**Java DataBase Connectivity**

**Steps to connect Database with Java application :**

Import the Packages

1. Load the drivers using the *forName() method*
2. Register the drivers *using DriverManager*
3. Establish a connection*using the Connection class object*
4. Create a statement
5. Execute the query
6. Close the connections

The **forName()** static method of Class class is used to register the driver class. This method is used to dynamically load the driver class. if driver class not found it throws classnotfoundexception

Class.forName("com.mysql.jdbc.Driver");  //load the driver

The **getConnection()** method of DriverManager class is used to establish connection of java application with your database

Syntax :-

getConnection(  "jdbc:mysql://localhost:port\_num(3306)/Database\_name","user","password")

Connection con=DriverManager.getConnection(  "jdbc:mysql://localhost:3306/sonoo","root","root");  // Register the drivers *using DriverManager*

//here sonoo is database name, root is username and password

 The **createStatement()** method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Statement stmt=con.createStatement();  //create a statement

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.

ResultSet rs=stmt.executeQuery("select \* from emp");  //Execute that statement

The **close()** method of Connection interface is used to close the connection.

con.close() //close the connection

**JDBC Drivers :**

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

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

**2.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.

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. |

**3.Resultset Interface :** The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row.

**4.Prepared Interface :** The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized query.

Example String sql="insert into emp values(?,?,?)";

PreparedSattement ps=con. prepareStatement(sql);

Syntax= ps.setDataType(col\_index\_table,value);

ps.setInt(1,101)

ps.setString(2”Akshay”);

ps.setDouble(3,2345.6);

As you can see, we are passing parameter (?) for the values. Its value will be set by calling the setter methods of PreparedStatement.

**Spring JDBC :** Spring **JdbcTemplate** is a powerful mechanism to connect to the database and execute SQL queries. It internally uses JDBC api, but eliminates a lot of problems of JDBC API.

The problems of JDBC API are as follows:

* We need to write a lot of code before and after executing the query, such as creating connection, statement, closing resultset, connection etc.
* We need to perform exception handling code on the database logic.
* We need to handle transaction.
* Repetition of all these codes from one to another database logic is a time consuming task.

**Spring JDBC provide JdbcTemplate class which all important method to perform database related operation(create,delete,update,retrieve)**

**Dependencies required in pom.xml**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>5.3.14</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.14</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.27</version>

</dependency>

**JDBC with XML based configuration---jdbc-config.xml**

**Step 1:- import DriverManagerClass package of jdbc**

<bean class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"* name=*"ds"*>

**Step 2:- create bean of JdbcTemplate class and insert property of JdbcTemplate class as Drivermanager ref**

<bean class=*"org.springframework.jdbc.core.JdbcTemplate"* name=*"jdbcTemplate"*>

<property name=*"dataSource"*>

<ref bean=*"ds"*/>

</property>

</bean>

**Step3 : Use JdbcTemplate class reference in Dao implementation class**

<bean class=*"com.app.springJDBC.dao.StudentDaoImpl"* name=*"studentDao"*>

<property name=*"jdbcTemplate"*>

<ref bean=*"jdbcTemplate"*></ref>

</property>

</bean>

**JDBC without XML based configuration**

**ResultSet Extractor :** **ResultSetExtractor** is an interface that is used to fetch the records from the database. It’s a callback interface that is used by [JDBC](https://www.geeksforgeeks.org/introduction-to-jdbc/) Template’s query() method where we need to pass the instance of ResultSetExtractor in order to fetch the data.

public T query(String sqlQuery, ResultSetExtractor<T> resultSetExtractor)

In order to fetch the data using ResultSetExtractor, we need to implement the ResultSetExtractor interface and provide the definition for its method. It has only one method. i.e., extractData() which takes an instance of ResultSet as an argument and returns the list.

public T extractData(ResultSet resultSet) throws SQL Exception, DataAccessException

**RowMapper Interface :**

n Spring, the **RowMapper** interface is used to fetch the records from the database using the **query()** method of the **JdbcTemplate** class.

**Syntax for query() method of JdbcTemplate class:**

public T query(String sqlQuery, RowMapper<T> rowMapper)

RowMapper is a callback interface that is called for each row and maps the row of relations with the instances to the model(user-defined) class. Unlike ResultSetExtractor the RowMapper iterates the ResultSet internally and adds the extracted data into a collection, And we do not need to write the code for collections as we do in ResultSetExtractor. It has only one method mapRow() which takes two arguments ResultSet and rowNumber respectively.

public T mapRow(ResultSet resultSet, int rowNumber)throws SQLException