JAVA DATABASE CONNECTIVITY

1. What is JDBC Driver?

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

* JDBC-ODBC bridge driver
* Native-API driver (partially java driver)
* Network Protocol driver (fully java driver)
* Thin driver (fully java driver)

1. What are the steps to connect to a database in java?

* Registering the driver class
* Creating connection
* Creating statement
* Executing queries
* Closing connection

1. What are the JDBC API components?

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

Interfaces:

* Connection
* Statement
* PreparedStatement
* ResultSet
* ResultSetMetaData
* DatabaseMetaData
* CallableStatement etc.

Classes:

* DriverManager
* Blob
* Clob
* Types
* SQLException etc.

1. What is the role of the JDBC DriverManager class?

The DriverManager class manages the registered drivers. It can be used to register and unregister drivers. It provides a factory method that returns the instance of Connection.

1. What is 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.

1. What is the purpose of 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.

1. What is the JDBC ResultSetMetaData interface?

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

1. What is 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.

1. What do you mean by batch processing in JDBC?

Batch processing helps you to group related SQL statements into a batch and execute them instead of executing a single query. By using the batch processing technique in JDBC, you can execute multiple queries which makes the performance faster.

1. What is the difference between execute, executeQuery, and executeUpdate?

Statement execute(String query) is used to execute an SQL query and it returns TRUE if the result is a ResultSet such as running Select queries. The output is FALSE when there is no ResultSet object such as running Insert or Update queries. We can use getResultSet() to get the ResultSet and getUpdateCount() method to retrieve the update count.

Statement executeQuery(String query) is used to execute Select queries and returns the ResultSet. ResultSet returned is never null even if there are no records matching the query. When executing select queries we should use the executeQuery method so that if someone tries to execan ute insert/update statement it will throw java.sql.SQLException with the message “executeQuery method can not be used for update”.

Statement executeUpdate(String query) is used to execute Insert/Update/Delete (DML) statements or DDL statements that returns nothing. The output is int and equals the row count for SQL Data Manipulation Language (DML) statements. For DDL statements, the output is 0.

You should use execute() method only when you are not sure about the type of statement else use the executeQuery or executeUpdate method.

1. What do you understand by JDBC Statements?

JDBC statements are the statements that are used to send SQL commands to the database and retrieve data back from the database. Various methods like execute(), executeUpdate(), executeQuery, etc. are provided by JDBC to interact with the database.

JDBC supports 3 types of statements:

Statement: Used for general-purpose access to the database and executes a static SQL query at runtime.

PreparedStatement: Used to provide input parameters to the query during execution.

CallableStatement: Used to access the database stored procedures and helps in accepting runtime parameters.

JAVA свързване на база данни

1. Какво представлява JDBC Driver?

JDBC Driver е софтуерен компонент, който дава възможност на java приложения да взаимодействат с базата данни. Има 4 вида JDBC драйвери:

        JDBC-ODBC мостов шофьор

        Native-API драйвер (частично драйвер за Java)

        Драйвер за мрежов протокол (напълно драйвер за Java)

        Тънък драйвер (напълно драйвер за Java)

2. Какви са стъпките за свързване към база данни в java?

        Регистриране на класа на водача

        Създаване на връзка

        Създаване на изявление

        Изпълнение на заявки

        Затваряне на връзката

3. Какви са компонентите на JDBC API?

Пакетът java.sql съдържа интерфейси и класове за JDBC API.

Интерфейси:

        Връзка

        Твърдение

        Подготвенастет

        ResultSet

        РезултатSetMetaДанни

        База данниМетаДанни

        CallableStatement т.н.

Класове:

        ШофьорМанаж

        Петно

        Клоб

        Видове

        SQLException и т.н.

4. Каква е ролята на класа JDBC DriverManager?

Клас driverManager управлява регистрираните драйвери. Тя може да се използва за регистриране и отрегистриране на драйвери. Той предоставя фабрично метод, който връща екземпляра на Връзка.

5. Какво представлява интерфейсът на JDBC Connection?

Интерфейсът връзка поддържа сесия с базата данни. Може да се използва за управление на транзакции. Той предоставя фабрични методи, които връщат екземпляра на Изявление, PreparedStatement, CallableStatement и DatabaseMetaData.

6. Каква е целта на интерфейса JDBC ResultSet?

Обектът ResultSet представлява ред от таблица. Може да се използва за промяна на показалец на курсора и получаване на информацията от базата данни.

7. Какво представлява интерфейсът JDBC ResultSetMetaData?

Интерфейсът ResultSetMetaData връща информацията на таблицата като общия брой колони, име на колона, тип колона и др.

8. Какво е JDBC база данниMetaData интерфейс?

Интерфейсът DatabaseMetaData връща информацията на базата данни като потребителско име, име на драйвер, версия на драйвера, брой таблици, брой изгледи и др.

9. Какво имате предвид под пакетна обработка в JDBC?

Пакетната обработка ви помага да групирате свързаните SQL отчети в папка и да ги изпълнявате, вместо да изпълнявате една заявка. С помощта на техниката за обработка на партиди в JDBC можете да изпълните няколко заявки, което прави производителността по-бърза.

10. Каква е разликата между изпълнение, изпълнениеQuery и executeUpdate?

Изявление изпълнение(Низа заявка) се използва за изпълнение на SQL заявка и тя връща TRUE, ако резултатът е ResultSet като изпълнение Избор на заявки. Изходът е FALSE, когато няма Обект ResultSet като изпълнение На Вмъкване или Актуализиране на заявки. Можем да използваме getResultSet(), за да получим метода ResultSet и getUpdateCount() за извличане на броя на актуализациите.

Изявление executeQuery(Низа заявка) се използва за изпълнение на Изберете заявки и връща ResultSet. ResultSet върнат никога не е null дори ако няма записи, отговарящи на заявката. При изпълнение на изберете заявки трябва да използваме метода executeQuery, така че ако някой се опита да execan ute вмъкване / актуализация изявление ще хвърли java.sql.SQLException със съобщението "executeQuery метод не може да се използва за актуализация".

Изявление executeUpdate(Низа заявка) се използва за изпълнение на Вмъкване / актуализиране / изтриване (DML) изявления или DDL изявления, която не връща нищо. Изходът е int и се равнява на броя редове за SQL данни манипулиране език (DML) изявления. За DDL изявления изходът е 0.

Трябва да използвате execute() метод само когато не сте сигурни за вида на изявление друго използвате executeQuery или изпълниUpdate метод.

11. Какво разбирате от JDBC Изявления?

JDBC изявления са изявленията, които се използват за изпращане на SQL команди в базата данни и извличане на данни обратно от базата данни. Различни методи като execute(), executeUpdate(), executeQuery и т.н. се предоставят от JDBC за взаимодействие с базата данни.

JDBC поддържа 3 вида изявления:

Изявление: Използва се за достъп с общо предназначение до базата данни и изпълнява статична SQL заявка по време на изпълнение.

PreparedStatement: Използва се за предоставяне на входни параметри на заявката по време на изпълнение.

CallableStatement: Използва се за достъп до базата данни съхранени процедури и помага при приемането на параметрите по време на изпълнение.