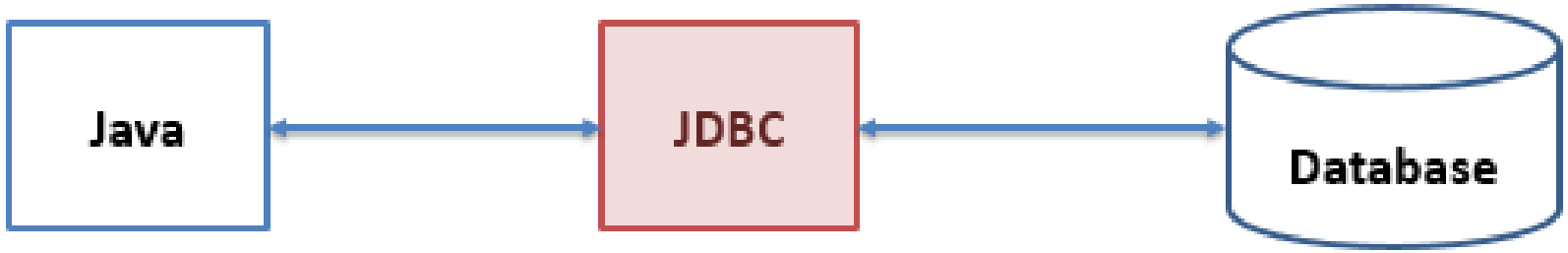
TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Java Data Base Connectivity

EXPERIENTIAL learning factory

Introduction



- Java Data Base Connectivity is an **API**, as the name implies, it helps to achieve the connectivity between Java Programs & Database
- If we have a Web Application & if it has a DB, then it needs to interact with DB to read / modify the data. JDBC helps to do this & in the world of Java, JDBC is the "One & Only API" that helps to interact ONLY with RDBMS (DB) Application
- Also JDBC is "DB Independent" i.e. using JDBC we can interact with any RDBMS Applications exist in the world (like Oracle, MySQL, DB2, MS-SQL, SyBase, etc.,)

NOTE:

1. MongoDB
2. Cassandra
3. Hadoop Distributed File System (HDFS/Hadoop)

They are Applications to store the data but they are not RDBMS Applications

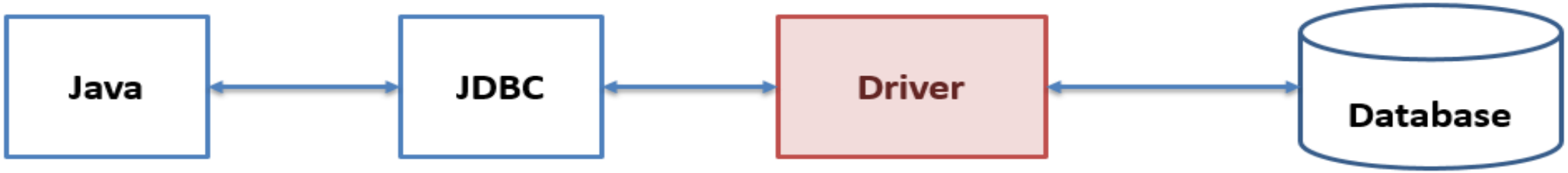
Steps to work with JDBC

1. Load the "Driver"
2. Get the "DB Connection" via "Driver"
3. Issue "SQL Queries" via "Connection"
4. "Process the Results" returned by "SQL Queries"
5. Close All "JDBC Objects"

First JDBC Program

- "java.sql.*" is the Package Representation of JDBC
- i.e. Any Class / Interface belongs to this package means, it’s part of JDBC

Drivers



1. Driver is an additional software component required by JDBC to interact with database

2. **Drivers are provided by DB Vendor & they are DB dependent**

3. "Driver Class" is a **Concrete Class**, present in driver JAR file, is the one that implements the **java.sql.Driver interface**. This interface is present in JDBC API & every JDBC driver provider has to implement this Interface

4. The Driver helps us to establish DB Connection, transfers the DB query and results between Java program and DB

Note: DB Vendor providers Driver Software in the form of a "JAR File"

Driver Class

- “Driver Class” is a Concrete Class, present in driver JAR file, is the one that implements the "java.sql.Driver" interface
- This interface is present in JDBC API & every JDBC driver provider has to implement this Interface
- By referring Driver Manual we can get the "Driver Class" information

There are 2 ways to load the Driver Class

1. By invoking "registerDriver()" method presentin "java.sql.DriverManager" Class by passing an instance of "Driver Class“
2. Load the Driver Class is with the help of Java's "Class.forName()" Method
   Class.forName("com.mysql.jdbc.Driver").newInstance();

➢ This is the most common approach to register a Driver Class which helps us to pass "Driver Class Name at Runtime"

➢ But if we create an instance of driver class using new operator, then driver class name can't be passed at Runtime

DB URL

- Data Base Uniform Resource Locator (DB URL), as the name implies, it uniquely identifies Database OR a RDBMS Application in the Network

- The structure of DB URL is **<Protocol>:<Subprotocol>:<Subname>**

- **Protocol:** It's a Mandatory Information & is "Case In-sensitive“. In case of Java, Protocol is always "jdbc"
- **Subprotocol:** It's a Mandatory Information & "Case In-sensitive". This information is provided by DB Vendor & we have to refer the Driver Manual to get this info. In case of MySQL, Subprotocol is "mysql" but, incase of Oracle or DB2 its different
- **Subname:** It's a Mandatory Information, It Consists of,

1. Host Name (Computer Name/IP Address and Case In-sensitive)
2. Port Number (should be Digits)
3. Database Name / Schema Name (Case In-sensitive)
4. User Name & Password (Optional and Case Sensitive)

NOTE: Arrangement of Subname varies from driver to driver, we have to refer the manual & arrange accordingly

DB URL Examples…

Oracle

jdbc:oracle:thin:myuser/mypassword@myserver:1521:mydb

Microsoft SQL Server

jdbc:microsoft:sqlserver://myserver:1433;DatabaseName=mydb;user=myuser;password=mypassword

MySQL

jdbc:mysql://myserver:3306/mydb?user=myuser&password=mypassword

DriverManager

- DriverManager is a "Concrete Class" present in JDBC API & as the name implies, it manages the drivers
- It helps Java Program to establish the connection to DB & for that it requires following information

1. Driver Class
2. DB URL

- By invoking "registerDriver()" method on DriverManager we can provide an "Object of Driver Class"
- By invoking "getConnection()" method on DriverManager we can provide "DB URL"

- DriverManager's getConnection() method helps us to establish the connection to the DB. This method

➢ throws "SQLException" if it is unable to establish the connection to DB

OR

➢ returns an object of "Connection" if it is able to establish the connection to DB

- "java.sql.Connection" is an interface & It's an "Object representation of Physical DB Connection" that is used by Java program to communicate with DB

DriverManager Continued…

- DriverManager consist of only one constructor which is "Private Default" in nature

- Hence it cannot be inherited or instantiated. So whatever the methods it exposes to outside world, they "should be public static" in nature

- DriverManager has overloaded versions of getConnection() methods. They are,

1. Connection getConnection(String dbUrl) throws SQLException
2. Connection getConnection(String dbUrl, String userNM, String password) throws SQLException
3. Connection getConnection (String url, Properties info) throws SQLException

Results of RDBMS Application

- Whenever we issue "Select SQL Queries" to DB it returns DB Results

- Whenever we issue "Other than Select SQL Queries" to DB then it returns "No. of Rows Affected Count" in the form of Integer

- Hence w.r.t results we can group SQL Queries into 2 Groups

1. Select SQL Query
2. Other Than Select SQL Query

Static vs Dynamic SQL Queries

Static SQL Queries

ANY SQL queries "without conditions" OR "ALL Conditions with hardcoded values" are called as "Static SQL Queries"

Example :

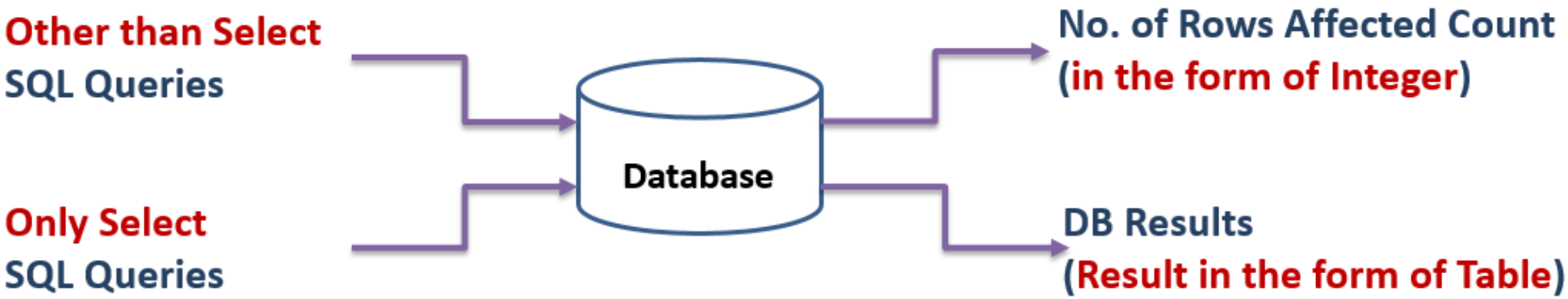
1. select * from ABC;
2. create database DB_NAME;
3. select * from ABC where X = 1;
4. insert into ABC values (1, 'Praveen');

Dynamic SQL Queries

ANY SQL Queries which “MUST have conditions” AND “One/More conditions value get decided at runtime” are known as "Dynamic SQL Queries"

Examples:

1. select * from ABC where X=? and Y=?;
2. select * from ABC where X=1 and Y=?;
3. insert into ABC values (?, 'Praveen');



JDBC Statements

1. Statement
   (for Static SQL Queries)

2. PreparedStatement
   (for Dynamic SQL Queries)

1. int executeUpdate()
   (Other than Select SQL Queries)

2. ResultSet executeQuery()
   (ONLY for Select SQL Queries)

Processing the Results

- Whenever we issue SQL Queries to RDBMS Application via JDBC there are two kinds of results expected out of RDBMS Application

1. No. of Rows Affected Count
2. DB Results

- JDBC returns

1. "No. of Rows Affected Count" as "Integer Value"
2. "DB Results" in the form of "ResultSet Object"

java.sql.ResultSet

- Its an interface & an Object of ResultSet is an "Object representation of DB Results" produced by Select SQL Query
- ResultSet object is produced by invoking "executeQuery()" Method on Statement OR PreparedStatement Objects
- ResultSet consists of N number of Rows with each row containing N number of Columns
- Number of rows and columns in Resultset directly depends on "where condition" & "column list" respectively in "Select SQL Query"
- ResultSet object may consist of "Zero/More" OR "Zero/One" rows
- ResultSet consists of "Zero/One" row if where condition is on Primary Key with "equal to (=)“ for rest of the cases it consists of "Zero/More" rows

ResultSet Continued …

- If ResultSet consist of zero/more rows of data then we must use "while loop"

- If ResultSet consist of zero/one row of data then we can use either "while loop" or "if block“ (preferred)

- Once the ResultSet is produced, data from ResultSet can be extracted as follows

1. Move to desired Row by calling necessary ResultSet methods (next(), first(), last(), etc.,)
2. Retrieve the desired column value using any one of the below getXXX() methods
   - public XXX getXXX(String columnName) throws SQLException
   - public XXX getXXX(int columnIndex) throws SQLException

   where XXX = Java Data Type corresponding to DB Table column data type

NOTE : getXXX() methods are the ONLY way to retrieve data from ResultSet object

Close JDBC Objects ???

- JDBC Objects such as “Connection”, “Statement”, “PreparedStatement”, “ResultSet” etc., make use of memory
- In case of Connection Object, further RDBMS Application resources are consumed
- Also memory consumed by ResultSet object is comparatively more compared to other JDBC Objects
- Hence forgetting to close any of the JDBC objects "will heavily impact Java as well as RDBMS application performance" & Garbage Collection should not be relied upon
- So it's important to close any of the JDBC Object as soon as their job is done
- To close any of the JDBC Objects invoke "close()" method

TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Thank You !!!

No.01, 3rd cross Basappa Layout, Gavipuram Extension,
Kempegowda Nagar, Bengaluru, Karnataka 560019

sagar.g@testyantra.com
gurupreetham.c@testyantra.com
praveen.d@testyantra.com

www.testyantra.com

EXPERIENTIAL
learning factory