

Java-Success.com

Industrial strength Java/JEE Career Companion to open more doors

[Home](#)
[Java FAQs](#)
[600+ Java Q&As](#)
[Career](#)
[Tutorials](#)
[Member](#)
[Why?](#)
[Can u Debug?](#)
[Java 8 ready?](#)
[Top X](#)
[Productivity Tools](#)
[Judging Experience?](#)

[Home](#) › [Interview](#) › [JEE Interview Q&A](#) › [JTA](#) › JTA interview Q&A

JTA interview Q&A

Posted on [August 27, 2014](#) by [Arulkumaran Kumaraswamipillai](#)

Tweet

Q1. What is a Transaction? What does setAutoCommit do?

A1. A transaction is a set of operations that should be completed as a unit. If one operation fails then all the other operations fail as well. For example if you transfer funds between two accounts there will be two operations in the set

1. Withdraw money from one account.
2. Deposit money into other account.

These two operations should be completed as a single unit. Otherwise your money will get lost if the withdrawal is successful and the deposit fails. There are four characteristics (ACID properties) for a Transaction.

Atomicity	Consistency	Isolation	Durability
All the individual operations should either complete or fail.	The design of the transaction should update the database correctly.	Prevents data being corrupted by concurrent access by two different sources. It keeps transactions isolated or separated from each other until they are finished.	Ensures that the database is definitely updated once the Transaction is completed.

ACID properties

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

[open all](#) | [close all](#)

- [Ice Breaker Interview](#)
- [Core Java Interview C](#)
- [JEE Interview Q&A \(3](#)
- [JEE Overview \(2\)](#)
- [Web basics \(8\)](#)
- [WebService \(11\)](#)
- [JPA \(2\)](#)
- [JTA \(1\)](#)
- [JTA interview Q&](#)
- [JDBC \(4\)](#)
- [JMS \(5\)](#)
- [JMX \(3\)](#)
- [JNDI and LDAP \(1\)](#)
- [Pressed for time? Jav](#)
- [SQL, XML, UML, JSC](#)
- [Hadoop & BigData Int](#)
- [Java Architecture Inte](#)
- [Scala Interview Q&As](#)
- [Spring, Hibernate, & I](#)
- [Testing & Profiling/Sa](#)
- [Other Interview Q&A 1](#)

Transactions maintain data integrity. A transaction has a beginning and an end like everything else in life. The `setAutoCommit(...)`, `commit()` and `rollback()` are used for marking the transactions (known as transaction demarcation). When a connection is created, it is in auto-commit mode. This means that each individual SQL statement is treated as a transaction and will be automatically committed immediately after it is executed. The way to allow two or more statements to be grouped into a transaction is to disable auto-commit mode:

```

1  try{
2      Connection myConnection = dataSource.getConnection();
3
4      // set autoCommit to false
5      myConnection.setAutoCommit(false);
6
7      withdrawMoneyFromFirstAccount(.....)
8      depositMoneyIntoSecondAccount(.....)
9
10     myConnection .commit();
11 }
12 catch(Exception sqle){
13     try{
14         myConnection .rollback();
15     }catch( Exception e){}
16 }
17 finally{
18     try{
19         if( conn != null) {
20             conn.close();
21         }
22     } catch( Exception e) {}
23 }
24

```

















The above code ensures that both operation 1 and operation 2 succeed or fail as an atomic unit and consequently leaves the database in a consistent state. Also, turning auto-commit off will provide better performance.

Q2. What is transaction demarcation? What are the different ways of defining transactional boundaries?

A2. Data Access Objects (DAO) are transactional objects. Each operation associated with CRUD operations like Create, Update and/or Delete operations should be associated with transactions. Transaction demarcation is the manner in which









16 Technical Key Areas

[open all](#) | [close all](#)

-  [Best Practice \(6\)](#)
-  [Coding \(26\)](#)
-  [Concurrency \(6\)](#)
-  [Design Concepts \(7\)](#)
-  [Design Patterns \(11\)](#)
-  [Exception Handling \(3\)](#)
-  [Java Debugging \(21\)](#)
-  [Judging Experience \(1\)](#)
-  [Low Latency \(7\)](#)
-  [Memory Management \(1\)](#)
-  [Performance \(13\)](#)
-  [QoS \(8\)](#)
-  [Scalability \(4\)](#)
-  [SDLC \(6\)](#)
-  [Security \(13\)](#)
-  [Transaction Management \(1\)](#)

80+ step by step Java Tutorials

[open all](#) | [close all](#)

-  [Setting up Tutorial \(6\)](#)
-  [Tutorial - Diagnosis \(2\)](#)
-  [Akka Tutorial \(9\)](#)
-  [Core Java Tutorials \(2\)](#)
-  [Hadoop & Spark Tutorial \(1\)](#)
-  [JEE Tutorials \(19\)](#)
-  [Scala Tutorials \(1\)](#)
-  [Spring & Hibernate Tutorials \(1\)](#)
-  [Tools Tutorials \(19\)](#)
-  [Other Tutorials \(45\)](#)

transaction boundaries are defined. There are two approaches for transaction demarcation.

There are 2 types of transaction demarcation

1. Declarative transaction demarcation
2. Programmatic transaction demarcation

1. Declarative transaction demarcation:

The programmer declaratively specifies the transaction boundaries using transaction attributes for an EJB via ejb-jar.xml deployment descriptor.

Note: Spring framework has support for declarative transaction demarcation by specifying transaction attributes via Spring config files. If you choose Spring framework to mark the transaction boundaries, then you need to turn off transaction demarcation in your EJB by marking it as `NotSupported`.

Q. How are these declarative transactions know when to rollback?

EJBs: When the EJB container manages the transaction, it is automatically rolled back when a **System Exception** occurs. This is possible because the container can intercept "System Exception". However, when an "Application Exception" occurs, the container does not intercept it and therefore leaves it to the code to roll back using `ctx.setRollbackOnly()` method.

Spring Framework: Provided `@Transactional` annotation in the service layer.

2. Programmatic transaction demarcation:

The programmer is responsible for coding transaction logic as shown above. The application controls the transaction via an API like JDBC API, JTA API, Hibernate API etc. JDBC transactions are controlled using the `java.sql.Connection`

100+ Java pre-interview coding tests

[open all](#) | [close all](#)

- [Can you write code? \(](#)
- [♦ Complete the given](#)
- [Converting from A to I](#)
- [Designing your classe](#)
- [Java Data Structures](#)
- [Passing the unit tests](#)
- [What is wrong with th](#)
- [Writing Code Home A](#)
- [Written Test Core Jav](#)
- [Written Test JEE \(1\)](#)

How good are your?

[open all](#) | [close all](#)

- [Career Making Know-](#)
- [Job Hunting & Resum](#)

object. There are two modes: auto-commit and manual commit. Following methods are provided in the JDBC API via non-XA `java.sql.Connection` class for programmatically controlling transactions:

```
1 public void setAutoCommit(boolean mode);
2 public boolean getAutoCommit();
3 public void commit();
4 public void rollback();
5
```

For XA-Connections use the following methods on `javax.transaction.UserTransaction`.

```
1 public void begin();
2 public void commit();
3 public void rollback();
4 public int getStatus();
5 public void setRollbackOnly();
6 public void setTransactionTimeout(int)
7
```

Q3. What support does Spring framework have for transaction management?

A3.

1. Spring supports both global transactions across multiple transactional resources through JTA and resource specific local transaction associated with a JDBC connection. It provides a consistent programming model to support both local and global transactions. You write your code once, and it can benefit from different transaction management strategies in different environments.

2. Supports both declarative transaction management via AOP and programmatic transaction management with the Spring Framework transaction abstraction which can run over any underlying transaction infrastructure. With the preferred declarative model, developers typically write little or no code related to transaction management.

Q4. What is a distributed (aka JTA/XA) transaction? How does it differ from a local transaction?

A4. There are two types of transactions:

Local transaction: Transaction is within the same database. As we have seen above, with JDBC transaction demarcation, you can combine multiple SQL statements into a single transaction, but the transactional scope is limited to a single database connection. A JDBC transaction cannot span multiple databases.

Distributed Transaction (aka Global Transaction, JTA/XA transaction): The transactions that constitute a distributed transaction might be in the same database, but more typically are in different databases and often in different locations. For example, A distributed transaction might consist of money being transferred from an account in one bank to an account in another bank. You would not want either transaction committed without assurance that both will complete successfully. The Java Transaction API (JTA) and its sibling Java Transaction Service (JTS), provide distributed transaction services for the JEE platform. A distributed transaction (aka JTA/XA transaction) involves a transaction manager and one or more resource managers. A resource manager represents any kind of data store. The transaction manager is responsible for coordinating communication between your application and all the resource managers. A transaction manager decides whether to commit or rollback at the end of the transaction in a distributed system. A resource manager is responsible for controlling of accessing the common resources in the distributed system.

Q5. What is two-phase commit?

A5. two-phase commit is an approach for committing a distributed transaction in 2 phases.

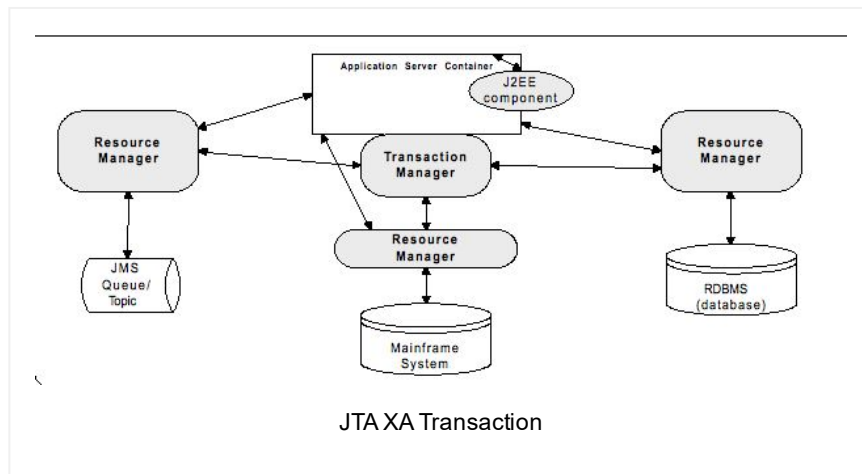
Q6. What do you understand by JTA and JTS?

A6. JTA is a high level transaction interface which allows transaction demarcation in a manner that is independent of the transaction manager implementation. JTS specifies the implementation of a Transaction Manager which supports the JTA. The code developed by developers does not call the

JTS methods directly, but only invokes the JTA methods. The JTA internally invokes the JTS routines.

Q7. What is a XA resource?

A7. The XA specification defines how an application program uses a transaction manager to coordinate distributed transactions across multiple resource managers. Any resource manager that adheres to XA specification can participate in a transaction coordinated by an XA-compliant transaction manager.



JTA transaction demarcation requires a JDBC driver that implements XA interfaces like `javax.sql.XADataSource`, `javax.sql.XAConnection` and `javax.sql.XAResource`. A driver that implements these interfaces will be able to participate in JTA transactions. You will also require to set up the `XADataSource` using your application server specific configuration files, but once you get a handle on the `DataSource` via JNDI lookup, you can get a XA connection via `javax.sql.DataSource.getConnection()` in a similar manner you get a non-XA connections. XA connections are different from non-XA connections and do not support JDBC's auto-commit feature. You cannot also use the `commit()`, `rollback()` methods on the `java.sql.Connection` class for the XA connections. A J2EE component can begin a transaction programmatically using `javax.transaction.UserTransaction` interface or it can also be started declaratively by the EJB container if an EJB bean uses container managed transaction. For explicit (i.e. programmatic) JTA/XA transaction you should use the `UserTransaction.begin()`,

UserTransaction.commit() and UserTransaction.rollback() methods. For example:

```
1 // programmatic JTA transaction
2 InitialContext ctx = new InitialContext();
3 UserTransaction utx = (UserTransaction)ctx.lookup("java:comp/env/jta/UT");
4
5 try {
6     //...
7     utx.begin();
8     //...
9     DataSource ds = getXDataSource();
10    Connection con = ds.getConnection(); // get
11    PreparedStatement pstmt = con.prepareStatement("insert into emp values(?,?)");
12    pstmt.setInt(1, 12456);
13    pstmt.executeUpdate();
14
15    utx.commit();//transaction manager uses two-phase commit
16 }
17 catch(SQLException sqle){
18     utx.rollback();
19     throw new RuntimeException(sqle);
20 }
21
```

Q8. Why JTA transactions are more powerful than JDBC transactions?

A8. JTA transactions are more powerful than JDBC transactions because a JDBC transaction is limited to a single database whereas a JTA transaction can have multiple participants like:

- JDBC connections.
- JMS queues/topics.
- Enterprise JavaBeans (EJBs).
- Resource adapters that comply with JEE Connector Architecture (JCA) specification.

Popular Posts

♦ [11 Spring boot interview questions & answers](#)

825 views

♦ [Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers](#)

766 views

[18 Java scenarios based interview Questions and Answers](#)

400 views

001A: ♦ 7+ Java integration styles & patterns

interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views

Bio

Latest Posts



Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via [Amazon.com](https://www.amazon.com) in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)



About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job



interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via [Amazon.com](https://www.amazon.com) in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)

◀ 01: ♥♦ 15+ Hibernate basics Q1 – Q7 interview questions & answers

8 JPA interview questions and answers ▶

Posted in JTA, member-paid, Transaction Management

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ [Java generics in no time](#) ☀ [Top 6 tips to transforming your thinking from OOP to FP](#) ☀ [How does a HashMap internally work? What is a hashing function?](#)
☀ [10+ Java String class interview Q&As](#) ☀ [Java auto un/boxing benefits & caveats](#) ☀ [Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect](#)

Non-Technical Know Hows

☀ [6 Aspects that can motivate you to fast-track your career & go places](#) ☀ [Are you reinventing yourself as a Java developer?](#) ☀ [8 tips to safeguard your Java career against offshoring](#) ☀ [My top 5 career mistakes](#)

Prepare to succeed

☀ [Turn readers of your Java CV go from “Blah blah” to “Wow”?](#) ☀ [How to prepare for Java job interviews?](#) ☀ [16 Technical Key Areas](#) ☀ [How to choose from multiple Java job offers?](#)

Select Category ▼

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.