## [51 Java Servlet Interview Questions And Answers For Experienced (codingcompiler.com)](https://codingcompiler.com/java-servlet-interview-questions-answers/)

## **JDBC Interview Questions And Answers**

#### 1. What is ANSI SQL?

SQL (structured query language) is a formal non-procedural programming language used to create, modify, and manage data in an arbitrary relational database controlled by an appropriate database management system (DBMS). SQL is based on the calculation of tuples. The SQL standard is defined using ANSI code.

\* The question “tell me about SQL” is very broad and does not fit into the scope of this article. By reading any info from the Internet, for example:

#### 2. The main elements of the database are tables, procedures, functions, constraints, etc.

**A field** is a minimal element of a database containing one indivisible quantum of information. Each field is characterized by the *name* and *type of*data stored in it.

**A record** is a collection of several heterogeneous fields, describing some essence of a subject domain.

**A database table** is a collection of uniform records.

**A stored procedure** is a database object, which is a set of SQL statements that is compiled once and stored on the server. Stored procedures are very similar to ordinary procedures of high-level languages, they can have input and output parameters and local variables, they can be used for numerical calculations and operations on character data, the results of which can be assigned to variables and parameters.

In stored procedures, standard database operations (both DDL and DML) can be performed. In addition, cycles and branches are possible in stored procedures, that is, they can use instructions to control the execution process.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | CREATE [OR REPLACE] PROCEDURE procedure\_name  [ (parameter [, parameter, …]) ] IS              BEGIN              executable sentences         [EXCEPTION              exception handlers]         END [procedure\_name]; |

**A function**  is a subroutine that calculates a value. There are a large number of built-in functions (they can vary for different databases).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE [OR REPLACE] FUNCTION function\_name  [ (parameter [, parameter, …]) ]  RETURN datatype IS | AS         BEGIN  executable sentences         [EXCEPTION  exception handlers]         END [function\_name]; |

Constraints (constraints) – declaration of rules (restrictions), which must  correspond to the data in the table.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE TABLE table\_name  (  column\_name1 data\_type(size) constraint\_name,  column\_name2 data\_type(size) constraint\_name,  column\_name3 data\_type(size) constraint\_name,  ….  );  For example, to specify a primary key, you can use the following entry:  CONSTRAINT <constraint name> PRIMARY KEY (<list of primary key columns>) |

Types of constraints:

* + **NOT NULL** column cannot contain NULL values.
  + **UNIQUE** – each row in a column must have a unique value.
  + **PRIMARY KEY** is a combination of NOT NULL and UNIQUE.
  + **FOREIGN KEY** – provides referential integrity. Indicates that the referencing data has a match in another table.
  + **CHECK** – check for a specific rule execution.
* **DEFAULT** – sets the default value for the column

#### 3. How do you understand null in databases?

The meaning of NULL values ​​is the absence of information or the inapplicability of this attribute in this tuple.

NULL value can mean that the value is not applicable to this column (for example, in the column “flight speed” for the table animals and “Elephant” entries)

NULL value can also mean the absence of information. Replacing the missing values, for example, with -1, ” or something, is not correct.

#### 4. Aggregate functions, how they work with null. Do not forget about group by and having

The standard provides the following aggregate functions:

|  |  |
| --- | --- |
| **Title** | **Description** |
| COUNT (\*) | Returns the number of rows in the record source. |
| COUNT | Returns the number of values ​​in the specified column. |
| SUM | Returns the sum of the values ​​in the specified column. |
| Avg | Returns the average value in the specified column. |
| MIN | Returns the minimum value in the specified column. |
| MAX | Returns the maximum value in the specified column. |

All of these functions return a single value. The COUNT , MIN, and MAX functions are applicable to data of any type, while SUM and AVG are used only for data of the numeric type. The difference between the COUNT (\*) function and COUNT (column name | expression) is that the second (like the other aggregate functions) does not take into account NULL values ​​when calculating.

The GROUP BY clause is used to define groups of output strings to which aggregate functions can be applied ( COUNT , MIN , MAX , AVG, and SUM ).

If the WHERE clause defines a predicate for filtering rows, then the HAVING clause is applied after grouping to define a similar predicate that filters groups by the values ​​of aggregate functions.

This clause is necessary to check the values ​​that are obtained using the aggregate function not from individual rows of the source of records defined in the FROM clause , but from groups of such rows. Therefore, such a check cannot be contained in the WHERE clause.

#### 5. How is it better to add a large number of records to a table?

You can interpret this question in different ways. One answer is to use a subquery:

|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO <table name> [(<column name>, …)]  SELECT \*  FROM <table name>  WHERE value = ‘something’; |

#### 6. What is the first normal form and normalization process? What are the normal forms?

The first normal form (1NF) is the basic normal form of the relation in the relational data model.

The normal form is a relation property in the relational data model, which characterizes it in terms of redundancy, potentially leading to logically erroneous results of sampling or changing data.

Normal form is defined as the set of requirements that a relation must satisfy. Normal forms: the first normal form, the second normal form, the third normal form, the Boyes-Codd normal form, the fourth normal form, the fifth normal form.

The topic is basic and necessary for detailed study. The size of the answer to this question does not fit the format of this article.

#### 7. What is the meaning of the DBMS index, how are they arranged, how are they stored? How would you implement the same functionality?

Index (eng. Index) – a database object created to improve the performance of data retrieval. Tables in the database can have a large number of rows that are stored in arbitrary order, and searching them by a given criterion by sequentially viewing the table row by row can take a lot of time.

The index is formed from the values ​​of one or several columns of the table and pointers to the corresponding rows of the table and, thus, allows you to search for rows that meet the search criteria. Acceleration of work with the use of indexes is achieved primarily due to the fact that the index has a structure optimized for the search – for example, a balanced tree.

The answer to this question also does not fit in one article. Learn by yourself.

#### 8. What is the JDBC API and when is it used?

JDBC is a standard for application interaction with various DBMS. JDBC is based on the driver concept, which allows you to get a connection to the database using a special url.

The JDBC APIs are in the java.sql and javax.sql packages. Using the JDBC API, you can create connections to the database, execute SQL queries, stored procedures, and process the results. The JDBC API makes it easy to work with databases from Java programs.

#### 9. What is the JDBC Driver and what different types of JDBC drivers do you know?

JDBC is based on the concept of so-called drivers that allow you to get a connection to a database at a specially described URL. Drivers can load dynamically (while the program is running). Having loaded, the driver itself registers itself and is called automatically when the program requires a URL containing the protocol for which the driver is responsible.

There are four types of drivers. The Java program works with the database in two parts. The first part is the JDBC API, and the second is the driver, which does all the work. Each type defines an implementation of the JDBC driver in terms of increasing platform independence, performance, and ease of administration. These four types are as follows:

**Type 1: JDBC-ODBC Bridge**  ( **JDBC-ODBC Bridge plus ODBC Driver)** – translates JDBC to ODBC and uses the ODBC driver to interact with the database. Sun included one such driver in the JDK – the JDBC / ODBC bridge. Now there are more successful implementations.

**Type 2: Native API / partly Java driver (Native API partly Java technology-enabled driver)** – translates JDBC calls into database-specific calls such as SQL Server, Informix, Oracle or Sybase. The type 2 driver communicates directly with the database server, hence it requires that some binary code be on the side of the client machine.

**Type 3: Network Protocol / Pure Java Driver for Database Middleware**– uses a three-tier architecture, where JDBC calls are sent to an intermediate so-called. the application server, then this server translates calls (explicitly or indirectly) into calls to the native interface specific to the DBMS for further access to the database. If the middle layer server is written in Java, it can use type 1 and type 2 drivers for translating JDBC.

**Type 4: Native protocol / “pure” Java driver (Direct-to-Database Pure Java Driver)** – converts JDBC calls into a specific database vendor protocol, so client applications can directly contact the database server. Type 4 drivers are fully implemented in Java in order to achieve platform independence and troubleshoot administration and deployment problems.

#### 10. How does the JDBC API help achieve weak communication between the Java program and the JDBC Drivers API?

The JDBC API uses reflection in java to achieve a weak connection between the Java program and the JDBC drivers. The driver is actually loaded once using Class.forName () , and then the JDBC API in Java is used. Thus, we write the code without really thinking about which database we will work with. If necessary, it is enough to specify a different driver and not rewrite a large amount of code.

### Java JDBC Interview Questions And Answers

#### 11. What is JDBC Connection? Show the steps to connect the program to the database.

JDBC Connection – a connection established with a database server. This is a kind of session or socket connection. Creating a JDBC connection requires only two steps:

1) Register and load the driver using Class.forName () . The driver class will be registered for the DriverManager and loaded into memory.

2) Using DriverManager.getConnection () to get the Connection object . Pass the database URL, name and password to the method.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | Connection con = null;  try{     // load the Driver Class     Class.forName(“com.mysql.jdbc.Driver”);     // create the connection now     con = DriverManager.getConnection(“jdbc:mysql://localhost:3306/UserDB”,                     “pankaj”,                     “pankaj123”);     }catch (SQLException e) {             System.out.println(“Check database is UP and configs are correct”);             e.printStackTrace();     }catch (ClassNotFoundException e) {             System.out.println(“Please include JDBC MySQL jar in classpath”);             e.printStackTrace();     } |

#### 12. How is the JDBC DriverManager class used?

JDBC DriverManager is a factory through which you can get a Database Connection object. After loading the JDBC driver into memory, it registers itself with the DriverManager (which can be checked by looking at the source code for the JDBC Driver class). DriverManager is used to get a connection using registered drivers ( getConnection () method ).

#### 13. How to get information about the database server from a java program?

Using the DatabaseMetaData interface object,  you can get detailed information about the server. After connecting to the database, we can call the getMetaData () method and get a DatabaseMetaData object . There are many methods for obtaining various information, such as the database version, configuration, etc.

|  |  |
| --- | --- |
| 1  2 | DatabaseMetaData metaData = con.getMetaData();  String dbProduct = metaData.getDatabaseProductName(); |

#### 14. What is the JDBC Statement?

JDBC API Statement is used to perform SQL queries to the database. The Statement object can be retrieved using the Connection.getStatement () method . By calling the execute () , executeQuery () , executeUpdate () , etc. methods , you can execute various static SQL queries.

In the case of dynamically generated SQL queries inside a java program, when user input may not be validated, you can use SQL injection.

By default, only one ResultSet object for each Statement can be opened at the same time. Thus, if you need to work with several ResultSetobjects at the same time, we must use different Statement objects . All execute () methods in the Statement interface will close the current open ResultSet object at run time.

#### 15. What are the differences between execute, executeQuery, executeUpdate?

There are several ways to execute SQL queries depending on the type of this query. For this, the Statement interface has three different methods: executeQuery () , executeUpdate () , and also execute () . Consider them separately.

The most basic method, executeQuery (), is required for queries that result in a single set of values, such as for SELECT queries. Returns a ResultSet that cannot be null even if no result was found for the query result.

Method execute () is used when SQL statements return more than one dataset, more than one update count, or both. The method returns true if the result is a ResultSet, like a SELECT query. Returns false if there is no ResultSet, for example, when you are requesting Insert, Update. Using the getResultSet () methods, we can get a ResultSet, and getUpdateCount () is the number of updated records.

The executeUpdate () method is used to execute INSERT, UPDATE or DELETE statements, as well as for DDL (Data Definition Language) statements, such as CREATE TABLE and DROP TABLE. The result of an INSERT, UPDATE, or DELETE statement is a modification of one or more columns in zero or more rows in a table.

MethodexecuteUpdate () returns an integer indicating how many lines have been modified. For expressions of type CREATE TABLE and DROP TABLE that do not operate on strings, the value returned by the executeUpdate () method is always zero.

All methods for executing SQL queries close the previous result set (result set) for this Statement object. This means that before you execute the following query on the same Statement object, you must finish processing the results of the previous one (ResultSet).

#### 16. What is JDBC PreparedStatement?

The PreparedStatement object is used to execute precompiled SQL queries with or without input (IN) parameters. We can use setters to set values ​​in a query. Since Since a PreparedStatement is precompiled, it can be effectively used many times.

PreparedStatement is considered a better choice than a Statement, since it automatically processes special characters, as well as prevents the so-called SQL injection attack (when you can substitute your code in a request).

#### 17. How to set NULL values ​​in JDBC Prepared Statement?

Using the setNull () method to set a null variable as a parameter. This method takes an index and SQL type as arguments:  s.setNull (10, java.sql.Types.INTEGER);

#### 18. How is the getGeneratedKeys () method used in the Statement?

If automatic key generation is used in the table, then the method getGeneratedKeys () , which returns the generated key , is used to obtain them .

#### 19. What are the advantages in using PreparedStatement over Statement?

* + PreparedStatement allows you to prevent attacks like SQL injection, because it automatically escapes special characters.
  + PreparedStatement allows you to use dynamic queries with the implementation of parameters.
  + PreparedStatement is faster than Statement. This is especially noticeable with frequent use of a PreparedStatement or when used to invoke a group of queries.
* PreparedStatement allows you to write object-oriented code using setters \ getters. While using a Statement, you must use string concatenation to create a query. For large queries, concatenation looks at least large, and also carries a high risk of error in the query.

#### 20. What are the limitations of PreparedStatement and how to overcome them?

In PreparedStatement, you cannot use queries directly with IN (input) parameters. There are some workarounds:

* 1. **Run Single Queries** – poor performance and generally not recommended to do so.
  2. **Use the Stored Procedure (Stored Procedures)** – are specific to a specific database and therefore bad for applications with the ability to connect to different databases.
  3. **Creating a PreparedStatement Query dynamically** is a good solution, but with the loss of PreparedStatement caching.

1. **Using NULL in PreparedStatement Query** is a good solution if you know the maximum number of IN variables. You can expand to use an unlimited number of parameters by using a split into parts.

### Advanced JDBC Interview Questions And Answers

#### 21. What is JDBC ResultSet?

JDBC ResultSet is an interface whose object is created as a result of a database query. It can be represented in the form of a data table, which was formed in response to a query.

The ResultSet object supports a cursor that points to the current row of data. During initialization, the cursor is set to the first line. To move in rows, use the next () method . If there are rows after the current position, the next () method returns true, which can be used to iterate over the table of the results.

By default, the ResultSet object is not modifiable and supports a cursor that is only capable of moving forward. To circumvent this limitation, you can use the following design, which will allow bidirectional movement on the table, as well as the possibility of updating:

|  |  |
| --- | --- |
| 1  2 | Statement stmt = con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE,                                 ResultSet.CONCUR\_UPDATABLE); |

The ResultSet object closes automatically when you close the object that generated it. Also, closing will occur when the query is re-executed or a result is returned from another result set.

To use the ResultSet getter, you can use the column name or an index that starts with 1.

#### 22. What are the different types of JDBC ResultSet?

When creating a Statement, you can specify the different types of ResultSet to get.

Three types of ResultSet objects:

* 1. **ResultSet.TYPE\_FORWARD\_ONLY** : The default type. Supports cursor movement only in the forward direction.
  2. **ResultSet.TYPE\_SCROLL\_INSENSITIVE** : Bidirectional cursor. The object is not sensitive to changes that occurred with the table after receiving the result.

1. **ResultSet.TYPE\_SCROLL\_SENSITIVE** : Bidirectional cursor. The object is sensitive to changes that have occurred to the database after creating the ResultSet object.

Two types of thread-safe ResultSet objects:

* 1. **ResultSet.CONCUR\_READ\_ONLY** : Supports only read (read only). It is applied by default.

1. **ResultSet.CONCUR\_UPDATABLE** : Supports the ResultSet update method for updating rows in a data table.

#### 23. How are the setFetchSize () and SetMaxRows () methods used in the Statement?

To limit the number of rows that a query can return, use the setMaxRows (int i) method . Of course, this result can be obtained using a SQL query (for example, there is a LIMIT command for MySQL).

To understand the setFetchSize () method, you need to understand how the Statement and ResultSet work. When a database request is made, the result is processed and stored in the database cache and returned as a ResultSet. ResultSet is a cursor that references a result in a database.

Now let’s say we have a query that returns 100 rows and we set setFetchSize (10) . Now, for each access to the database, only 10 lines are allocated and it will take 10 queries to get all the data. Choosing the optimal amount of fetchSize () can improve the performance of executing a large number of calls to each line and in the case of a large number of lines in the output.

The fetchSize value can be specified inside the Statement object, but it can be overridden in the ResultSet object with setFetchSize () .

#### 24. How to call Stored Procedures using JDBC API?

Stored procedures are groups of SQL queries that are compiled into a database and can be invoked using the JDBC API. The CallableStatement object is used to call stored procedures. We need to set the output parameters OUT before CallableStatement.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | CallableStatement stmt = con.prepareCall(“{call insertEmployee(?,?,?,?,?,?)}”);  stmt.setInt(1, id);  stmt.setString(2, name);  stmt.setString(3, role);  stmt.setString(4, city);  stmt.setString(5, country);  //register the OUT parameter before calling the stored procedure  stmt.registerOutParameter(6, java.sql.Types.VARCHAR);    stmt.executeUpdate(); |

#### 25. What is JDBC Batch Processing and what are its benefits?

It may be necessary to immediately execute a group of similar queries, for example, when loading data from CSV files of a relational database. This can be done simply by using a Statement or PreparedStatement to step through these requests in steps.

There is another possibility in the JDBC API that provides the ability to execute a group of queries at once. This type of task is performed using the JDBC API Batch Processing.

The JDBC API supports batch processing using the addBatch () and executeBatch () methods of the Statement and PreparedStatement. The advantages of this approach include faster work, because Calls to the database can be significantly less.

#### 26. What is JDBC Transaction Management and why is it needed?

By default, when creating a database connection, auto-commit mode will be selected. This means that each time the request is executed, it will be automatically confirmed upon completion.

Each SQL query is transactional and executing any DML or DDL queries upon their completion will be accepted (saved) by the database. If we have to refuse to save the execution of a query (or a group of queries) in case something went wrong, then we can use transaction support in the JDBC API.

Using the setAutoCommit (boolean flag) method, you can disable auto commits in a particular connection. It should be noted that when auto-commit is disabled, not one change will be saved in the database until the commit () method is called and this should be monitored.

The database server will block the necessary part of the database before confirming the transaction, and since this is a resource-intensive task, you must confirm the transaction immediately after completing the task.

#### 27. How to roll back a JDBC transaction?

To do this, use the Connection rollback () method of the object , which rolls back the transaction. All changes to the transaction will be canceled and the database lock on this Connection object will be canceled.

#### 28. What is JDBC Savepoint and how is it used?

JDBC Savepoint allows you to create “checkpoints” in a transaction with which we can roll back not the entire transaction, but only a part to the save point. Any savepoint is automatically released and becomes unavailable after confirming the transaction or its rollback. A rollback to the save point makes all subsequent saves unavailable and it will be impossible to return to them.

#### 29. Tell us about the JDBC DataSource. What advantages does it give?

JDBC DataSource is a javax.sql package interface and is more advanced than DriverManager for connecting to a database. We can use the DataSource to create a database connection and implement a driver class that will do all the work of maintaining the connection. In addition to connecting through the Database, DataSource provides the following additional features:

* + PreparedStatement caching to speed query processing
  + Connection timeout settings
  + Logging capabilities
  + ResultSet Maximum Size Threshold
* Connection Pooling support in a servlet container using JNDI support.

#### 30. How to create a JDBC connection pool using JDBC DataSource and JNDI in Apache Tomcat Server?

Creating a JDBC connection pool when using Tomcat requires some simple steps. You must create a JDBC JNDI resource in the server configuration file (server.xml or context.xml).

server.xml :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <Resource name=”jdbc/MyDB”       global=”jdbc/MyDB”       auth=”Container”       type=”javax.sql.DataSource”       driverClassName=”com.mysql.jdbc.Driver”       url=”jdbc:mysql://localhost:3306/DataBaseName”       username=”root”       password=”admin”         maxActive=”100″       maxIdle=”20″       minIdle=”5″       maxWait=”10000″/> |

context.xml :

|  |  |
| --- | --- |
| 1  2  3  4 | <ResourceLink name=”jdbc/MyLocalDB”                 global=”jdbc/MyDB”                 auth=”Container”                 type=”javax.sql.DataSource” /> |

In a web application using InitialContext, we use the following entry to search for the JNDI resource specified in the configuration above. And then you can get the connection.

|  |  |
| --- | --- |
| 1  2 | Context ctx = new InitialContext();  DataSource ds = (DataSource) ctx.lookup(“java:/comp/env/jdbc/MyLocalDB”); |

### JDBC Interview Questions And Answers For Experienced

#### 31. Tell us about the Apache DBCP API.

When using a DataSource to get a database connection, there is a problem of tight connection between the code and the DataSource implementation driver. In addition, most of the code is pattern-repeatable (the so-called boilerplate code), except for the choice of the data source implementation class.

The Apache DBCP API helps us get rid of these problems by providing a DataSource implementation that works as an abstraction layer between our program and various JDBC drivers. The Apache DBCP API library is based on the Commons Pool library, so you need to make sure that all the necessary dependencies are properly connected to the project.

#### 32. What are the isolation levels of connections in JDBC?

The transaction isolation level is a value that determines the level at which inconsistent data is allowed in a transaction, that is, the degree of isolation of one transaction from another. A higher level of isolation improves data accuracy, but it may decrease the number of concurrent transactions. On the other hand, a lower level of isolation allows for more concurrent transactions, but reduces the accuracy of the data.

When we use transactions in JDBC to ensure data integrity, the DBMS uses locks to block the access of other accesses to the data involved in the transaction. Such locks are necessary to prevent dirty reading (Dirty Read), non-repeatable reading (Non-Repeatable Read) and phantom reading (Phantom-Read).

The isolation level of the JDBC transaction used by the DBMS for the locking mechanism can be set using the setTransactionIsolation () method . You can get information about the level of isolation used by the Connection getTransactionIsolation () method .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Isolation level** | **Transaction** | **Dirty read** | **Non-Repeatable Read** | **Phantom Read** |
| TRANSACTION\_NONE | Not Supported | Not applicable | Not applicable | Not applicable |
| TRANSACTION\_READ\_COMMITTED | Supported | Prevented | Allowed | Allowed |
| TRANSACTION\_READ\_UNCOMMITTED | Supported | Allowed | Allowed | Allowed |
| TRANSACTION\_REPEATABLE\_READ | Supported | Prevented | Prevented | Allowed |
| TRANSACTION\_SERIALIZABLE | Supported | Prevented | Prevented | Prevented |

#### 33. What do you know about JDBC RowSet? What are the different types of RowSet?

JDBC RowSet contains tabular data in a more flexible format than ResultSet. All RowSet objects are derived from ResultSet, so they have all the capabilities of a ResultSet with some additional functions. The RowSet interface is defined in the javax.sql package. The following additional functions provided by RowSet can be highlighted:

* + Functions similar to Java Beans with properties and get \ set methods for working with them. RowSet uses the JavaBeans event model. As a result, you can send notifications to any registered component for such events, for example, cursor movement, updates / insert / delete from the row and change the contents of the RowSet.
* RowSet objects support scrolling by data (scrollable), as well as being updated by default. So if the DBMS does not support scrolling or an updated ResultSet, we can use RowSet to get these functions.

RowSet can be divided into two types:

* 1. **Connected RowSet Objects** – these objects connect to the database and are very similar to ResultSet objects. The JDBC API provides only one connection to the RowSet object –  javax.sql.rowset.JdbcRowSet  and this is the standard implementation of the com.sun.rowset.JdbcRowSetImpl class  .

1. **Disconnected RowSet Objects** – these RowSet objects do not require a database connection. They are more lightweight and can be serialized. Such objects are well suited for data transmission over the network. There are four implementations of this type of object (disconnected RowSet objects).

Briefly consider the four implementation of **Disconnected RowSet Objects** :

* + **CachedRowSet** – objects can receive a connection and execute a query, read ResultSet data to populate RowSet data. We can manage and update the data at the time of disconnection from the database and record the changed data during the next connection.
  + **WebRowSet** obtained from CachedRowSet – such objects can read and write XML documents.
  + **JoinRowSet** obtained from WebRowSet – can form SQL JOIN without connecting to a data source.
* **FilteredRowSet is** obtained from WebRowSet – support for applying filtering criteria; therefore, only selected (received) data is visible.

#### 34. What is the difference between ResultSet and RowSet?

RowSet objects are derived from ResultSet, so they have all the capabilities of a ResultSet with some additional functions. One of the significant advantages is the ability to work with data without connecting to the database, as well as their lightness and the ability to send data objects over the network.

Whether to use a ResultSet or RowSet depends on your requirements. ResultSet may be suitable for long-term connections, while RowSet will be the best choice for connections to databases with the possibility of disconnection and processing of received data.

#### 35. Give an example of the most common exceptions in JDBC.

Some of the most common JDBC exceptions are:

* + java.sql.SQLException is the base class for JDBC exceptions.
  + java.sql.BatchUpdateException – occurs when batch requests are excepted. May depend on the type of JDBC driver that the base SQLException may throw instead.
  + java.sql.SQLWarning – for warning messages of various SQL operations.
* java.sql.DataTruncation – when data values ​​are unexpectedly truncated due to reasons independent of exceeding MaxFieldSize.

#### 36. Tell us about the data types CLOB and BLOB in JDBC.

Character Large OBjects (CLOBs) is a data type (internal character object) used to store large objects. When selecting a value of any LOB type, a pointer is returned via the SELECT statement, and not the value itself; In addition, LOB types can be external. This data type is suitable for storing textual information that may fall outside the normal VARCHAR data type (upper limit of 32 KB).

An internal blob (BLOB) is a large binary object that can contain a variable amount of data. This data type can store data larger than VARBINARY (32K limit). A type of data intended primarily for storing images, audio and video, as well as compiled program code.

#### 37. What do you know about dirty read in JDBC? What isolation level does this type of reading prevent?

“Dirty” reading (eng. Dirty read) – reading data added or modified by a transaction, which is subsequently not confirmed (rolled back). Obtaining an invalid value after the transaction (after rolling back the transaction) may lead to unexpected results.

|  |  |
| --- | --- |
| **Transaction 1** | **Transaction 2** |
|  | SELECT f2 FROM tbl1 WHERE f1=1; |
| UPDATE tbl1 SET f2=f2+1 WHERE f1=1; |  |
|  | SELECT f2 FROM tbl1 WHERE f1=1; |
| ROLLBACK WORK; |  |

Dirty Read can be prevented using the following isolation levels:  TRANSACTION\_READ\_COMMITTED , TRANSACTION\_REPEATABLE\_READand TRANSACTION\_SERIALIZABLE .

#### 38. What are the two commit phases?

When we work in distributed systems where multiple databases are involved, we must use a protocol with 2 commit phases. Phase 2 commit protocol is an atomic protocol for distributed systems.

In the first step, the transaction manager passes commit-request to all transaction resources. If all transaction resources are OK, then the transaction manager records transaction changes for all resources. If any of the transaction resources notifies of the cancellation, then the transaction manager can roll back all transaction changes.

#### 39. Give an example of the different types of blocking in JDBC.

At a broader level, there are two types of locking mechanisms to prevent data from being damaged due to multiple users working with data simultaneously. According to the implementation logic, there are two types of locks.

**Optimistic locking** does not limit the modification of the data being processed by third-party sessions, but before starting the intended modification it requests the value of some selected attribute of each of the data lines (usually the VERSION name and the integer type with the initial value 0 are used).

Before writing modifications to the database, the value of the selected attribute is checked and, if it has changed, the transaction is rolled back or different collision resolution schemes are applied. If the value of the selected attribute has not changed – the modifications are recorded while changing the value of the selected attribute (for example, increment) to signal to other sessions that the data has changed.

**Pessimistic blocking**– it is superimposed before the intended modification of the data on all the lines that such modification presumably affects. All the time such a lock is in effect, modification of data from third-party sessions is excluded; data from blocked rows is available according to the isolation level of the transaction. Upon completion of the proposed modification, a consistent recording of the results is guaranteed.

### Top 47 JDBC Interview Questions & Answers For Experienced

#### 40. How do you understand DDL and DML expressions?

Data Definition Language (DDL) is a family of computer languages ​​used in computer programs to describe the structure of databases. DDL functions are defined by the first word in a sentence (often called a query), which is almost always a verb. In the case of SQL, these are verbs – “create” (“create”), “alter” (“change”), “drop” (“delete”).

Data Manipulation Language (DML) is a family of computer languages ​​used in computer programs or by database users to get, insert, delete, or change data in databases. DML functions are defined by the first word in a sentence (often called a query), which is almost always a verb. In the case of SQL, these verbs are “select” (“select”), “insert” (“insert”), “update” (“update”), and “delete” (“delete”). This turns the nature of the language into a series of mandatory statements (commands) to the database.

#### 41. What is the difference between java.util.Date and java.sql.Date?

java.util.Date contains information about the date and time, whereas java.sql.Date contains information about the date, but has no information about the time. If you need to store time information in the database, it is desirable to use the Timestamp or DateTime fields.

Java.util.Date is the main universal object. It simply stores the date (as long).

java.sql.Date extends java.util.Date and adds the following functionality:

1) toString prints the date as “yyyy-mm-dd”, rather than as a specific locale of the string (locale).

2) Added the valueOf method to read the “yyyy-mm-dd” format strings and further parse it into the sql.Date object.

#### 42. How to insert an image or raw data into a database?

To do this, you can use the BLOB data type to insert a picture or binary data into the database.

#### 43. What can you tell about phantom reading? What level of insulation prevents it?

The situation when, when re-reading in the same transaction, the same sample gives different sets of rows. Suppose there are two transactions opened by various applications in which the following SQL statements are executed:

|  |  |
| --- | --- |
| **Transaction 1** | **Transaction 2** |
|  | SELECT SUM(f2) FROM tbl1; |
| INSERT INTO tbl1 (f1,f2) VALUES (15,20); |  |
| COMMIT; |  |
|  | SELECT SUM(f2) FROM tbl1; |

In Transaction 2, a SQL statement is executed using all the values ​​of the f2 field. Then, in transaction 1, a new line is inserted, causing the repeated execution of the SQL statement in transaction 2 to produce a different result. This situation is called phantom reading. It differs from non-repeating reading in that the result of repeated access to data has changed not because of the change / deletion of this data itself, but because of the appearance of new (phantom) data.

Phantom read can only be prevented at the isolation level –  TRANSACTION\_SERIALIZABLE .

#### 44. What is SQL Warning? How to return SQL warnings in a JDBC program?

SQLWarning is a subclass of SQLException that we can get by calling the getWarnings () method on Connection , Statement , ResultSet  objects  . SQL Warnings does not stop the execution of the query, but displays warning messages for the user.

#### 45. How to run Oracle Stored Procedure with IN / OUT database objects?

If the Oracle stored procedure contains IN / OUT parameters as DB objects, then we need to create an array of objects of the same size in the program, and then use it to create an Oracle STRUCT  object. Then we can set this STRUCT object to a database object by calling the setSTRUCT () method .

#### 46. ​​Give an example of java.sql.SQLException occurrence: No suitable driver found.

The exception java.sql.SQLException: No suitable driver found can be caused, for example, by an incorrectly formatted SQL address string. You can get this exception in a simple Java application both via  DriverManager , tick and using the JNDI DataSource . The exception stack tracing is shown below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | org.apache.tomcat.dbcp.dbcp.SQLNestedException: Cannot create JDBC driver of class ‘com.mysql.jdbc.Driver’ for connect URL ”jdbc:mysql://localhost:3306/UserDB’     at org.apache.tomcat.dbcp.dbcp.BasicDataSource.createConnectionFactory(BasicDataSource.java:1452)     at org.apache.tomcat.dbcp.dbcp.BasicDataSource.createDataSource(BasicDataSource.java:1371)     at org.apache.tomcat.dbcp.dbcp.BasicDataSource.getConnection(BasicDataSource.java:1044)  java.sql.SQLException: No suitable driver found for ‘jdbc:mysql://localhost:3306/UserDB     at java.sql.DriverManager.getConnection(DriverManager.java:604)     at java.sql.DriverManager.getConnection(DriverManager.java:221)     at com.journaldev.jdbc.DBConnection.getConnection(DBConnection.java:24)     at com.journaldev.jdbc.DBConnectionTest.main(DBConnectionTest.java:15)  Exception in thread “main” java.lang.NullPointerException |

You can see that we use the URL of the form – ‘the jdbc: the mysql: // localhost: 3306 / userdb , while should be specified URL – the jdbc: the mysql: // localhost: 3306 / userdb .

#### 47. Best Practices in JDBC.

Some Best Practices in JDBC:

* + Always close the database resources immediately after working with them. Connection, Statement, ResultSet, and other JDBC objects have a close () method that allows you to close them.
  + Always explicitly close the ResultSet, Statement and Connection result set in code, because if you use a connection pool, the connection can be returned to the pool, leaving open result sets and statement objects and there will be a leak.
  + Close the resources at the end of the finally block to ensure that they are closed even if an exception occurs.
  + Use batch processing for repeated requests.
  + Always use PreparedStatement instead of Statement to avoid SQL Injection and take advantage of the precompiled and cached PreparedStatement query.
  + If you extract large data arrays in a result set, then set the optimal fetchSize value in advance, which will help you get better performance.
  + The database server may not support all isolation levels, so check them beforehand.
  + More stringent isolation levels can lead to poor performance, so make sure you use the optimal set of isolation levels for your database connections.
  + If you are creating database connections from a web application, try using JDBC DataSource resources by using the JNDI context to reuse connections.
* Try using disconnected RowSet when you need to work with ResultSet for a long time.

# JDBC Interview Questions

A list of top frequently asked JDBC interview questions and answers is given below.

### 1) What is JDBC?

JDBC is a Java API that is used to connect and execute the query to the database. JDBC API uses JDBC drivers to connect to the database. JDBC API can be used to access tabular data stored into any relational database.

 [More details.](https://www.javatpoint.com/jdbc-tutorial)

### 2) What is JDBC Driver?

JDBC Driver is a software component that enables Java application to interact with the database. There are 4 types of JDBC drivers:

1. **JDBC-ODBC bridge driver:** The JDBC-ODBC bridge driver uses the ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of the thin driver. It is easy to use and can be easily connected to any database.
2. **Native-API driver (partially java driver):** The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in Java. Its performance is better than JDBC-ODBC bridge driver. However, the native driver must be installed on each client machine.
3. **Network Protocol driver (fully java driver):** The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is entirely written in Java. There is no requirement of the client-side library because of the application server that can perform many tasks like auditing, load balancing, logging, etc.
4. **Thin driver (fully java driver):** The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as the thin driver. It is entirely written in Java language. Its performance is better than all other drivers however these drivers depend upon the database.

[More details.](https://www.javatpoint.com/jdbc-driver)

### 3) What are the steps to connect to the database in java?

The following steps are used in database connectivity.

* **Registering the driver class:**

The forName() method of the Class class is used to register the driver class. This method is used to load the driver class dynamically. Consider the following example to register OracleDriver class.

* 1. Class.forName("oracle.jdbc.driver.OracleDriver");
* **Creating connection:**

The getConnection() method of DriverManager class is used to establish the connection with the database. The syntax of the getConnection() method is given below.

* 1. 1) **public** **static** Connection getConnection(String url)**throws** SQLException
  2. 2) **public** **static** Connection getConnection(String url,String name,String password)
  3. **throws** SQLException

Consider the following example to establish the connection with the Oracle database.

* 1. Connection con=DriverManager.getConnection(
  2. "jdbc:oracle:thin:@localhost:1521:xe","system","password");
* **Creating the statement:**

The createStatement() method of Connection interface is used to create the Statement. The object of the Statement is responsible for executing queries with the database.

* 1. **public** Statement createStatement()**throws** SQLException

consider the following example to create the statement object

* 1. Statement stmt=con.createStatement();
* **Executing the queries:**

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax of executeQuery() method is given below.

* 1. **public** ResultSet executeQuery(String sql)**throws** SQLException

Example to execute the query

* 1. ResultSet rs=stmt.executeQuery("select \* from emp");
  2. **while**(rs.next()){
  3. System.out.println(rs.getInt(1)+" "+rs.getString(2));
  4. }

However, to perform the insert and update operations in the database, executeUpdate() method is used which returns the boolean value to indicate the successful completion of the operation.

* **Closing connection:**

By closing connection, object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax of close() method is given below.

* 1. **public** **void** close()**throws** SQLException

Consider the following example to close the connection.

* 1. con.close();

[More details.](https://www.javatpoint.com/steps-to-connect-to-the-database-in-java)

### 4) What are the JDBC API components?

The java.sql package contains following interfaces and classes for JDBC API.

**Interfaces:**

* **Connection:** The Connection object is created by using getConnection() method of DriverManager class. DriverManager is the factory for connection.
* **Statement:** The Statement object is created by using createStatement() method of Connection class. The Connection interface is the factory for Statement.
* **PreparedStatement:** The PrepareStatement object is created by using prepareStatement() method of Connection class. It is used to execute the parameterized query.
* **ResultSet:** The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points before the first row. The executeQuery() method of Statement interface returns the ResultSet object.
* **ResultSetMetaData:** The object of ResultSetMetaData interface cotains the information about the data (table) such as numer of columns, column name, column type, etc. The getMetaData() method of ResultSet returns the object of ResultSetMetaData.
* **DatabaseMetaData:** DatabaseMetaData interface provides methods to get metadata of a database such as the database product name, database product version, driver name, name of the total number of tables, the name of the total number of views, etc. The getMetaData() method of Connection interface returns the object of DatabaseMetaData.
* **CallableStatement:** CallableStatement interface is used to call the stored procedures and functions. We can have business logic on the database through the use of stored procedures and functions that will make the performance better because these are precompiled. The prepareCall() method of Connection interface returns the instance of CallableStatement.

**Classes:**

* **DriverManager:** The DriverManager class acts as an interface between the user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. It contains several methods to keep the interaction between the user and drivers.
* **Blob:** Blob stands for the binary large object. It represents a collection of binary data stored as a single entity in the database management system.
* **Clob:** Clob stands for Character large object. It is a data type that is used by various database management systems to store character files. It is similar to Blob except for the difference that BLOB represent binary data such as images, audio and video files, etc. whereas Clob represents character stream data such as character files, etc.
* **SQLException** It is an Exception class which provides information on database access errors.

### 5) What are the JDBC statements?

In JDBC, Statements are used to send SQL commands to the database and receive data from the database. There are various methods provided by JDBC statements such as execute(), executeUpdate(), executeQuery, etc. which helps you to interact with the database.

There is three type of JDBC statements given in the following table.

|  |  |
| --- | --- |
| **Statements** | **Explanation** |
| Statement | Statement is the factory for resultset. It is used for general purpose access to the database. It executes a static SQL query at runtime. |
| PreparedStatement | The PreparedStatement is used when we need to provide input parameters to the query at runtime. |
| CallableStatement | CallableStatement is used when we need to access the database stored procedures. It can also accept runtime parameters. |

### 6) What is the return type of Class.forName() method?

The Class.forName() method returns the object of java.lang.Class object.

### 7) What are the differences between Statement and PreparedStatement interface?

|  |  |
| --- | --- |
| **Statement** | **PreparedStatement** |
| The Statement interface provides methods to execute queries with the database. The statement interface is a factory of ResultSet; i.e., it provides the factory method to get the object of ResultSet. | The PreparedStatement interface is a subinterface of Statement. It is used to execute the parameterized query. |
| In the case of Statement, the query is compiled each time we run the program. | In the case of PreparedStatement, the query is compiled only once. |
| The Statement is mainly used in the case when we need to run the static query at runtime. | PreparedStatement is used when we need to provide input parameters to the query at runtime. |

[More details.](https://www.javatpoint.com/PreparedStatement-interface)

### 8) How can we set null value in JDBC PreparedStatement?

By using setNull() method of PreparedStatement interface, we can set the null value to an index. The syntax of the method is given below.

1. **void** setNull(**int** parameterIndex, **int** sqlType) **throws** SQLException

### 9) What are the benefits of PreparedStatement over Statement?

The benefits of using PreparedStatement over Statement interface is given below.

* The PreparedStatement performs faster as compare to Statement because the Statement needs to be compiled everytime we run the code whereas the PreparedStatement compiled once and then execute only on runtime.
* PreparedStatement can execute Parameterized query whereas Statement can only run static queries.
* The query used in PreparedStatement is appeared to be similar every time. Therefore, the database can reuse the previous access plan whereas, Statement inline the parameters into the String, therefore, the query doesn't appear to be same everytime which prevents cache reusage.

### 10) What are the differences between execute, executeQuery, and executeUpdate?

|  |  |  |
| --- | --- | --- |
| **execute** | **executeQuery** | **executeUpdate** |
| The execute method can be used for any SQL statements(Select and Update both). | The executeQuery method can be used only with the select statement. | The executeUpdate method can be used to update/delete/insert operations in the database. |
| The execute method returns a boolean type value where true indicates that the ResultSet s returned which can later be extracted and false indicates that the integer or void value is returned. | The executeQuery() method returns a ResultSet object which contains the data retrieved by the select statement. | The executeUpdate() method returns an integer value representing the number of records affected where 0 indicates that query returns nothing. |

### 11) What are the different types of ResultSet?

ResultSet is categorized by the direction of the reading head and sensitivity or insensitivity of the result provided by it. There are three general types of ResultSet.

|  |  |
| --- | --- |
| Type | Description |
| ResultSet.TYPE\_Forward\_ONLY | The cursor can move in the forward direction only. |
| ResultSet.TYPE\_SCROLL\_INSENSITIVE | The cursor can move in both the direction (forward and backward). The ResultSet is not sensitive to the changes made by the others to the database. |
| ResultSet.TYPE\_SCROLL\_SENSITIVE | The cursor can move in both the direction. The ResultSet is sensitive to the changes made by the others to the database. |

### 12) What are the differences between ResultSet and RowSet?

|  |  |
| --- | --- |
| ResultSet | RowSet |
| ResultSet cannot be serialized as it maintains the connection with the database. | RowSet is disconnected from the database and can be serialized. |
| ResultSet object is not a JavaBean object | ResultSet Object is a JavaBean object. |
| ResultSet is returned by the executeQuery() method of Statement Interface. | Rowset Interface extends ResultSet Interface and returned by calling the RowSetProvider.newFactory().createJdbcRowSet() method. |
| ResultSet object is non-scrollable and non-updatable by default. | RowSet object is scrollable and updatable by default. |

### 13) How can we execute stored procedures using CallableStatement?

Following are the steps to create and execute stored procedures. Here, we are creating a table user420 by using a stored procedure and inserting values into it.

* **Create the procedure in the database.**

To call the stored procedure, you need to create it in the database. Here, we are assuming that the stored procedure looks like this.

* 1. create or replace procedure "INSERTR"
  2. (id IN NUMBER,
  3. name IN VARCHAR2)
  4. is
  5. begin
  6. insert into user420 values(id,name);
  7. end;
  8. /

The table structure is given below:

* 1. create table user420(id number(10), name varchar2(200));
* **Establish a network connection.**
  1. Class.forName("oracle.jdbc.driver.OracleDriver");
  2. Connection con=DriverManager.getConnection(
  3. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
* **Create the Object of CallableStatement.**
  1. CallableStatement stmt=con.prepareCall("{call insertR(?,?)}");
* **Provide the values and execute the query by using the following syntax.**
  1. stmt.setInt(1,1011);
  2. stmt.setString(2,"Amit");
  3. stmt.execute();
* **Check the database; the values will be found there. However, the complete code will look like the following.**
  1. **import** java.sql.\*;
  2. **public** **class** Proc {
  3. **public** **static** **void** main(String[] args) **throws** Exception{
  5. Class.forName("oracle.jdbc.driver.OracleDriver");
  6. Connection con=DriverManager.getConnection(
  7. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
  9. CallableStatement stmt=con.prepareCall("{call insertR(?,?)}");
  10. stmt.setInt(1,1011);
  11. stmt.setString(2,"Amit");
  12. stmt.execute();
  14. System.out.println("success");
  15. }
  16. }

### 14) What is the role of the JDBC DriverManager class?

The DriverManager class acts as an interface between user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. The DriverManager class maintains a list of Driver classes that have registered themselves by calling the method DriverManager.registerDriver().

[More details.](https://www.javatpoint.com/DriverManager-class)

### 15) What are the functions of the JDBC Connection interface?

The **Connection interface** maintains a session with the database. It can be used for transaction management. It provides factory methods that return the instance of Statement, PreparedStatement, CallableStatement, and DatabaseMetaData.

[More details.](https://www.javatpoint.com/Connection-interface)

### 16) What does the JDBC ResultSet interface?

The ResultSet object represents a row of a table. It can be used to change the cursor pointer and get the information from the database. By default, ResultSet object can move in the forward direction only and is not updatable. However, we can make this object to move the forward and backward direction by passing either TYPE\_SCROLL\_INSENSITIVE or TYPE\_SCROLL\_SENSITIVE in createStatement(int, int) method.

[More details.](https://www.javatpoint.com/ResultSet-interface)

### 17) What does the JDBC ResultSetMetaData interface?

The ResultSetMetaData interface returns the information of table such as the total number of columns, column name, column type, etc.

[More details.](https://www.javatpoint.com/ResultSetMetaData-interface)

### 18) What does the JDBC DatabaseMetaData interface?

The DatabaseMetaData interface returns the information of the database such as username, driver name, driver version, number of tables, number of views, etc. Consider the following example.

1. **import** java.sql.\*;
2. **class** Dbmd{
3. **public** **static** **void** main(String args[]){
4. **try**{
5. Class.forName("oracle.jdbc.driver.OracleDriver");
7. Connection con=DriverManager.getConnection(
8. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
9. DatabaseMetaData dbmd=con.getMetaData();
11. System.out.println("Driver Name: "+dbmd.getDriverName());
12. System.out.println("Driver Version: "+dbmd.getDriverVersion());
13. System.out.println("UserName: "+dbmd.getUserName());
14. System.out.println("Database Product Name: "+dbmd.getDatabaseProductName());
15. System.out.println("Database Product Version: "+dbmd.getDatabaseProductVersion());
17. con.close();
18. }**catch**(Exception e){ System.out.println(e);}
19. }
20. }

**Output**

Driver Name: Oracle JDBC Driver

Driver Version: 10.2.0.1.0XE

Database Product Name: Oracle

Database Product Version: Oracle Database 10g Express Edition Release 10.2.0.1.0 -Production

[More details.](https://www.javatpoint.com/DatabaseMetaData-interface)

### 19) Which interface is responsible for transaction management in JDBC?

The **Connection interface** provides methods for transaction management such as commit(), rollback() etc.

[More details.](https://www.javatpoint.com/transaction-management-in-jdbc)

### 20) What is batch processing and how to perform batch processing in JDBC?

By using the batch processing technique in JDBC, we can execute multiple queries. It makes the performance fast. The java.sql.Statement and java.sql.PreparedStatement interfaces provide methods for batch processing. The batch processing in JDBC requires the following steps.

* Load the driver class
* Create Connection
* Create Statement
* Add query in the batch
* Execute the Batch
* Close Connection

Consider the following example to perform batch processing using the Statement interface.

1. **import** java.sql.\*;
2. **class** FetchRecords{
3. **public** **static** **void** main(String args[])**throws** Exception{
4. Class.forName("oracle.jdbc.driver.OracleDriver");
5. Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
6. con.setAutoCommit(**false**);
8. Statement stmt=con.createStatement();
9. stmt.addBatch("insert into user420 values(190,'abhi',40000)");
10. stmt.addBatch("insert into user420 values(191,'umesh',50000)");
12. stmt.executeBatch();//executing the batch
14. con.commit();
15. con.close();
16. }}

[More details.](https://www.javatpoint.com/batch-processing-in-jdbc)

### 21) What are CLOB and BLOB data types in JDBC?

**BLOB:** Blob can be defined as the variable-length, binary large object which is used to hold the group of Binary data such as voice, images, and mixed media. It can hold up to 2GB data on MySQL database and 128 GB on Oracle database. BLOB is supported by many databases such as MySQL, Oracle, and DB2 to store the binary data (images, video, audio, and mixed media).

**CLOB:** Clob can be defined as the variable-length, character-large object which is used to hold the character-based data such as files in many databases. It can hold up to 2 GB on MySQL database, and 128 GB on Oracle Database. A CLOB is considered as a character string.

### 22) What are the different types of lockings in JDBC?

A lock is a certain type of software mechanism by using which, we can restrict other users from using the data resource. There are four type of locks given in JDBC that are described below.

* **Row and Key Locks:** These type of locks are used when we update the rows.
* **Page Locks:** These type of locks are applied to a page. They are used in the case, where a transaction remains in the process and is being updated, deleting, or inserting some data in a row of the table. The database server locks the entire page that contains the row. The page lock can be applied once by the database server.
* **Table locks:** Table locks are applied to the table. It can be applied in two ways, i.e., shared and exclusive. Shared lock lets the other transactions to read the table but not update it. However, The exclusive lock prevents others from reading and writing the table.
* **Database locks:** The Database lock is used to prevent the read and update access from other transactions when the database is open.

### 23) How can we store and retrieve images from the database?

By using the PreparedStatement interface, we can store and retrieve images. Create a table which contains two columns namely NAME and PHOTO.

1. CREATE TABLE  "IMGTABLE"
2. (    "NAME" VARCHAR2(4000),
3. "PHOTO" BLOB
4. )

Consider the following example to store the image in the database.

1. **import** java.sql.\*;
2. **import** java.io.\*;
3. **public** **class** InsertImage {
4. **public** **static** **void** main(String[] args) {
5. **try**{
6. Class.forName("oracle.jdbc.driver.OracleDriver");
7. Connection con=DriverManager.getConnection(
8. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
10. PreparedStatement ps=con.prepareStatement("insert into imgtable values(?,?)");
11. ps.setString(1,"sonoo");
13. FileInputStream fin=**new** FileInputStream("d:\\g.jpg");
14. ps.setBinaryStream(2,fin,fin.available());
15. **int** i=ps.executeUpdate();
16. System.out.println(i+" records affected");
18. con.close();
19. }**catch** (Exception e) {e.printStackTrace();}
20. }
21. }

Consider the following example to retrieve the image from the table.

1. **import** java.sql.\*;
2. **import** java.io.\*;
3. **public** **class** RetrieveImage {
4. **public** **static** **void** main(String[] args) {
5. **try**{
6. Class.forName("oracle.jdbc.driver.OracleDriver");
7. Connection con=DriverManager.getConnection(
8. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
10. PreparedStatement ps=con.prepareStatement("select \* from imgtable");
11. ResultSet rs=ps.executeQuery();
12. **if**(rs.next()){//now on 1st row
14. Blob b=rs.getBlob(2);//2 means 2nd column data
15. **byte** barr[]=b.getBytes(1,(**int**)b.length());//1 means first image
17. FileOutputStream fout=**new** FileOutputStream("d:\\sonoo.jpg");
18. fout.write(barr);
20. fout.close();
21. }//end of if
22. System.out.println("ok");
24. con.close();
25. }**catch** (Exception e) {e.printStackTrace();  }
26. }
27. }

[More details.](https://www.javatpoint.com/storing-image-in-oracle-database)

### 24) How can we store the file in the Oracle database?

The setCharacterStream() method of PreparedStatement interface is used to set character information into the parameterIndex. For storing the file into the database, CLOB (Character Large Object) datatype is used in the table. For example:

1. CREATE TABLE  "FILETABLE"
2. (    "ID" NUMBER,
3. "NAME" CLOB
4. )

**Java Code**

1. **import** java.io.\*;
2. **import** java.sql.\*;
4. **public** **class** StoreFile {
5. **public** **static** **void** main(String[] args) {
6. **try**{
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8. Connection con=DriverManager.getConnection(
9. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
11. PreparedStatement ps=con.prepareStatement(
12. "insert into filetable values(?,?)");
14. File f=**new** File("d:\\myfile.txt");
15. FileReader fr=**new** FileReader(f);
17. ps.setInt(1,101);
18. ps.setCharacterStream(2,fr,(**int**)f.length());
19. **int** i=ps.executeUpdate();
20. System.out.println(i+" records affected");
22. con.close();
24. }**catch** (Exception e) {e.printStackTrace();}
25. }
26. }

### 25) How can we retrieve the file in the Oracle database?

The getClob() method of PreparedStatement is used to get file information from the database. Let's see the table structure of the example to retrieve the file.

1. CREATE TABLE  "FILETABLE"
2. (    "ID" NUMBER,
3. "NAME" CLOB
4. )

The example to retrieve the file from the Oracle database is given below.

1. **import** java.io.\*;
2. **import** java.sql.\*;
4. **public** **class** RetrieveFile {
5. **public** **static** **void** main(String[] args) {
6. **try**{
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8. Connection con=DriverManager.getConnection(
9. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
11. PreparedStatement ps=con.prepareStatement("select \* from filetable");
12. ResultSet rs=ps.executeQuery();
13. rs.next();//now on 1st row
15. Clob c=rs.getClob(2);
16. Reader r=c.getCharacterStream();
18. FileWriter fw=**new** FileWriter("d:\\retrivefile.txt");
20. **int** i;
21. **while**((i=r.read())!=-1)
22. fw.write((**char**)i);
24. fw.close();
25. con.close();
27. System.out.println("success");
28. }**catch** (Exception e) {e.printStackTrace();  }
29. }
30. }

### 26) What are the differences between stored procedure and functions?

The differences between stored procedures and functions are given below:

|  |  |
| --- | --- |
| **Stored Procedure** | **Function** |
| Is used to perform business logic. | Is used to perform the calculation. |
| Must not have the return type. | Must have the return type. |
| May return 0 or more values. | May return only one value. |
| The procedure supports input and output parameters. | The function supports only input parameter. |
| Exception handling using try/catch block can be used in stored procedures. | Exception handling using try/catch can't be used in user-defined functions. |

### 27) How can we maintain the integrity of a database by using JDBC?

To maintain the integrity of a database, we need to ensure the ACID properties. ACID properties mean Atomicity, Consistency, Isolation, and durability. In JDBC, Connection interface provides methods like setAutoCommit(), commit(), and rollback() which can be used to manage transaction. Let's see an example of transaction management in JDBC.

1. **import** java.sql.\*;
2. **class** FetchRecords{
3. **public** **static** **void** main(String args[])**throws** Exception{
4. Class.forName("oracle.jdbc.driver.OracleDriver");
5. Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
6. con.setAutoCommit(**false**);
8. Statement stmt=con.createStatement();
9. stmt.executeUpdate("insert into user420 values(190,'abhi',40000)");
10. stmt.executeUpdate("insert into user420 values(191,'umesh',50000)");
12. con.commit();
13. con.close();
14. }}

### 28) What is the JDBC Rowset?

JDBC Rowset is the wrapper of ResultSet. It holds tabular data like ResultSet, but it is easy and flexible to use. The implementation classes of RowSet interface are as follows:

* JdbcRowSet
* CachedRowSet
* WebRowSet
* JoinRowSet
* FilteredRowSet

### 29) What is the major difference between java.util.Date and java.sql.Date data type?

The major difference between java.util.Date and java.sql.Date is that, java.sql.Date represents date without time information whereas, java.util.Date represents both date and time information.

### 30) What does JDBC setMaxRows method do?

The setMaxRows(int i) method limits the number of rows the database can return by using the query. This can also be done within the query as we can use the limit cause in MySQL.

## **Java Servlet Interview Questions And Answers**

#### 1. What is a servlet?

The servlet is a Java interface whose implementation extends the functionality of the server. The servlet interacts with clients through a request-response principle.

Although servlets can serve any requests, they are usually used to extend web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes. The javax.servlet and javax.servlet.http packages provide interfaces and classes for creating servlets.

#### 2. What is the structure of a web project?

|  |  |
| --- | --- |
| src / main / java | Application / Library sources |
| src / main / resources | Application / Library resources |
| src / main / filters | Resource filter files |
| src / main / webapp | Web application sources |
| src / test / java | Test sources |
| src / test / resources | Test resources |
| src / test / filters | Test resource filter files |
| src / it | Integration Tests (primarily for plugins) |
| src / assembly | Assembly descriptors |
| src / site | Site |
| LICENSE.txt | Project’s license |
| NOTICE.txt | Project depends on |
| README.txt | Project’s readme |

##### **https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html**

#### 3. What is a servlet container?

**A servlet container**  is a program that represents a server that is engaged in system support for servlets and ensures their life cycle in accordance with the rules defined in the specifications.

Known implementations: Apache Tomcat, Jetty, JBoss, GlassFish, IBM WebSphere, Oracle Weblogic.

#### 4. What are the tasks, functionality of a servlet container?

A servlet container can work as a fully-fledged stand-alone web server, be a provider of pages for another web server, such as Apache, or integrate into a Java EE application server. It provides data exchange between the servlet and clients, undertakes such functions as creating a software environment for a functioning servlet, identifying and authorizing clients, and organizing a session for each of them.

#### 5. What do you know about servlet filters?

The servlet filter, according to the specification, is a reusable Java code that allows you to translate the content of HTTP requests, HTTP responses, and information contained in HTTP headers. A servlet filter is involved in preprocessing a request before it enters a servlet, and / or in the subsequent processing of a response coming from a servlet. Servlet filters can:

– intercept the servlet initialization before the servlet is initiated;

– determine the content of the request before the servlet is initiated;

– modify the headers and request data into which the incoming request is packaged;

– modify the headers and response data in which the resulting response is packaged;

– intercept the servlet initialization after accessing the servlet.

A servlet filter can be configured to work with a single servlet or group of servlets. The basis for the formation of filters is the javax.servlet.Filterinterface , which implements three methods:

void init (FilterConfig config) throws ServletException;

void destroy ();

void doFilter (ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException;

The init ()  method is called before the filter starts working and configures the filter configuration object. The doFilter method performs the filter directly. Thus, the server calls init () once to start the filter into operation, and then calls doFilter () as many times as many requests are made directly to this filter. After the filter finishes its work, the destroy () method is called*.*

#### 6. Why are servlet listeners needed?

Context and session listeners are classes that can keep track of when the context or session has been initialized, or track the time when they should be destroyed, and when attributes have been added or removed from the context or session.

Servlet 2.4 extends the listeners model of the query, allowing you to track how the query is created and destroyed, and how attributes are added and removed from the servlet. The following classes are added to Servlet 2.4:

* + ServletRequestListener
  + ServletRequestEvent
  + ServletRequestAttributeListener
* ServletRequestAttributeEvent

#### 7. When will you use filters, and when will listeners?

##### **Servlet Listener Example: https://www.journaldev.com/1945/servlet-listener-example-servletcontextlistener-httpsessionlistener-and-servletrequestlistener**

#### 8. How to handle exceptions thrown by another servlet in an application?

Since the browser only understands HTTP, then when the application throws an exception, the servlet container will handle the exception and create an HTTP response.

This is similar to what happens with error codes like 404, 403, etc. The Servlet API provides native servlet support for handling exceptions and errors that we can specify in the deployment descriptor.

The main task of such servlets is to handle the error or exception and send a clear HTTP response to the user. For example, you can provide a link to the main page, as well as a description of some details about the error.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <error-page>         <error-code>404</error-code>         <location>/AppExceptionHandler</location>     </error-page>     <error-page>         <exception-type>javax.servlet.ServletException</exception-type>         <location>/AppExceptionHandler</location>     </error-page> |

#### 9. What is a deployment descriptor?

A deployment descriptor is an artifact configuration file that will be deployed in a servlet container. In the Java Platform, Enterprise Edition specification, the deployment descriptor describes how a component, module or application (such as a web application or an enterprise application) should be deployed.

This configuration file specifies deployment options for a module or application with specific settings, security settings, and describes specific configuration requirements. The syntax for the deployment descriptor files is XML.

#### 10. How to implement the launch of the servlet with the launch of the application?

A servlet container usually loads a servlet when the client first requests it, but sometimes it is necessary to load the servlet right at the start of the application (for example, if the servlet is large and will take a long time to load). To do this, you must use the load-on-startup element in the descriptor (or the loadOnStartup annotation ), which will indicate the need to load the servlet at startup.

|  |  |
| --- | --- |
| 1  2  3  4  5 | <servlet>     <servlet-name>foo</servlet-name>     <servlet-class>com.foo.servlets.Foo</servlet-class>     <load-on-startup>5</load-on-startup>  </servlet> |

The value must be an int . If the value is negative, the servlet will be loaded when the client requests, and if 0 and further, it will be loaded at the start of the application. The smaller the number, the earlier the servlet will be in the download queue.

### Servlets in Java Interview Questions Answers

#### 11. What is the ServletConfig object?

The javax.servlet.ServletConfig interface is used to transfer configuration information to the servlet. Each servlet has its own ServletConfig object, the creation of which is the responsibility of the servlet container. To set configuration parameters, init parameters are used in the web.xml (or WebInitParam annotations). To get the ServletConfig object of this servlet, use the getServletConfig () method .

#### 12. What is the ServletContext object?

The javax.servlet.ServletContext interface provides access to the web application parameters to the servlet. The ServletContext object is unique and accessible to all servlets of the web application. We can use the ServletContext object when we need to provide access to one or more servlets to the initialized parameters of a web application. To do this, use the <context-param> element in web.xml . You can get the ServletContext object using the getServletContext () method from the ServletConfig interface .

Servlet containers can also provide context objects unique to a servlet group. Each group will be associated with its own set of host path URLs.

The ServletContext has been expanded into the Servlet 3 specification and provides programmatic addition of listeners and filters to the application. Also, this interface has many useful methods like getMimeType () , getResourceAsStream () , etc.

#### 13. What are the differences between ServletContext and ServletConfig?

* + ServletConfig is a unique object for each servlet, while a ServletContext is unique for the entire application.
  + ServletConfig is  used to provide initialization parameters to the servlet, and ServletContext to provide application initialization parameters for all servlets.
* We do not have the ability to set attributes in a ServletConfig object , while it is possible to set attributes in a ServletContext object that will be available to other servlets.

#### 14. What is Request Dispatcher?

The RequestDispatcher interface is used to send a request to another resource (it can be HTML, JSP, or another servlet in the same application). We can use this to add content from another resource to the response. This interface is used for internal communication between servlets in the same context. The interface implements two methods:

void forward (ServletRequest var1, ServletResponse var2) – sends a request from a servlet to another resource (servlet, JSP or HTML file) on the server.

void include (ServletRequest var1, ServletResponse var2) – includes the content of the resource (servlet, JSP or HTML page) in the response.

The interface can be accessed using the ServletContext method getRequestDispatcher (String s). The path must begin with / , which will be interpreted relative to the current root path of the context.

#### 15. How can I create a deadlock in a servlet?

Dedlock can be obtained by implementing a looped method call, for example, by calling the doPost () method in the doGet () method and calling doGet () in the doPost () method .

#### 16. How to get the servlet address on the server?

To get the actual servlet path on the server, you can use this construct: getServletContext (). GetRealPath (request.getServletPath ())

#### 17. How to get server information from servlet?

Server information can be obtained using the ServletContext object using the getServerInfo () method . Those. getServletContext (). getServerInfo () .

#### 18. How to get the client’s ip address on the server?

Use request.getRemoteAddr () to get the client’s ip in the servlet.

#### 19. What do you know about servlet wrappers?

The Servlet HTTP API provides two wrapper classes –  HttpServletRequestWrapper and  HttpServletResponseWrapper . They help developers implement their own implementation of the servlet’s request and response types . We can extend these classes and override only the necessary methods to implement our own types of response and query objects. These classes are not used in standard servlet programming.

#### 20. What is the life cycle of a servlet and when are some methods called?

A servlet container manages four phases of a servlet’s life cycle:

* + **Servlet class loading** – when a container receives a request for a servlet, the servlet class is loaded into memory and a constructor without parameters is called.
  + **Initialization of the servlet class** – after the class is loaded, the container initializes the ServletConfig object  for this servlet and injects it through the init () method. This is where the servlet class is converted from a regular class into a servlet.
  + **Request Processing** – After initialization, the servlet is ready to process requests. For each client request, the servlet container spawns a new thread (stream) and calls the  service () method by passing a reference to the response and request object.
* **Deletion from Service** — when a container stops or stops an application, the servlet container destroys the servlet classes by calling the destroy () method.

It can be described as a sequence of calling methods: init () , service () , destroy () .

* + public void init (ServletConfig config) – used by the container to initialize the servlet. Called once per servlet lifetime.
  + public void service (ServletRequest request, ServletResponse response) – called for each request. The method cannot be called before the init () method is executed .
* public void destroy () – called to destroy a servlet (once during the life of the servlet).

### Advanced Java Servlet Interview Questions and Answers

#### 21. What methods need to be determined when creating servlets?

To create a servlet, you must describe the servlet with:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <servlet-mapping>   <servlet-name>MyOwnDefaultServlet</servlet-name>   <url-pattern>/myservlet/\*</url-pattern>  </servlet-mapping>  <servlet>   <servlet-name>HelloWorld2</servlet-name>   <servlet-class>examples.servlets.HelloWorld2</servlet-class>    <init-param>     <param-name>greeting</param-name>     <param-value>Welcome</param-value>   </init-param>    <init-param>     <param-name>person</param-name>     <param-value>WebLogic Developer</param-value>   </init-param>  </servlet> |

Then extend the servlet class from the HttpServlet class .

Implement the service () or doGet () , doPost () method (or the first or second).

#### 22. In which case will you override the service () method?

The service () method is redefined when we want the servlet to handle both GET and POST requests in the same method. When the servlet container receives a client request, the service () method is called , which in turn calls the doGet () , doPost () methods based on the HTTP request method. It is believed that the service () method does not redefine much, except for the case of using one method for two types of requests specified at the beginning.

#### 23. Does it make sense to define a constructor for the servlet, how best to initialize the data?

This possibility is, but is considered meaningless. It is better to initialize the data by overriding the init () method , in which you can access the initialization parameters of the servlet through the use of the ServletConfig object .

#### 24. What are the differences between GenericServlet and HttpServlet?

Abstract class GenericServlet is a protocol -independent implementation of the Servlet interface . HttpServlet , as the name implies , is a servlet interface implementation for the HTTP protocol. It should be noted that HttpServlet extends GenericServlet.

#### 25. How to call another servlet from the same servlet of the same application?

If it is necessary to invoke a servlet from the same application, then it is necessary to use the internal servlet communication mechanism (inter-servlet communication mechanisms).

We can call another servlet using the RequestDispatcher forward () and include () methods to access additional attributes in the request for use in another servlet. The forward () method is used to transfer the processing request to another servlet. The include () method is used if we want to attach the result of another servlet to the returned response.

If it is necessary to call a servlet from another application, then RequestDispatcher will not work (determined for the application). Therefore, you can use the ServletResponse sendRedirect () method and provide the full URL from another servlet. To transfer data, you can use cookies as part of the servlet’s response, and then use them in our servlet.

#### 26. What do you know and what are the differences between the methods forward () and sendRedirect ()?

* + RequestDispatcher forward () is used to send the same request to another resource, while ServletResponse sendRedirect () is a two-step method. In the second method, the web application returns a response to the client with status code 302 (redirect) with a link to send a request. The request sends a completely new request.
  + forward () is processed inside the container, and sendRedirect () is processed by the browser.
  + It is necessary to use forward () to organize access within the same application, since it is faster than sendRedirect () , which requires additional networking.
  + In the forward () method, the browser does not know about the resource being actually processed and the URL in the string remains the same. In the sendRedirect () method, the URL is changed to the forwarding resource.
* In the forward () method, you cannot use to embed a servlet into another context. For this you can use only sendRedirect () .

#### 27. Should I worry about “multithreaded security” when working with servlets?

The methods of the class HTTPServlet init () and destroy () are called once during the life cycle of the servlet – so there’s no need to worry about them. The doGet () , doPost () methods are called for each client request and since servlets use multithreading, then you need to think about thread-safe work.

If there are local variables in these methods, there is no need to think about multi-threaded security, since they will be created separately for each thread. But if global resources are used, then you need to use synchronization as in any multithreaded Java application.

#### 28. What is the difference between a web server and an application server?

A web server is required to process an HTTP request from a client browser and respond to a client using an HTTP response. The web server understands the HTTP language and runs over the HTTP protocol. An example of a web server is an implementation from Apache – Tomcat.

The application server provides additional features, such as support for JavaBeans, JMS Messaging, Transaction Management, etc. You can say that the application server is a web server with additional features that help develop enterprise applications.

#### 29. Which HTTP method is not immutable?

An HTTP method is called immutable if it always returns the same result. HTTP methods GET, PUT, DELETE, HEAD, OPTIONS are immutable. You must implement the application so that these methods return the same result. Variable methods include the HTTP POST method. Post method is used to implement something that changes with each request.

For example, to access an HTML page or image, you must use the GET method, since it returns the same result. But if we need to save the order information in the database, then we need to use the POST method. Immutable methods are also known as safe methods and there is no need to worry about repeated requests from the client for these methods.

#### 30. Why is the HttpServlet class declared as abstract?

The HTTPServlet class provides the HTTP implementation of the servlet protocol (and therefore helps avoid constant coding of duplicate information), but it does not have the implemented methods doGet () and doPost () (they have HTTP 405 Method Not Implemented error by default), and therefore declared abstract. The implementation of these methods is passed on to the developer.

### Java Servlet Interview Questions And Answers For Experienced

#### 31. What is the difference between GET and POST methods?

* + The GET method is immutable, while POST is mutable.
  + Using the GET method, you can send a limited number of data that will be sent in the URL header. In the case of the POST method, we can send large amounts of data, because they will be in the body of the method.
  + GET method data is transmitted in clear text, which can be used for malicious purposes. POST data is transmitted in the request body and hidden from the user.
  + The GET method is the default HTTP method, and the POST method must be specified explicitly to send a request.
* GET method is used by hyperlinks on the page.

#### 32. What is the MIME type?

MIME (pronounced “maym”, English. Multipurpose Internet Mail Extensions – multi-purpose Internet mail extensions) – a standard that describes the transmission of various types of data via e-mail, as well as, in general, the specification for encoding information and formatting messages in such a way so that they can be sent over the Internet. The Content-Type response header is the MIME type.

The server sends the MIME type to the client so that it understands what type of data is being sent. This helps to display the data correctly on the client. The most commonly used MIME types are:  text / html , text / xml , application / xml, and many others.

In the ServletContext, there is a getMimeType () method to get the correct MIME file type and then use this information to specify the type of content in the response.

#### 33. What are the advantages of Servlet over CGI?

Servlet technology was created to overcome the weaknesses of the Common Gateway Interface (common gateway interface). The following advantages of servlets over CGI can be highlighted:

* + Servlets provide better performance in terms of processing requests, better memory usage by using the advantage of multithreading (a new thread is created for each request, which is faster than allocating memory for a new object for each request, as in CGI).
  + Servlets, platform and system are independent. Thus, a web application written using servlets can be run in any servlet container that implements the standard and in any operating system.
  + Servlet usage improves program reliability, because the servlet container itself takes care of the servlet life cycle (and therefore memory leaks), security, and garbage collection.
* Servlets are relatively easy to learn and maintain, so the developer only needs to care about the business logic of the application, and not the internal implementation of web technologies.

#### 34. What are the most common tasks performed in a servlet container?

* + **Support data exchange** . A servlet container provides an easy way to exchange data between a web client (browser) and a servlet. Thanks to the container, there is no need to create a socket listener on the server to track requests from the client, as well as parse the request and generate a response. All these important and complex tasks are solved using the container and the developer can focus on the business logic of the application.
  + **Servlet and resource lifecycle management** . Starting from loading the servlet into memory, initializing, implementing methods and ending with the destruction of the servlet. The container also provides additional utilities, such as JNDI, for managing a pool of resources.
  + **Multithreading support** . The container independently creates a new thread for each request and provides it with a request and response for processing. Thus, the servlet is not reinitialized for each request and thus saves memory and reduces the time to process the request.
  + **JSP support** . JSP classes are not similar to standard Java classes, but the servlet container converts each JSP to a servlet and is then managed by the container as a regular servlet.
* **Various tasks** . The servlet container manages the resource pool, application memory, and garbage collector. Provides security settings and more.

#### 35. What is the difference between PrintWriter and ServletOutputStream?

PrintWriter is a class for working with a character stream, and  ServletOutputStream is a class for working as a byte stream. PrintWriter is used to write character-based information, such as an array of characters or a string in the response, while ServletOutputStream is used to write a byte array to the response.

To get an instance of  ServletOutputStream, use the ServletResponse getOutputStream () method , and for PrintWriter , use the ServletResponse getWriter () method  .

#### 36. Can we get the PrintWriter and ServletOutputStream simultaneously in a servlet?

We cannot create two objects of these classes in one servlet. If we try to embed both getWriter () and getOutputStream () methods in the response, we will get a java.lang.IllegalStateException exception  with the message that a different method has already been called for this answer.

#### 37. Tell us about the interface SingleThreadModel.

The interface was created to guarantee thread safety and the impossibility of creating two threads in the servlet method service. However, the SingleThreadModel interface  does not solve all thread safety issues.

For example, session attributes or static variables can be accessed by different requests in different threads at the same time (even when using this interface). In general, it killed all the multithreading profit and the interface was declared deprecated since Servlet 2.4.

#### 38. What are the attributes of servlets and what is their scope?

Servlet attributes are used for internal servlet communication. We can use the attributes set, get, remove in a web application. There are three attribute scopes – request scope , session scope , application scope .

The ServletRequest , HttpSession, and  ServletContext interfaces  provide methods for the get () , set () , remove () attributes from the request scope, session scope, application scope, respectively.

#### 39. Why is it necessary to override only the init () method with no arguments?

If we need to initialize some resources before the servlet starts processing requests, then we need to override the init () method . If you override the init method (ServletConfig config ), the super (config) method must be called first , which will call the init method (ServletConfig config) of the superclass.

That is why GenericServlet provides another init () method without parameters, which will be called at the end of the init method (ServletConfig config) . The developer should use the override init () method without parameters to initialize the variables to avoid any problems, for example without specifying a callsuper () in the overridden init method (ServletConfig config) .

#### 40. What does URL encoding mean? Why do we need methods encode () and decode ()?

URL Encoding is the process of converting data into a Common Gateway Interface (CGI) form that will allow you to surf the net without problems. URL Encoding separates spaces and replaces special characters with escape characters.

For example, the java.net.URLEncoder.encode (String str, String unicode) method is used to encode a string   . The reverse operation of decoding is possible thanks to the  java.net.URLDecoder.decode method (String str, String unicode) .

### Top 51 Java Servlet Interview Questions And Answers

#### 41. Why are the methods encodeUrl () and encodeRedirectUrl () needed and how are they different?

HttpServletResponse provides methods for converting URLs to HTML hyperlinks with the conversion of special characters and spaces, as well as adding session id to the URL. This behavior is similar to URLEncoder encode () , but with the addition of the additional parameter jsessionid at the end of the URL.

The HttpServletResponse method encodeRedirectUrl () is used to convert the redirect URL in the response. Thus, when providing URL rewriting support for HTML hyperlinks, you must use encodeURL () , and for redirect URLs, use encodeRedirectUrl () .

#### 42. What are the different methods of managing a session in servlets?

A session is a normal state of interaction between the server and the client, and may contain many requests and responses from the client / server. Since HTTP and web server do not remember state (stateless), the only way to maintain the session is to send unique information (session id) in each request and response between the client and the server.

There are several common ways to manage servlet sessions:

* + User authentication
  + HTML hidden field
  + Cookies
  + URL Rewriting
* Session management API

#### 43. What does URL Rewriting mean?

We can use HTTPSession to manage a session in servlets , but it works with cookies, and they are sometimes disabled. For this case, the servlet provides URL Rewriting. From a programming point of view, only one action is needed – URL coding. Another advantage is that this method is like a backup and is turned on only when the cookies are turned off.

Using the HttpServletResponse encodeURL () method, we can encode the URL. If redirecting to another resource is necessary, then the encodeRedirectURL () method is used to provide information about the session  .

#### 44. How are cookies used in servlets?

Cookies (cookies) are used in client-server interaction and they are not specific to Java. Servlet API provides cookies support through the  javax.servlet.http.Cookie class implements Serializable, Cloneable . To get an array of cookies from the request, you must use the HttpServletRequest getCookies () method  . There are no methods for adding cookies to the request.

Similar to  HttpServletResponse addCookie (Cookie c) – can add cookies in the response header, but there is no getter for this type of data transfer.

#### 45. How to notify an object in a session that a session is invalid or has expired?

To be sure that the object has been notified about the session termination, the object must implement the javax.servlet.http.HttpSessionBindingListener interface  . Two methods of this interface: valueBound () and valueUnbound () are used to implement logic when adding an object as an attribute to a session and when a session is destroyed.

#### 46. ​​What is the effective way to make sure that all servlets are accessible only to a user with a valid session?

Servlet filters are used to intercept all requests between the servlet container and the servlet. Therefore, it is logical to use a filter to check the necessary information (for example, the validity of the session) in the request.

#### 47. How can we provide transport layer security for our web application?

For this, you must configure SSL for your servlet container. How to do this is described in the manuals for the specific implementation of the container.

#### 48. How to organize a connection to the database and provide log4j logging in the servlet?

When working with a large number of database connections, it is recommended to initialize them in the servlet context listener and set the context attribute to be used by other servlets. Log4j logging is connected using an XML configuration (or property file) and further this information is used when configuring the context listener.

#### 49. What important features exist in Servlet 3?

* + **Servlet Annotations** . Before Servlet 3, the entire mapping was in web.xml, which led to errors and was trivially inconvenient with a large number of servlets. Examples of annotations:   @WebServlet , @WebInitParam , @WebFilter , @WebListener .
  + **Web Fragments.**With the advent of web fragments, we can contain a lot of modules in a single-page web application. All modules are written in the fragment.xml in the META-INF directory. This allows you to split a web application into separate modules included as JAR files in a separate lib directory.
  + **Dynamic addition of web components** . Using the ServletContext object, we can programmatically add filters and listeners. This helps to build a dynamic system in which the required object will be called only when necessary. To do this, use the  addServlet () , addFilter () , addListener () methods .
* **Asynchronous execution** . Support for asynchronous processing allows you to transfer the execution of the request to another thread without keeping the entire server busy.

#### 50. What are the different ways to authenticate a servlet?

A servlet container provides various authentication methods:

* + HTTP Basic Authentication
  + HTTP Digest Authentication
  + HTTPS Authentication