## **Java Transaction Management Interview Questions**

1. **Question 1. Describe Java Transaction API (JTA)?**

**Answer:**

* + Java Transaction API (JTA) specifies standard Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the resource manager, the application server, and the transactional applications.
  + The JTA allows applications to perform distributed transactions, that is, transactions that access and update data on two or more networked computer resources.
  + The Java Transaction API consists of three elements: a high-level application transaction demarcation interface, a high-level transaction manager interface intended for an application server, and a standard Java mapping of the X/Open XA protocol intended for a transactional resource manager.

1. **Question 2. Define Transaction Demarcation?**

**Answer:**

Transaction demarcation always wraps a sequence of actions, such as a single request, a single method, or a section of code within a method. The demarcation initializes some transactional behavior before the demarcated area begins, then ends that transactional behavior when the demarcated area ends. The application server uses these demarcations to determine the appropriate calls to the Transaction Manager object.

1. **Question 3. What Are The Different Types Of Transaction Demarcation?**

**Answer:**

**Declarative Demarcation:** When using declarative demarcation, you specify what transaction demarcation modes should be used around certain areas of code. Rather than implementing these demarcations directly in your code, you declare the demarcations in a configuration file or deployment descriptor. The application server is then responsible for making sure that the correct transactional behavior is used around the specified area.

**Programmatic Demarcation:** At times, you might need to demarcate transactions in your code. Generally, you should use programmatic demarcation as little as possible, as it is error-prone and can interfere with the application servers own transaction demarcation mechanisms. If you find it necessary to use programmatic demarcation, you must be very careful to ensure that your code handles any unexpected errors and conditions.

1. **Question 4. Explain Bean-managed Transaction?**

**Answer:**

* + When an enterprise bean defines the boundaries of the transaction it is referred as bean-managed transaction. In bean-managed transaction demarcation, the code in the session or message-driven bean explicitly marks the boundaries of the transaction.
  + Although beans with container-managed transactions require less coding, they have one limitation: When a method is executing, it can be associated with either a single transaction or no transaction at all. If we use bean managed transaction we are responsible for programming transaction logic into your application code so that developers are responsible for issuing a 'begin' statement and either a 'commit' or an 'abort' statement.
  + The benefit of bean managed transaction is that as a developer you have full control over transactional boundaries. For instance, you can run series of mini transactions within a bean's method using bean managed transaction.

1. **Question 5. Difference between a Transaction and Distributed Transaction?**

**Answer:**

A transaction defines a logical unit of work that either completely succeeds or produces no result at all.

A distributed transaction is simply a transaction that accessed and updated data on two or more networked resources, and therefore must be coordinated among those resources.

1. **Question 6. Name Few Components Involved in the Distributed Transaction Processing (DTP)?**

**Answer:**

* + The application,
  + Application server,
  + Transaction manager,
  + Resource adapter,
  + And the resource manager.

1. **Question 7. Explain Resource Manager In JTA Context?**

**Answer:**

The resource manager is nothing but a relational database management system (RDBMS), such as Oracle, SQL Server. The actual database management is handled by this component.

1. **Question 8. What Is Resource Adapter In JTA Context?**

**Answer:**

The resource adapter is the component that acts as a communications channel, or request translator, between the "outside world,” the application, and the resource manager. A JDBC driver is an example of resource adapter.

1. **Question 9. Explain The Transactional Attribute Required In JTA?**

**Answer:**

Required is the default transaction attribute that ensures the methods are invoked within Java Transaction API transaction context. Required makes the transactional context used by the bean. If not the new context will be created.

1. **Question 10. Explain The Transactional Attribute Required New In JTA?**

**Answer:**

Required New is used when the required results of the transactions to be committed irrespective of the caller's transactions.

1. **Question 11. Explain The Transactional Attribute Mandatory?**

**Answer:**

Use the Mandatory attribute if the enterprise beans method must use the transaction of the client.

If the client is running within a transaction and invokes the enterprise beans method, the method executes within the clients transaction. If the client is not associated with a transaction, the container throws the Transaction Required Exception.

1. **Question 12. What Are The Different Transactional Attributes?**

**Answer:**

* + Required,
  + Requires New,
  + Mandatory,
  + Not Supported,
  + Supports,
  + Never.

1. **Question 13. What Is Java Transaction Service (JTS)?**

**Answer:**

JTS is a specification for implementing a Java transaction manager. A transaction manager serves as an intermediary between an application and one or more transaction-capable resource managers such as database servers and messaging systems. The JTA specification encompasses the JTA API specification.

1. **Question 14. What Is X/open Xa Architecture?**

**Answer:**

In the X/Open XA architecture, a transaction manager or transaction processing monitor (TP monitor) coordinates the transactions across multiple resources such as databases and message queues. Each resource has its own resource manager. The resource manager typically has its own API for manipulating the resource, for example the JDBC API to work with relational databases. In addition, the resource manager allows a TP monitor to coordinate a distributed transaction between its own and other resource managers. Finally, there is the application which communicates with the TP monitor to begin, commit or rollback the transactions. The application also communicates with the individual resources using their own API to modify the resource.

1. **Question 15. What is @transactional Annotation?**

**Answer:**

The javax.transaction.Transactional annotation provides the application the ability to control transaction boundaries declaratively. This annotation can be applied to any class that the Java EE specification defines as a managed bean (which includes CDI managed beans).

1. **Question 16. Does The J2ee Platform Support Nested Transactions?**

**Answer:**

No, the J2EE platform supports only flat transactions.

1. **Question 17. Should I Put A Transactional Attribute On An Asynchronous Action Such As Sending An Email?**

**Answer:**

No. Simply putting a transactional attribute on a method won't help if the resource manager can't use a transactional context.

1. **Question 18. Can An Entity Bean Use Bean-managed Transaction Demarcation?**

**Answer:**

No. Entity beans always use container-managed transaction demarcation. Session beans can use either container-managed or bean-managed transaction demarcation, but not at the same time.

1. **Question 19. What Are Transactions In JavaEE Application?**

**Answer:**

Transactions in Java EE application are a series of actions that all must be completed successfully, or else the changes in each action are backed out. If all the actions are successful then the changes from all actions are committed. If any one of the actions is unsuccessful then the changes from all the actions are rolled back.

1. **Question 20. What Is JTA (java Transaction Api) In JavaEE?**

**Answer:**

Java Transaction API is an API specified in Java EE that provides applications a standard way to access transactions independent of any specific implementation. JTA transactions are controlled by Java EE transaction manager. Transaction are started, committed or rolled back by calling corresponding methods on the User Transaction API.

1. **Question 21. How Are Transactions Maintained In An EJB Application?**

**Answer:**

Transactions in EJB application are either Container-Managed or Bean-Managed.

**Container-Managed Transactions** - In Container-Managed transactions the transaction boundaries are set in the EJB container. Container-managed transactions can be set for both session beans as well as message-driven beans.

**Bean-Managed Transactions** - In Bean-Managed transactions the transaction boundaries are set specifically within the bean's code. Bean managed transactions can be set for session beans as well as message-driven beans. Bean manages transaction can either use JDBC transactions or JTA transactions.

1. **Question 22. What Are Transaction Attributes In Container-managed Transactions?**

**Answer:**

Transaction attribute determines the scope of a transaction across beans. For example, if method () of Bean A starts a transaction and calls method () of bean b, then does method () run within the transaction started by method (). This is determined by the transaction attribute set on method ()

**A transaction attribute can have one of the following values:**-

* + Required
  + Requires New
  + Mandatory
  + Not Supported
  + Supports
  + Never

1. **Question 23. What is 'required' Transaction Attribute That Can Be Defined In Container-managed Transactions?**

**Answer:**

If a client has an associated transaction and calls a container-managed bean's method then the method executes within the client transaction.

If the client is not associated with a transaction then the container starts a new transaction before calling the bean's method.

1. **Question 24. What is 'requires New' Transaction Attribute That Can Be Defined In Container-managed Transactions?**

**Answer:**

If a client has an associated transaction and calls a container-managed bean's method then the container suspends the client’s transaction, starts a new transaction, starts the call to the method and starts the client’s transaction after method completes.

If the client is not associated with a transaction then the container starts a new transaction before calling the beans method.

1. **Question 25. What is 'mandatory' Transaction Attribute That Can Be Defined In Container-managed Transactions?**

**Answer:**

If a client has an associated transaction and calls a container-managed bean's method then the method executes within the client transaction.

If a client is not associated with a transaction then the container throws a TransactionRequiredException.

1. **Question 26. What is 'not Supported' Transaction Attribute That Can Be Defined In Container-managed Transactions?**

**Answer:**

If a client has an associated transaction and calls a container-managed bean's method then the container suspends the client’s transaction, calls the bean's method, and starts the client’s transaction after method execution completes.

If a client is not associated with a transaction, then the container does not start a new transaction before calling the bean's method.

1. **Question 27. What is 'supports' Transaction Attribute That Can Be Defined In Container-managed Transactions?**

**Answer:**

If a client has an associated transaction and calls a container-managed bean method then the method executes within the client transaction.

If a client is not associated with a transaction, then the container does not start a new transaction before calling the bean's method.

1. **Question 28. What Is The Never Transaction Attribute That Can Be Defined In Container-managed Transactions?**

**Answer:**

If a client has an associated transaction and calls a container-managed bean's method then the container throws a Remote Exception.

If a client is not associated with a transaction, then the container does not start a new transaction before calling the bean's method.

1. **Question 29. How Do You Set Transaction Timeouts In Container-managed And Bean-managed Transactions?**

**Answer:**

**Container-Managed Transaction** - In container managed transactions the transaction timeout is set at the container's administration console.

**Bean-Managed Transactions** - In Bean-managed transactions, the transaction timeout is set by calling the setTransactionTimeout () method of the User Transaction interface.