Day 37

DriverManager class

The DriverManager class is the component of JDBC API and also a member of the *java.sql* package. The DriverManager class acts as an interface between users 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 all the appropriate methods to register and deregister the database driver class and to create a connection between a Java application and the database. The DriverManager class maintains a list of Driver classes that have registered themselves by calling the method DriverManager.registerDriver(). Note that before interacting with a Database, it is a mandatory process to register the driver; otherwise, an exception is thrown.

Methods of the DriverManager Class

Method	Description
1) public static synchronized void registerDriver(Driver driver):	is used to register the given driver with DriverManager. No action is performed by the method when the given driver is already registered.
2) public static synchronized void deregisterDriver(Driver driver):	is used to deregister the given driver (drop the driver from the list) with DriverManager. If the given driver has been removed from the list, then no action is performed by the method.
3) public static Connection getConnection(String url) throws SQLException:	is used to establish the connection with the specified url. The SQLException is thrown when the corresponding Driver class of the given database is not registered with the DriverManager.
4) public static Connection getConnection(String url,String userName,String password) throws SQLException:	is used to establish the connection with the specified url, username, and password. The SQLException is thrown when the corresponding Driver class of the given database is not registered with the DriverManager.
5) public static Driver getDriver(String url)	Those drivers that understand the mentioned URL (present in the parameter of the method) are returned by this method provided those drivers are mentioned in the list of registered drivers.
6) pubic static int getLoginTimeout()	The duration of time a driver is allowed to wait in order to establish a connection with the database is returned by this method.
7) pubic static void setLoginTimeout(int sec)	The method provides the time in seconds. sec mentioned in the parameter is the maximum time that a driver is allowed to wait in order to establish a connection with the database. If 0 is passed in the parameter of this method, the driver will have to wait infinitely while trying to establish the connection with the database.
8) public static Connection getConnection(String URL, Properties prop) throws SQLException	A connection object is returned by this method after creating a connection to the database present at the mentioned URL, which is the first parameter of this method. The second parameter, which is "prop", fetches the authentication details of the database (username and password.). Similar to the other variation of the getConnection() method, this method also throws the SQLException, when the corresponding Driver class of the given database is not registered with the DriverManager.

Connection interface

A Connection is a session between a Java application and a database. It helps to establish a connection with the database.

The Connection interface is a factory of Statement, PreparedStatement, and DatabaseMetaData, i.e., an object of Connection can be used to get the object of Statement and DatabaseMetaData. The Connection interface provide many methods for transaction management like commit(), rollback(), setAutoCommit(), setTransactionIsolation(), etc.

By default, connection commits the changes after executing queries.

Commonly used methods of Connection interface:

1) public Statement createStatement(): creates a statement object that can be used to execute SQL queries.

2) public Statement createStatement(int resultSetType,int resultSetConcurrency): Creates a Statement object that will generate ResultSet objects with the given type and concurrency.

3) public void setAutoCommit(boolean status): is used to set the commit status. By default, it is true.

4) public void commit(): saves the changes made since the previous commit/rollback is permanent.

5) public void rollback(): Drops all changes made since the previous commit/rollback.

6) public void close(): closes the connection and Releases a JDBC resources immediately.

Connection Interface Fields

There are some common Connection interface constant fields that are present in the Connect interface. These fields specify the isolation level of a transaction.

TRANSACTION_NONE: No transaction is supported, and it is indicated by this constant.

TRANSACTION_READ_COMMITTED: It is a constant which shows that the dirty reads are not allowed. However, phantom reads and non-repeatable reads can occur.

TRANSACTION_READ_UNCOMMITTED: It is a constant which shows that dirty reads, non-repeatable reads, and phantom reads can occur.

TRANSACTION_REPEATABLE_READ: It is a constant which shows that the non-repeatable reads and dirty reads are not allowed. However, phantom reads and can occur.

TRANSACTION_SERIALIZABLE: It is a constant which shows that the non-repeatable reads, dirty reads as well as the phantom reads are not allowed.

Statement interface

The **Statement interface** provides methods to execute queries with the database. The statement interface is a factory of ResultSet i.e. it provides factory method to get the object of ResultSet.

Commonly used methods of Statement interface:

The important methods of Statement interface are as follows:

1) public ResultSet executeQuery(String sql): is used to execute SELECT query. It returns the object of ResultSet.

2) public int executeUpdate(String sql): is used to execute specified query, it may be create, drop, insert, update, delete etc.

3) public boolean execute(String sql): is used to execute queries that may return multiple results.

4) public int[] executeBatch(): is used to execute batch of commands.

```
import java.sql.*;
class FetchRecord{
public static void main(String args[])throws Exception{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
Statement stmt=con.createStatement();

//stmt.executeUpdate("insert into emp765 values(33,'Irfan',50000)");
//int result=stmt.executeUpdate("update emp765 set name='Vimal',salary=10000 where id=33");
int result=stmt.executeUpdate("delete from emp765 where id=33");
System.out.println(result+" records affected");
con.close();
}}
```