JDBC Environment Setup

* To start developing with JDBC, you should setup your JDBC environmenJDBC is an unofficial acronym for java database connectivity.
* JDBC helps us to connect to a database and execute SQL statements against a database.  It provides RDBMS access by allowing you to embed SQL inside Java code.
* JDBC is an API (application program interface), means it has set standard for the database vendor. JDBC API has interfaces and abstract classes. Database vendor has to implement these interfaces and abstract classes, so that a single code can be used for any database. Every database is unique in its own way and we don’t need to create separate programs for each database.
* JDBC  is broken down into two packages:
  1. java.sql:Referred as the JDBC Core API.
  2. javax.sql: Referred as the JDBC Optional Package API. (In JDBC 2.0, this package was formerly known as the JDBC Standard Extension API.)

t by following the steps shown below. We assume that you are workinTo start developing with JDBC, you should setup your JDBC environment by following the steps shown below. We assume that you are working on a Windows platform.

## **Install Java**

Follow the instructions to download Java, and run the **.exe** to install Java on your machine. Once you have installed Java on your machine, you would need to set environment variables to point to correct installation directories.

g on a Windows platform.

## **Install Java**

Java SE is available for download for free. Toad please download a version compatible with your operating syst

Follow the instructions to download Java, and run the **.exe** to install Java on your machine. Once you have installed Java on your machine, you would need to set environment variables to point to correct installation directories.

# jdbc-environment-0JDBC - Environment Setup

## Configuring a JDBC development environment -loading the jar file

Database vendors provide the implementation for JDBC API in jar file. We have to load this jar file to our project. Two ways to load the jar file:

* Copy the jar files to jre/lib/ext folder, set classpath
* Add jars to eclipse. (Try to follow this)

### Copy the jar files to jre/lib/ext folder, set classpath

* Get the jar files from the location
  1. Oracle DB - C:\oraclexe\app\oracle\product\10.2.0\server\jdbc\lib\ojdbc14.jar
  2. MySql DB - C:\Program Files (x86)\MySQL\Connector J 8.0\ mysql-connector-java-8.0.11.jar
* Copy the jar files to the location C:\Program Files\Java\jdk1.7.0\_02\jre\lib\ext
* Set the classpath in environment variables. For Variable value use the following

#### Oracle DB

               C:\oraclexe\app\oracle\product\10.2.0\server\jdbc\lib\ojdbc14.jar

               Or

               C:\Program Files\Java\jdk1.7.0\_02\jre\lib\ext\ ojdbc14.jar

#### MySql DB

              C:\Program Files (x86)\MySQL\Connector J 8.0\ mysql-connector-java-8.0.11.jar

              Or

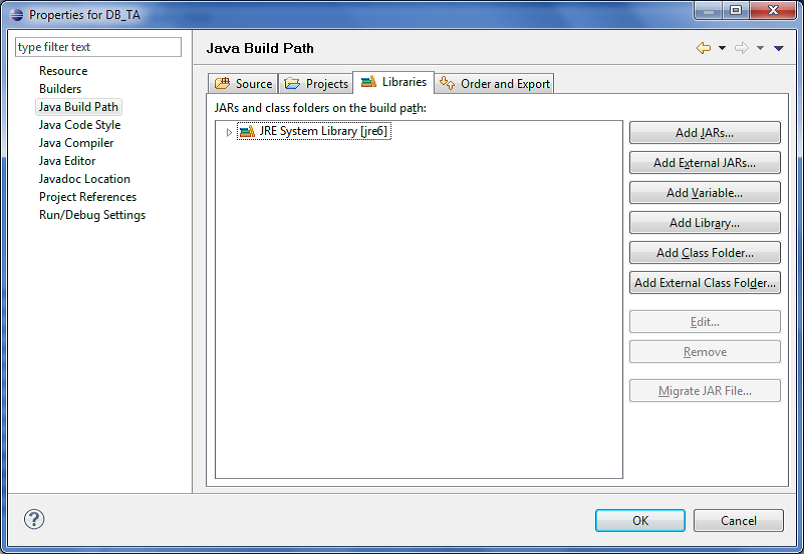
              C:\Program Files\Java\jdk1.7.0\_02\jre\lib\ext\ ojdbc14.jar\ mysql-connector-java-5.1.27-bin.jar

If you are not able to find the jar click this link to download: <https://www.java4coding.com/01downloads/jars/mysql-connector/mysql-connector-java-8.0.11.jar>

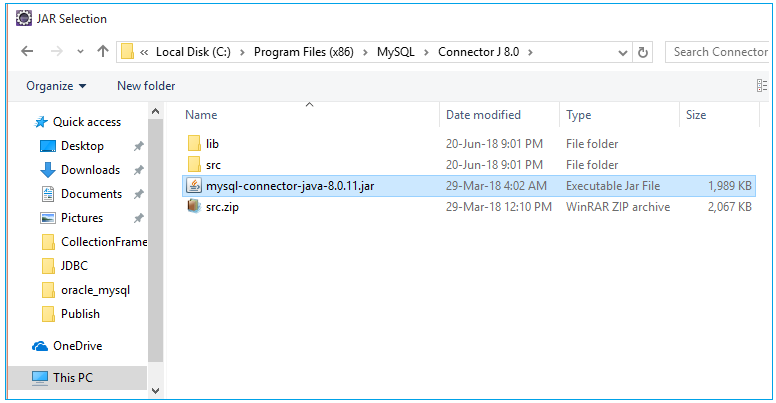
In the above ways of setting JDBC environment, we suggested to keep the jar inside ext folder of JDK. Actually you can keep the jar file anywhere and can be added that folder to classpath. But it is recommended to collect all the jars inside ext folder of JDK, so that it can be used throughout the system wherever JDK is used.

### Add jars to eclipse

Eclipse main menu-> Project -> Properties -> Java Build Path -> Libraries



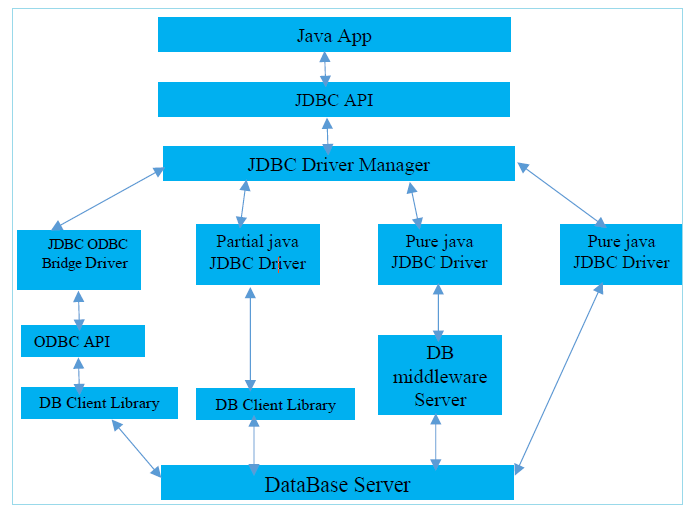
Click Add External JARs -> Select the JDBC driver (jar file). -> Open



In our example we are using MySql 8.0 version. This version requires java 1.8 and above version. Below table gives the details on version support.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Connector/J version** | **JDBC version** | **MySQL Server version** | **JRE Supported** | **JDK Required for Compilation** | **Status** |
| 8.0 | 4.2 | 5.5, 5.6, 5.7, 8.0 | 1.8.x | 1.8.x | General Availability |
| 5.1 | 3.0, 4.0, 4.1, 4.2 | 5.5, 5.6\*, 5.7\*, 8.0\* | 1.5.x, 1.6.x, 1.7.x, 1.8.x\* | 1.5.x and 1.8.x | General Availability |

## Types of JDBC Drivers



Generally driver means software which enables any external device to interact with computer. JDBC Driver is software that enables java application to interact with the database. There are four types of JDBC drivers in use:

* Type 1: JDBC-ODBC Bridge driver
* Type 2:  partial Java driver: Native-API driver
* Type 3: pure Java driver with database middleware:  Network Protocol driver
* Type 4: pure Java driver directly to database: Thin driver

## JDBC-ODBC bridge driver

ODBC drives acts as a bridge between JDBC and database. The JDBC-ODBC bridge driver converts database calls into the ODBC calls. Now ODBC driver connects to the database.

## Native-API driver

These drivers are typically provided by the database vendors. Native-API acts as a bride between JDBC and database. Native–API driver converts database calls into the native C/C++ API calls. Now Native API connects to the database.

## Network Protocol driver

Middleware server acts a bridge between JDBC and database. This Middleware server converts JDBC calls to database understandable calls.

## Thin driver

The thin driver converts JDBC calls directly into database understandable calls. That is why it is known as thin driver. It is completely written in Java language.