



Module: JDBC

Module Overview

- In this module, students will be able to familiarize themselves with the essential topics such as JDBC, JDBC driver types, JDBC Connection, execute queries, close connection object, etc.

Module Objective

At the end of this module, students should be able to demonstrate appropriate knowledge, and show an understanding of the following:

- To understand JDBC
- To know Type1: Jdbc-Odbc Bridge Driver
- To know Type2: Native-API driver
- To know Type3: Network Protocol driver
- To know Type4: Thin driver
- To understand JDBC Connection
- To understand Register JDBC Driver
- To understand how to create connection object
- To understand how to create statement object
- To know how to execute queries
- To understand close connection object

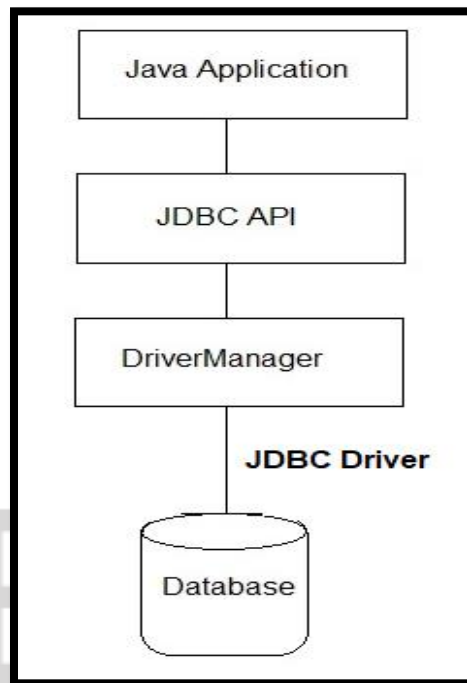


Introduction to JDBC

- JDBC can also be defined as the platform independent interface between a relational database and Java programming.
- JDBC stands for Java Database Connectivity.
- JDBC is an Application Programming Interface(API).
- Java API is used to connect and execute the query with the database.
- JDBC is used to interact with various type of Database such as Oracle, MS Access, My SQL, SQL Server, etc.
- JDBC API uses JDBC drivers to connect with the database.

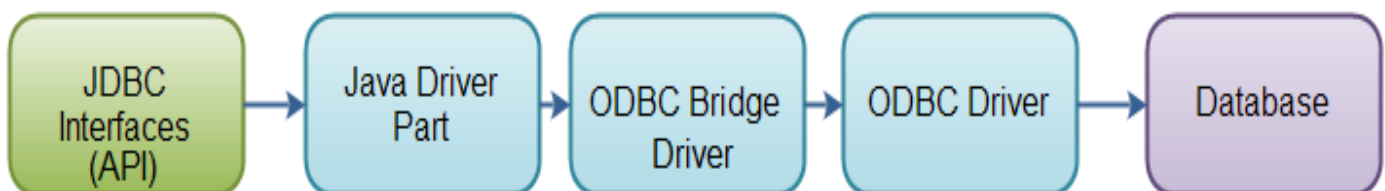
Java JDBC

- Java Database Connectivity is the Java API that manages connecting to a database, issuing queries and commands, and handling result sets obtained from the database.
- The JDBC API makes is possible to query and update relational databases, as well as call stored procedures, and obtain meta data about the database.
- The Java JDBC API is a set of interfaces.
- JDBC drivers are actually a set of classes that implement these interfaces.
- JDBC API, helps us to save, update, delete and fetch data from the database.
- It is like Open Database Connectivity (ODBC) provided by Microsoft.
- JDBC works with Java on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.
- There are 4 types of JDBC drivers, they are:
 1. Jdbc-Odbc Bridge Driver
 2. Native-API driver
 3. Network Protocol driver
 4. Thin driver



Type1: Jdbc-Odbc Bridge Driver

- This Driver will take the support of ODBC Driver to communicate with the database.
- Driver convert JDBC calls into ODBC calls and ODBC Driver convert ODBC calls into database-specific calls.

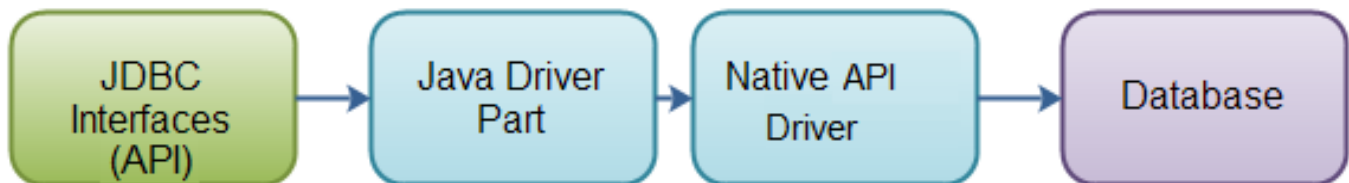




- **Advantages :**
 - It is database independent driver.
 - It is very easy to use.
 - Not require to install this driver separately
- **Disadvantages:**
 - It is the slowest driver.
 - Platform dependent driver.

Type2: Native-API driverAC

- The Native API driver uses the client side libraries of the database.
- Native API driver converts JDBC calls into database specific native libraries calls and these calls are directly understood by the database engine.
- Vendors like Oracle, IBM, etc. use this driver for their enterprise databases.

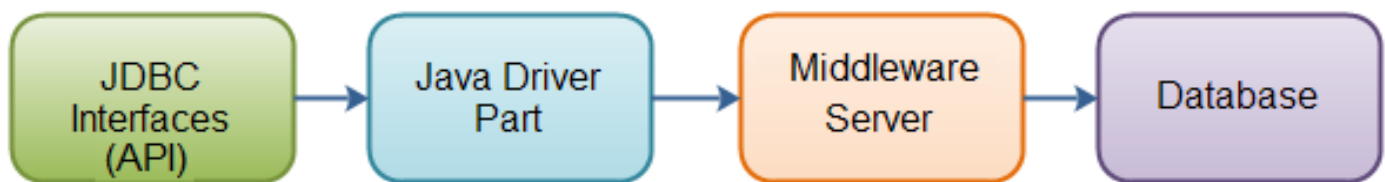


- **Advantages:**
 - Good performance as compared to the type1 driver.
 - Type2 Drivers are operating system specific.
- **Disadvantages:**
 - It is a database dependent driver.
 - It is a platform dependent driver.



Type3: Network Protocol driver

- Network protocol driver converts JDBC calls into middleware specific calls, the middleware server convert middleware specific calls into database specific calls.
- A single driver can actually provide access to multiple databases.



- **Advantage**
 - It don't require any native library to be installed.
 - It is database independent driver.
 - No client side libraries are required.
- **Disadvantage**
 - Slow due to increase number of network call.
 - Maintenance of Network Protocol driver becomes costly.



Type 4: Thin driver

- The thin driver converts JDBC calls directly into the client specific database protocol.
- Thin driver directly communicates with the database by using database specific native protocol provided by the database client.



- **Advantage:**
 - Better performance than all other drivers.
 - Platform independent Driver.
 - No software is required at the client.
- **Disadvantages:**
 - Drivers depend on the Database.

Introduction to Database services

- A database is an organized collection of data, so that it can be easily accessed and managed.
- It can organize data into tables, rows, columns, and index that will make it easier to find relevant information.
- There are many databases available like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.
- Database is a systematic collection of data.
- It supports storage and manipulation of data.
- Databases make data management easy.



- It can organize data into tables, rows, columns, and index that will make it easier to find relevant information.
- Example: Let's us consider the face book. It needs to store, manipulate and present data related to members, their friends, member activities, messages, advertisements and lot more.

JDBC

- JDBC stands for Java Database Connectivity.
- JDBC is a Java API to connect and execute the query with the database.
- Basic database operations such as Create, Retrieve, Update and Delete (CRUD) can be performed using JDBC, which stands for Java Database Connectivity API.
- It is like Open Database Connectivity (ODBC) provided by Microsoft.

JDBC Connection

- There are some simple steps to connect any java application with the database using JDBC.
- The following 5 steps are the basic steps involve in connecting a Java application with Database using JDBC.
- Those simple steps are as follows:
 1. Register JDBC Driver
 2. Create connection object
 3. Create statement object
 4. Execute queries
 5. Close connection object



Register JDBC Driver

- The registration is done only once in program, a JDBC driver can be register in one of two following ways.
- Method 1: The most common approach to register a driver is to use **Class.forName()** method.
- This method is preferable because it allows the driver registration configurable and portable.

- The **forName()** method of class is used to register the driver class.

- Method 1 : **Class.forName()**

We just need to put vender's jar in the classpath, and then JDBC driver manager can detect and load the driver automatically.

- Syntax :

```
Class.forName("com.mysql.jdbc.Driver");
```

- The registration is done only once in program, a JDBC driver can be register in one of two following ways.
- Method 1: The most common approach to register a driver is to use **Class.forName()** method.
- This method is preferable because it allows the driver registration configurable and portable.

- Method 2 : **DriverManager.registerDriver()**

The second approach you can use to register a driver, it accepts an object of the driver class as a parameter and, registers it with the JDBC driver manager.

- Syntax :

```
Driver myDriver = new com.mysql.jdbc.Driver();
```

```
DriverManager.registerDriver(myDriver);
```

- Method 2: The second approach to register a driver, is to use the Static **DriverManager.registerDriver()** method.
- This method is used when we use a non JDK compliant JVM.



Create connection object

- Connecting to a database, connection object is required.
- The Connection object uses a DriverManager.
- DriverManager has a method called getConnection.
- Following are the three forms of method to create a connection object.

1. Using Database URL : It includes the username and password and has the following general syntax form.

– Syntax :

```
String URL = "jdbc:oracle:thin:username/password@amrood:1521:EMP";
```

```
Connection conn = DriverManager.getConnection(URL);
```

- DriverManager has a method called getConnection. This needs a host name (which is the location of your database), a username, and a password. If a connection is successful, a Connection object is created.

2. Using Database URL with username and password:

It takes database URL, a user name and password as arguments and has the following general syntax form.

– Syntax:

```
String URL = "jdbc:oracle:thin:@localhost:1521:loTDB";
```

```
String USER = "Jetking";
```

```
String PASS = "Skillking";
```

```
Connection conn = DriverManager.getConnection(URL, USER, PASS);
```

3. Using a Database URL and Properties Object : It takes a database URL and a Properties object as argument.

– Syntax :

```
String URL = "jdbc:oracle:thin:@loTDB";
```

All rights reserved.



```
Properties properties = new Properties();
properties.put("user", "Skillking");
properties.put("password", "jetking");
Connection conn = DriverManager.getConnection(URL, properties);
```

Create statement object

- Once a connection is established we can interact with the database.
- There are 3 types of Statements, as given below:
 - JDBC Statement,
 - Callable Statement
 - Prepared Statement
- The object of statement is responsible to execute queries with the database.
- Syntax :
 - public Statement createStatement() throws SQLException
- Example to create a SQL statement
 - Statement s = con.createStatement();

- **JDBC Statement :**

It can be used for general purpose access to the database. It is useful when you are using static SQL statements at runtime.

- **Callable Statement:**

It can be used when we want to access database stored procedures.



- **Prepared Statement:**

It can be used when same SQL statement is use many times. The Prepared Statement interface accepts input parameters at runtime.

Execute 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.
- This method can be used to get all the records.
- There are multiple types of queries. Some of them are as follows:
 - Query for updating table in a database.
 - Query for inserting table in a database.
 - Query for retrieving data, etc.
- Syntax :
 - `public ResultSet executeQuery(String sql)throws SQLException`

Close connection object

- Once we are done with using Connection, we need to explicitly close it by calling close() method.
- The close() method of Connection interface is used to close the connection.
- Any connection left open is waste of resource and lead to various exceptions.
- **Syntax :**
 - `public void close() throws SQLException`



- **Example:**

- `con.close();`



Lab Activity:

Trainer will ask the participants to refer to the LMS and observe the lab on JDBC connection & perform the same.

Lab activity

- Connect java application with the database using JDBC.

Post completion of the activity lab, discuss with trainer for any query.