, Arial, Helvetica, sans-serif">public void
```

```
setEntityContext();
```

```
public void unsetEntityContext();
```

```
public void ejbLoad();
```

```
public void ejbStore();
```

```

public void ejbActivate();
public void ejbPassivate();
public void ejbRemove();</font></td>
</tr>
<tr>
<td><font size="2" face="Verdana, Arial, Helvetica, sans-serif"></font></td>
<td> </td>
</tr>
</tbody>
</table>

```

The setEntityContext() method provides the bean with an interface to the container called the EntityContext. The EntityContext interface contains methods for obtaining information about the context under which the bean is operating at any particular moment. The EntityContext interface is used to access security information about the caller; to determine the status of the current transaction or to force a transaction rollback; or to get a reference to the bean itself, its home, or its primary key. The EntityContext is set only once in the life of an entity bean instance, so its reference should be put into one of the bean instance's fields if it will be needed later.

The unsetEntityContext() method is used at the end of the bean's life cycle before the instance is evicted from memory to dereference the EntityContext and perform any last-minute clean-up.

The ejbLoad() and ejbStore() methods in CMP entities are invoked when the entity bean's state is being synchronized with the database. The ejbLoad() is invoked just after the container has refreshed the bean container-managed fields with its state from the database. The ejbStore() method is invoked just before the container is about to write the bean container-managed fields to the database. These methods are used to modify data as it's being synchronized. This is common when the data stored in the database is different than the data used in the bean fields.

The ejbPassivate() and ejbActivate() methods are invoked on the bean by the container just before the bean is passivated and just after the bean is activated, respectively. Passivation in entity beans means that the bean instance is disassociated with its remote reference so that the container can evict it from memory or reuse it. It's a resource conservation measure the container employs to reduce the number of instances in memory. A bean might be passivated if it hasn't been used for a while or as a normal operation performed by the container to maximize reuse of resources. Some containers will evict beans from memory, while others will reuse instances for other more active remote references. The ejbPassivate() and ejbActivate() methods provide the bean with a notification as to when it's about to be passivated (disassociated with the remote reference) or activated (associated with a remote reference).</font>

.... [Read More](#)

[What is the difference between Container-Managed Persistent \(CMP\) bean and Bean-Managed Persistent\(BMP\) ?](#)

Answer :: 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. 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). ....

[Read More](#)

### [What are the methods of Entity Bean?](#)

Answer :: 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. ....

[Read More](#)

### [What is Entity Bean?](#)

Answer :: 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 persistence (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. .... [Read More](#)

### [What is Session Bean?](#)

Answer :: A session bean is a non-persistent object that implements some business logic running on the server. One way to think of a session object is as a logical extension of the client program that runs on the server.

Session beans are used to manage the interactions of entity and other session beans, access resources, and

generally perform tasks on behalf of the client.

There are two basic kinds of session bean: stateless and stateful.

Stateless session beans are made up of business methods that behave like procedures; they operate only on the arguments passed to them when they are invoked. Stateless beans are called stateless because they are transient; they do not maintain business state between method invocations. Each invocation of a stateless business method is independent from previous invocations. Because stateless session beans are stateless, they are easier for the EJB container to manage, so they tend to process requests faster and use less resources.

Stateful session beans encapsulate business logic and state specific to a client. Stateful beans are called "stateful" because they do maintain business state between method invocations, held in memory and not persistent. Unlike stateless session beans, clients do not share stateful beans. When a client creates a stateful bean, that bean instance is dedicated to service only that client. This makes it possible to maintain conversational state, which is business state that can be shared by methods in the same stateful bean. .... [Read More](#)

[What are the different kinds of enterprise beans?](#)

Answer :: 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. .... [Read More](#)