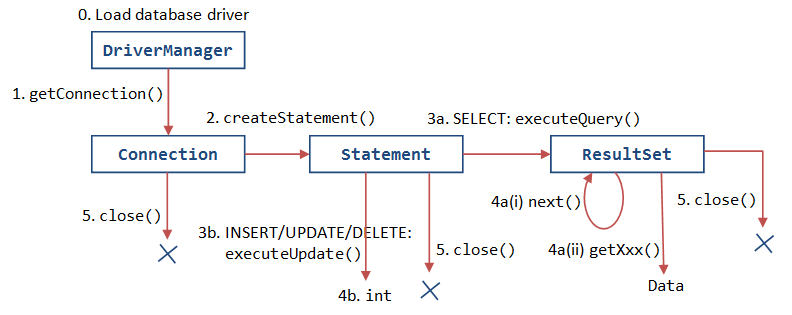
Java DataBase Connection

JDBC is an API or a tool which will help java application to connect to the any database and execute SQL statements.



## Download MySQL installer from the below link

https://dev.mysql.com/downloads/installer/

## Download JDBC driver for MySQL from the below link

<https://dev.mysql.com/downloads/connector/j/>

\*\*\*\***Once Driver is downloaded unzip the ZIP-file**

## Adding MySQL JDBC driver to the java project

1. Right click on your current **project folder.**
2. Select **build path and** click **configure build path**
3. Click on **ClassPath and** clickon **Add External JARs…**
4. **Locate Unzipped folder and select**

### Register

Register the driver and load the driver

### Crearte

Use the Connection object and create Statement

### Execute

Use the Statement Object and execute SQL query

JDBC

Complete JDBC guide

### Process

Iterate over the ResultSet using the Cursor to get data for process

### Close

use the Connection object and close the Database connection

# Steps to use JDBC

“JDBC Methods”

DriverManager :

The JDBC DriverManager is a very important class that defines objects which connect Java applications to a JDBC driver.

This class is mainly useful for the simple application, the most frequently used method of this class is DriverManager.getConnetion()

**getConnection(url, username, password)** :

try to establish a connection to the database by using the given database URL , username, password for driver from the set of registered JDBC drivers is selected and return the object of Connection. If any problems occur such as wrong url, wrong password, wrong username etc then, SqlException will be thrown.

Connection :

**createStatement()** :

this method will create and return a statement object which can be used to execute appropriate execute() methods SqlException will be thrown.

**executeUpdate()**

Executes the given SQL statement, which may be an INSERT, UPDATE, or DELETE statement or an SQL statement that returns nothing.

(1) the row count for CREATE , UPDATE , DELETE

(2) 0 for SQL statements like DROP,TRUNCATE etc. that return nothing

**executeQuery()**

Executes the given SQL SELECT statement, that returns a ResultSet which will contain the records which are selected by select statement.

**close()**

closes the I/O connection with database.

Statement :

* When you are using static SQL statements at runtime Statement can be used.
* We cannot pass dynamic parameters for the Statement.
* If you want to run SQL query only once then this interface is preferred over Statement.

PreparedStatement :

* If you want to execute Statement object many times then we should use PreparedStatement object as it reduces the execution time.
* PreparedStatement object is faster than the Statement object as it is precompiled.
* In PreparedStatement we use "?" as a parameter placeholder for those values whose values are not known when the Sql statement is created.