10 JDBC Interview question answer in Java

Here is my list of frequently asked JDBC question in Java, I have tried to provide answer to most of question. If you have any interesting JDBC question which you have faced and not in this list then please share with us.

>

**Question 1:  What is JDBC?**

Answer: One of the first JDBC interview question in most of interviews. JDBC is java database connectivity as name implies it’s a java API for communicating to relational database, API has java classes and interfaces using that developer can easily interact with database. For this we need database specific JDBC drivers. Sometime this also result in follow-up questions like [Difference between type 2 and type 4 JDBC drivers](http://javarevisited.blogspot.sg/2012/05/different-types-of-jdbc-drivers-in-java.html). See the link for answer.

**Question 2: What are the main steps in java to make JDBC connectivity?**

Answer: Another beginner level JDBC Interview question, mostly asked on telephonic interviews. Here are main steps to connect to database.

          **Load the Driver**: First step is to load the database specific driver which communicates with database.

          **Make Connection**: Next step is get connection from the database using connection object, which is used to send SQL statement also and get result back from the database.

          **Get Statement object**: From connection object we can get statement object which is used to query the database

          **Execute the Query**: Using statement object we execute the SQL or database query and get result set from the query.

          **Close the connection**: After getting resultset and all required operation performed the last step should be closing the database connection.

For complete code example you can also referee [Java program to connect to Oracle database](http://javarevisited.blogspot.ca/2012/04/java-program-to-connect-oracle-database.html)

**Question 3: What is the mean of “dirty read“in database?**

Answer: This kind of JDBC interview question is asked on 2 to 4 years’ experience Java programmer, they are expected to familiar with [database transaction](http://javarevisited.blogspot.ca/2011/11/database-transaction-tutorial-example.html) and isolation level etc. As the name it convey the meaning of dirty read *“read the value which may or may not be correct”.* in database when one transaction is executing and changing some field value same time some another transaction comes and read the change field value before first transaction **commit or rollback** the value ,which cause invalid value for that field, this scenario is known as **dirty read**.

**Question 4: What is 2 phase commit?**

Answer: This is one of the most popular JDBC Interview question and asked at advanced level, mostly to senior Java developers on J2EE interviews. Two phase commit is used in distributed environment where multiple process take part in distributed transaction process. In simple word we can understand like if any transaction is executing and it will effect multiple database then two phase commit will be used to make all database synchronized with each other.

In two phase commit, commit or rollback is done by two phases:

1.       **Commit request phase**: in this phase main process or coordinator process take vote of all other process that they are complete their process successfully and ready to commit if all the votes are “**yes**” then they go ahead for next phase. And if “No “then rollback is performed.

**2.       Commit phase:** according to vote if all the votes are yes then commit is done.

Similarly when any transaction changes multiple database after execution of transaction it will issue pre commit  command on each database and all database send acknowledgement and according to acknowledgement if all are positive transaction will issue the commit command otherwise rollback is done .

**Question 5: What are different types of Statement?**

Answer:  This is another classical JDBC interview question. Variants are Difference between Statement, PreparedStatemetn and CallableStatement in Java. Statement object is used to send SQL query to database and get result from database, and we get statement object from connection object.

There are three types of statement:

**1. Statement:** it’s a commonly used for getting data from database useful when we are using static SQL statement at runtime. It will not accept any parameter.

              Statement   stmt = conn.createStatement( );

      ResultSet rs = stmt.executeQuery();

**2. PreparedStatement**: when we are using same SQL statement multiple time it is useful and it will accept parameter at runtime.

              String SQL = "Update stock SET limit =? WHERE stockType =?";

      PreparedStatement pstmt = conn.prepareStatement(SQL);

      ResultSet rs = pstmt.executeQuery();

To learn more about PreparedStatement, see  [What is PreparedStatement in Java and Benefits](http://javarevisited.blogspot.sg/2012/03/why-use-preparedstatement-in-java-jdbc.html)

**3. Callable Statement**: when we want to access stored procedures then callable statement are useful and they also accept runtime parameter**.**It is called like this

      CallableStatement cs = con.prepareCall("{call SHOW\_SUPPLIERS}");

      ResultSet rs = cs.executeQuery();

**Question 6: How cursor works in scrollable result set?**

Answer: Another tough JDBC Interview question, not many Java programmer knows about using Cursor in Java.

In JDBC 2.0 API new feature is added to move cursor in resultset backward forward and also in a particular row .

There are three constant define in result set by which we can move cursor.

          **TYPE\_FORWARD\_ONLY**: creates a nonscrollable result set, that is, one in which the cursor moves only forward

          **TYPE\_SCROLL\_INSENSITIVE** : a scrollable result set does not reflects changes that are made to it while it is open

          **TYPE\_SCROLL\_SENSITIVE**: a scrollable result set reflects changes that are made to it while it is open

**Question 7:  What is connection pooling?**

Answer: This is also one of the most popular question asked during JDBC Interviews. Connection pooling is the mechanism by which we reuse the recourse like connection objects  which are  needed to make connection with database .In this mechanism client are not required every time make new connection and then interact with database instead of that connection objects are stored in connection pool and client will get it from there. So it’s a best way to share a server resources among the client and enhance the application performance. If you use spring framework, then you can also refer [How to setup JDBC Connection Pool using spring in Java](http://javarevisited.blogspot.sg/2012/06/jdbc-database-connection-pool-in-spring.html)

**Question 8: What do you mean by cold backup, hot backup?**

Answer: This question is not directly related to JDBC but some time asked during JDBC interviews. Cold back is the backup techniques in which backup of files are taken before the database restarted. In hot backup backup of files and table is taken at the same time when database is running. A warm is a recovery technique where all the tables are locked and users cannot access at the time of backing up data.

**Question 9: What are the locking system in JDBC?**

Answer: One more tough JDBC question to understand and prepare. There are 2 types of locking in JDBC by which we can handle multiple user issue using the record. if two user are reading the same record then there is no issue but what if users are updating the record , in this case changes done by first user is gone by second user if he also update the same record .so we need some type of locking so no lost update.

**Optimistic Locking:**optimistic locking lock the record only when update take place. Optimistic locking does not use exclusive locks when reading

**Pessimistic locking:** in this record are locked as it selects the row to update

**Question 10: Does the JDBC-ODBC Bridge support multiple concurrent open statements per connection?**

Answer: No, we can open only one statement object when using JDBC-ODBC Bridge.

==========================================================================

**1. What is the JDBC?**

Java Database Connectivity (**JDBC**) is a standard Java API to interact with relational databases form Java. **JDBC** has set of classes and interfaces which can use from Java application and talk to database without learning RDBMS details and using Database Specific JDBC Drivers.

**2. What are the new features added to JDBC 4.0?**

The major features added in JDBC 4.0 include:

* Auto-loading of JDBC driver class
* Connection management enhancements
* Support for RowId SQL type
* DataSet implementation of SQL using Annotations
* SQL exception handling enhancements
* SQL XML support

**3. Explain Basic Steps in writing a Java program using JDBC?**

JDBC makes the interaction with RDBMS simple and intuitive. When a Java application needs to access database:

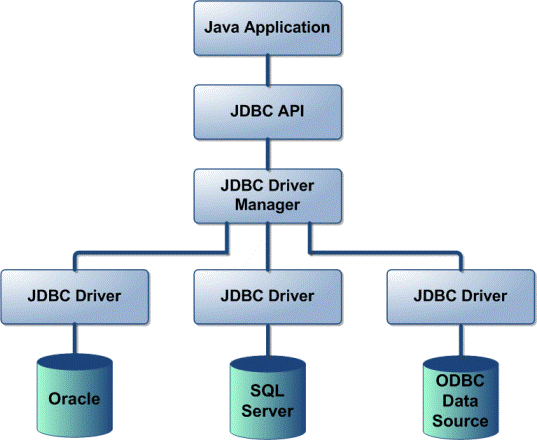
* Load the RDBMS specific JDBC driver because this driver actually communicates with the database (In case of JDBC 4.0 this is automatically loaded).
* Open the connection to database which is then used to send SQL statements and get results back.
* Create JDBC Statement object. This object contains SQL query.
* Execute statement which returns resultset(s). ResultSet contains the tuples of database table as a result of SQL query.
* Process the result set.
* Close the connection.

**4. Explain the JDBC Architecture.**

The JDBC Architecture consists of two layers:

* The JDBC API, which provides the **application-to-JDBC Manager** connection.
* The JDBC Driver API, which supports the **JDBC Manager-to-Driver** Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases. The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases. The location of the driver manager with respect to the JDBC drivers and the Java application is shown in Figure 1.



**Figure 1: JDBC Architecture**

**5. What are the main components of JDBC?**

The life cycle of a servlet consists of the following phases:

* **DriverManager**: Manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain sub protocol under JDBC will be used to establish a database Connection.
* **Driver**: The database communications link, handling all communication with the database. Normally, once the driver is loaded, the developer need not call it explicitly.
* **Connection:** Interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
* **Statement**: Encapsulates an SQL statement which is passed to the database to be parsed, compiled, planned and executed.
* **ResultSet**: The ResultSet represents set of rows retrieved due to query execution.

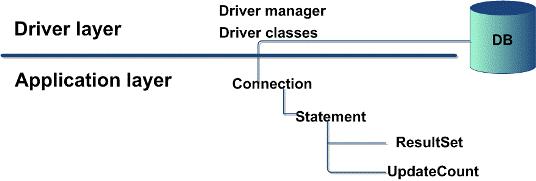
**6. How the JDBC application works?**

A JDBC application can be logically divided into two layers:

1. **Driver layer**

2. **Application layer**

* Driver layer consists of DriverManager class and the available JDBC drivers.
* The application begins with requesting the DriverManager for the connection.
* An appropriate driver is chosen and is used for establishing the connection. This connection is given to the application which falls under the application layer.
* The application uses this connection to create Statement kind of objects, through which SQL commands are sent to backend and obtain the results.



**Figure 2: JDBC Application**

|  |
| --- |
|  |

**7. How do I load a database driver with JDBC 4.0 / Java 6?**

Provided the JAR file containing the driver is properly configured, just place the JAR file in the classpath. Java developers **NO** longer need to explicitly load JDBC drivers using code like Class.forName() to register a JDBC driver. The DriverManager class takes care of this by automatically locating a suitable driver when the DriverManager.getConnection() method is called. This feature is backward-compatible, so no changes are needed to the existing JDBC code.

**8. What is JDBC Driver interface?**

The JDBC Driver interface provides vendor-specific implementations of the abstract classes provided by the JDBC API. Each vendor driver must provide implementations of the java.sql.Connection, Statement, PreparedStatement, CallableStatement, ResultSet and Driver.

**9. What does the connection object represents?**

The connection object represents communication context, i.e., all communication with database is through connection object only.

**10. What is Statement?**

Statement acts like a vehicle through which SQL commands can be sent. Through the connection object we create statement kind of objects.  
Through the connection object we create statement kind of objects.

Statement stmt = conn.createStatement();

This method returns object which implements statement interface.

**11. What is PreparedStatement?**

A prepared statement is an SQL statement that is precompiled by the database. Through precompilation, prepared statements improve the performance of SQL commands that are executed multiple times (given that the database supports prepared statements). Once compiled, prepared statements can be customized prior to each execution by altering predefined SQL parameters.

PreparedStatement pstmt = conn.prepareStatement("UPDATE EMPLOYEES SET SALARY =? WHERE ID =?");

pstmt.setBigDecimal(1, 153833.00);

pstmt.setInt(2, 110592);

Here: ***conn*** is an instance of the Connection class and *"****?****"* represents parameters. These parameters must be specified before execution.

**12. What is the difference between a Statement and a PreparedStatement?**

|  |  |
| --- | --- |
| **Statement** | **PreparedStatement** |
| A standard Statement is used to create a Java representation of a literal SQL statement and execute it on the database. | A PreparedStatement is a precompiled statement. This means that when the PreparedStatement is executed, the RDBMS can just run the PreparedStatement SQL statement without having to compile it first. |
| Statement has to verify its metadata against the database every time. | While a prepared statement has to verify its metadata against the database only once. |
| If you want to execute the SQL statement once go for STATEMENT | If you want to execute a single SQL statement multiple number of times, then go for PREPAREDSTATEMENT. PreparedStatement objects can be reused with passing different values to the queries |

**13. What are callable statements?**

Callable statements are used from JDBC application to invoke stored procedures and functions.

**14. How to call a stored procedure from JDBC?**

PL/SQL stored procedures are called from within JDBC programs by means of the prepareCall() method of the Connection object created. A call to this method takes variable bind parameters as input parameters as well as output variables and creates an object instance of the CallableStatement class.

The following line of code illustrates this:

CallableStatement stproc\_stmt = conn.prepareCall("{call procname(?,?,?)}");

Here conn is an instance of the Connection class.

**15. What are types of JDBC drivers?**

There are four types of drivers defined by JDBC as follows:

* **Type 1: JDBC/ODBC**—these require an ODBC (Open Database Connectivity) driver for the database to be installed. This type of driver works by translating the submitted queries into equivalent ODBC queries and forwards them via native API calls directly to the ODBC driver. It provides no host redirection capability.
* **Type2: Native API (partly-Java driver)**—this type of driver uses a vendor-specific driver or database API to interact with the database. An example of such an API is Oracle OCI (Oracle Call Interface). It also provides no host redirection.
* **Type 3: Open Protocol-Net**—this is not vendor specific and works by forwarding database requests to a remote database source using a net server component. How the net server component accesses the database is transparent to the client. The client driver communicates with the net server using a database-independent protocol and the net server translates this protocol into database calls. This type of driver can access any database.
* **Type 4: Proprietary Protocol-Net(pure Java driver)**—This has a same configuration as a type 3 driver but uses a wire protocol specific to a particular vendor and hence can access only that vendor's database. Again this is all transparent to the client.

**Note:** Type 4 JDBC driver is most preferred kind of approach in JDBC.

**16. Which type of JDBC driver is the fastest one?**

JDBC Net pure Java driver (Type IV) is the fastest driver because it converts the JDBC calls into vendor specific protocol calls and it directly interacts with the database.

**17. Does the JDBC-ODBC Bridge support multiple concurrent open statements per connection?**

No. You can open only one Statement object per connection when you are using the JDBC-ODBC Bridge.

**18. Which is the right type of driver to use and when?**

* Type I driver is handy for prototyping
* Type III driver adds security, caching, and connection control
* Type III and Type IV drivers need no pre-installation

Note: *Preferred by 9 out of 10 Java developers: Type IV*. [**Click here**](http://developersbook.com/jdbc/jdbc-drivers.php) to learn more about JDBC drivers.

**19. What are the standard isolation levels defined by JDBC?**

The values are defined in the class java.sql.Connection and are:

* TRANSACTION\_NONE
* TRANSACTION\_READ\_COMMITTED
* TRANSACTION\_READ\_UNCOMMITTED
* TRANSACTION\_REPEATABLE\_READ
* TRANSACTION\_SERIALIZABLE

Any given database may not support all of these levels.

**20. What is resultset?**

The ResultSet represents set of rows retrieved due to query execution.

ResultSet rs = stmt.executeQuery(sqlQuery);

**21. What are the types of resultsets?**

The values are defined in the class java.sql.Connection and are:

* TYPE\_FORWARD\_ONLY specifies that a resultset is not scrollable, that is, rows within it can be advanced only in the forward direction.
* TYPE\_SCROLL\_INSENSITIVE specifies that a resultset is scrollable in either direction but is insensitive to changes committed by other transactions or other statements in the same transaction.
* TYPE\_SCROLL\_SENSITIVE specifies that a resultset is scrollable in either direction and is affected by changes committed by other transactions or statements within the same transaction.

***Note****: A TYPE\_FORWARD\_ONLY resultset is always insensitive.*

**22. What’s the difference between TYPE\_SCROLL\_INSENSITIVE and TYPE\_SCROLL\_SENSITIVE?**

|  |  |
| --- | --- |
| **TYPE\_SCROLL\_INSENSITIVE** | **TYPE\_SCROLL\_SENSITIVE** |
| An insensitive resultset is like the snapshot of the data in the database when query was executed. | A sensitive resultset does NOT represent a snapshot of data, rather it contains points to those rows which satisfy the query condition. |
| After we get the resultset the changes made to data are not visible through the resultset, and hence they are known as insensitive. | After we obtain the resultset if the data is modified then such modifications are visible through resultset. |
| Performance not effected with insensitive. | Since a trip is made for every ‘**get’** operation, the performance drastically get affected. |

**22. What is rowset?**

A RowSet is an object that encapsulates a set of rows from either Java Database Connectivity (JDBC) result sets or tabular data sources like a file or spreadsheet. RowSets support component-based development models like JavaBeans, with a standard set of properties and an event notification mechanism.

|  |
| --- |
|  |

**24. What are the different types of RowSet?**

There are two types of RowSet are there. They are:

* **Connected**- A connected RowSet object connects to the database once and remains connected until the application terminates.
* **Disconnected -** A disconnected RowSet object connects to the database, executes a query to retrieve the data from the database and then closes the connection. A program may change the data in a disconnected RowSet while it is disconnected. Modified data can be updated in the database after a disconnected RowSet reestablishes the connection with the database.

**25. What is the need of BatchUpdates?**

The BatchUpdates feature allows us to group SQL statements together and send to database server in one single trip.

**26. What is a DataSource?**

A DataSource object is the representation of a data source in the Java programming language. In basic terms,

* A DataSource is a facility for storing data.
* DataSource can be referenced by JNDI.
* Data Source may point to RDBMS, file System, any DBMS etc...

|  |
| --- |
|  |

**27. What are the advantages of DataSource?**

The few advantages of data source are:

* An application does not need to hardcode driver information, as it does with the DriverManager.
* The DataSource implementations can easily change the properties of data sources. *For example*: There is no need to modify the application code when making changes to the database details.
* The DataSource facility allows developers to implement a DataSource class to take advantage of features like connection pooling and distributed transactions.

**28. What is connection pooling? What is the main advantage of using connection pooling?**

A connection pool is a mechanism to reuse connections created. Connection pooling can increase performance dramatically by reusing connections rather than creating a new physical connection each time a connection is requested...