Day19 Agenda

JDBC – Java DataBase Connectivity API (It’s an API used to interact with any database or RDBMS)

JDBC is a specification. It has many interfaces and DB developers has given implementation to these interfaces. (Driver)

Assume, you have hp laptop and a new Cannon Printer. You need to connect your printer with your laptop.

(The printer is unboxed and all the pre check-up is done) Powering ON, Network setup,

1) Power ON printer, Download & Install the Driver (Cannon printer driver software)

2) Physical wired/wireless connection between your computer and printer

3) Do Test print

4) Actual print

5) Powering Off printer.

Steps to interact with the Database [Java is a Process/Application & RDBMS/DB Server is a different process]

1. Loading & Registering the Driver
2. Establishing the connection between Java and Database
3. Create & Execute the SQL queries
4. Store & Process the Result
5. Closing all the resources.

|  |  |
| --- | --- |
| MySQL Driver Name | com.mysql.cj.jdbc.Driver |
| Oracle Driver Name | oracle.jdbc.driver.OracleDriver |
| Postgres Driver Name | org.postgres.Driver |

|  |  |
| --- | --- |
| MySQL URL | jdbc:mysql://localhost:3306/<db\_name> |
| Oracle URL | Jdbc:oracle:thin:@localhost:1521/xe |
| Postgres URL | Jdbc:postgres://localhost:5432/<db\_name> |

IP address of localhost = 127.0.0.1

Local host means the Database server is running on the same system

Important Interfaces in JDBC API (All the classes and interfaces of JDBC api are available in java.sql package)

1. Driver (I)
2. Connection (I)
3. Statement/PreparedStatement/CallableStatement (I)
4. ResultSet (I)
5. DatabaseMetaData(I) & ResultSetMetaData (I)

Classes in SQL package

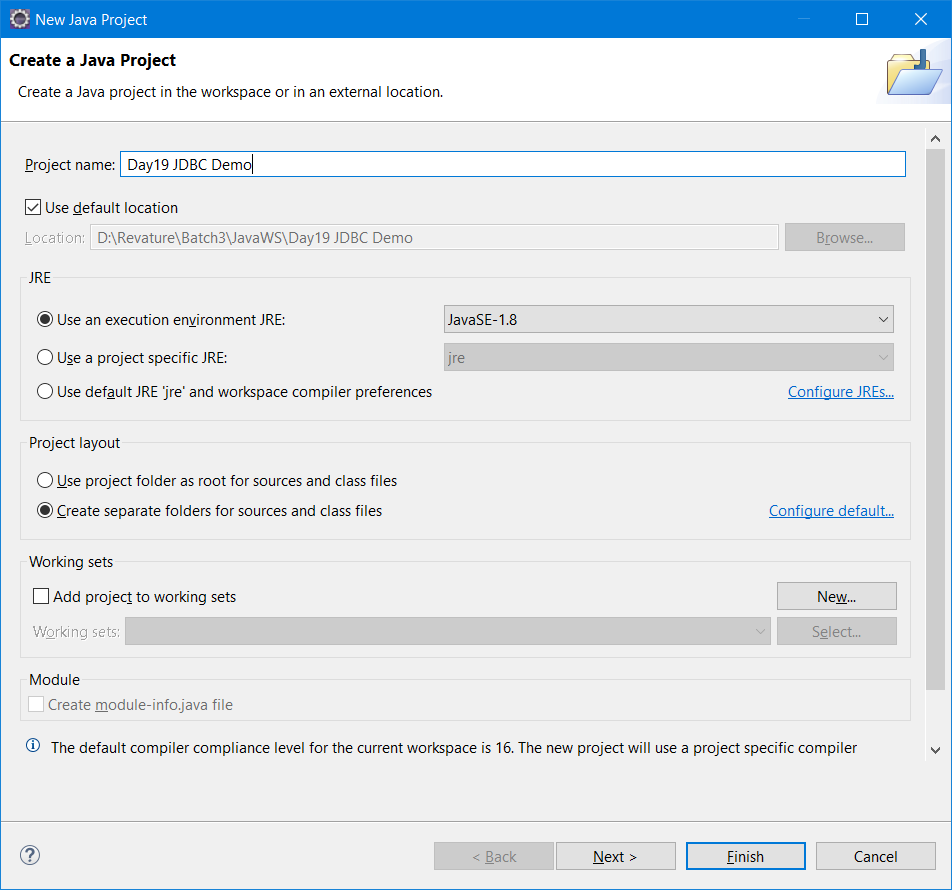
DriverManager[C]

Date[C]

Time[C]

Creating a JDBC application to interact with MySQL Database.

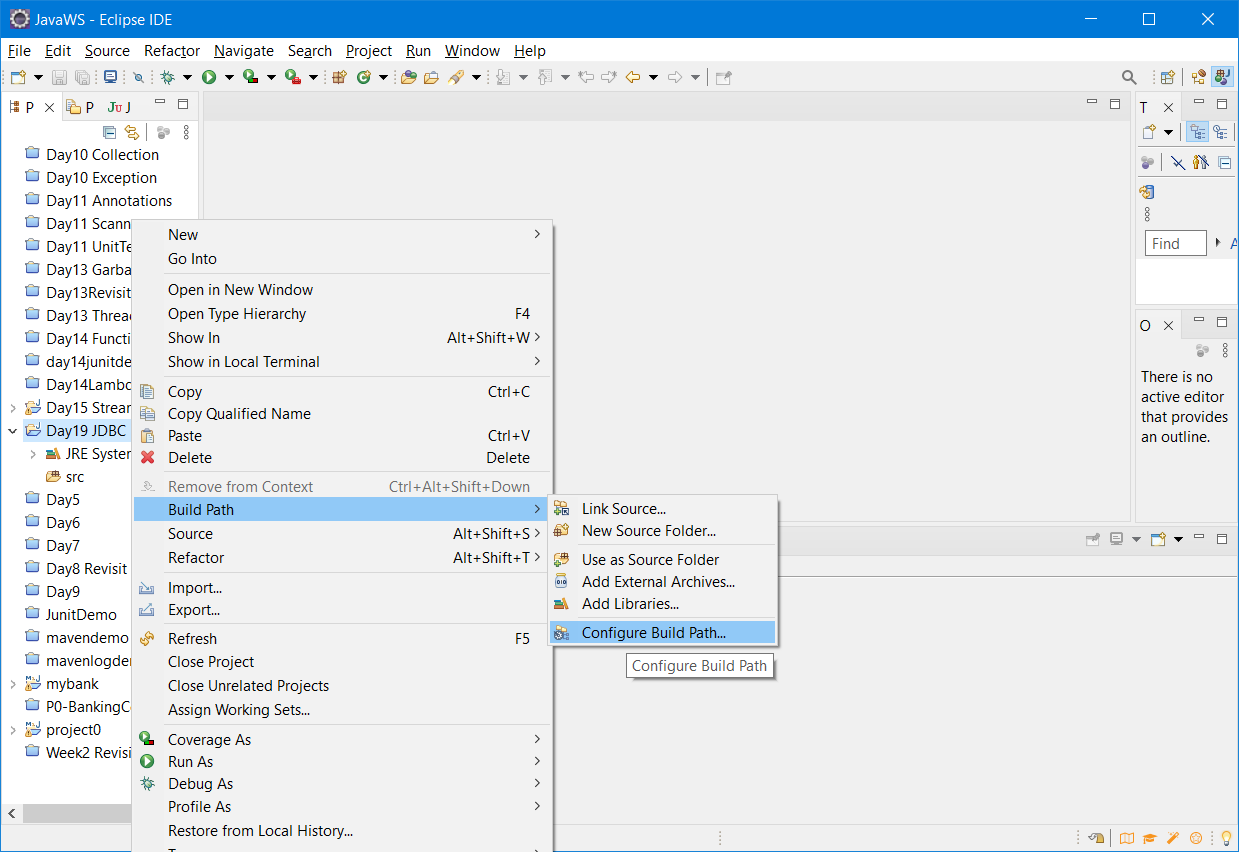
* Open Eclipse
* Create a New Java Project (File🡪New🡪Java Project)

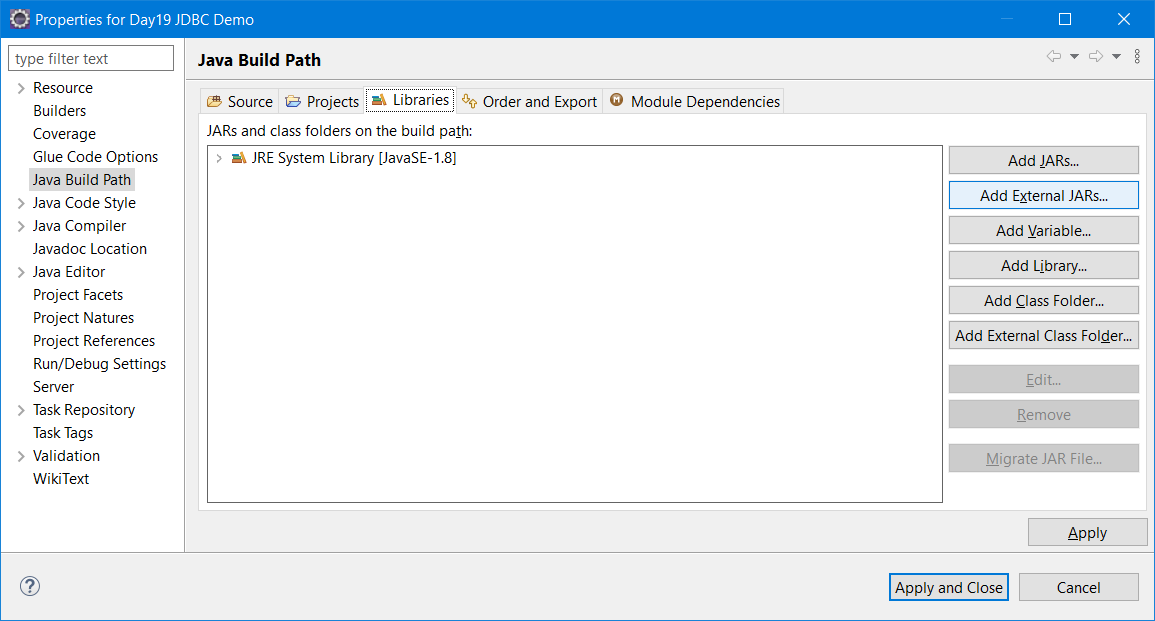


Click Finish button.

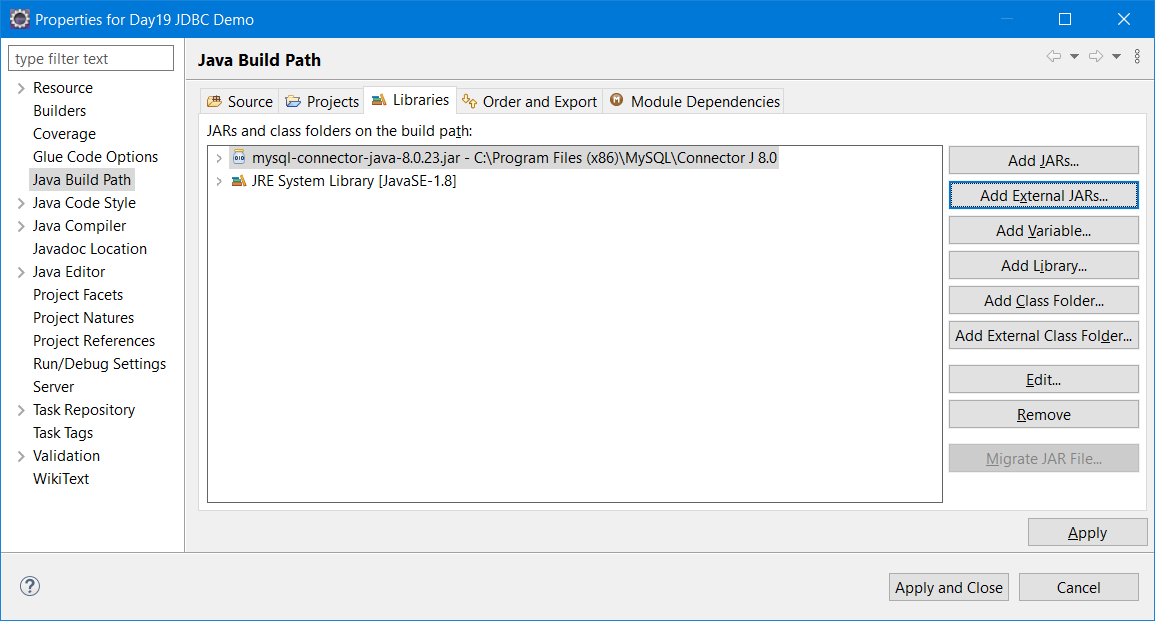
Adding Driver Jar to the Build Path

* Right click the project and select “Build Path” option 🡪 “Configure Build Path “



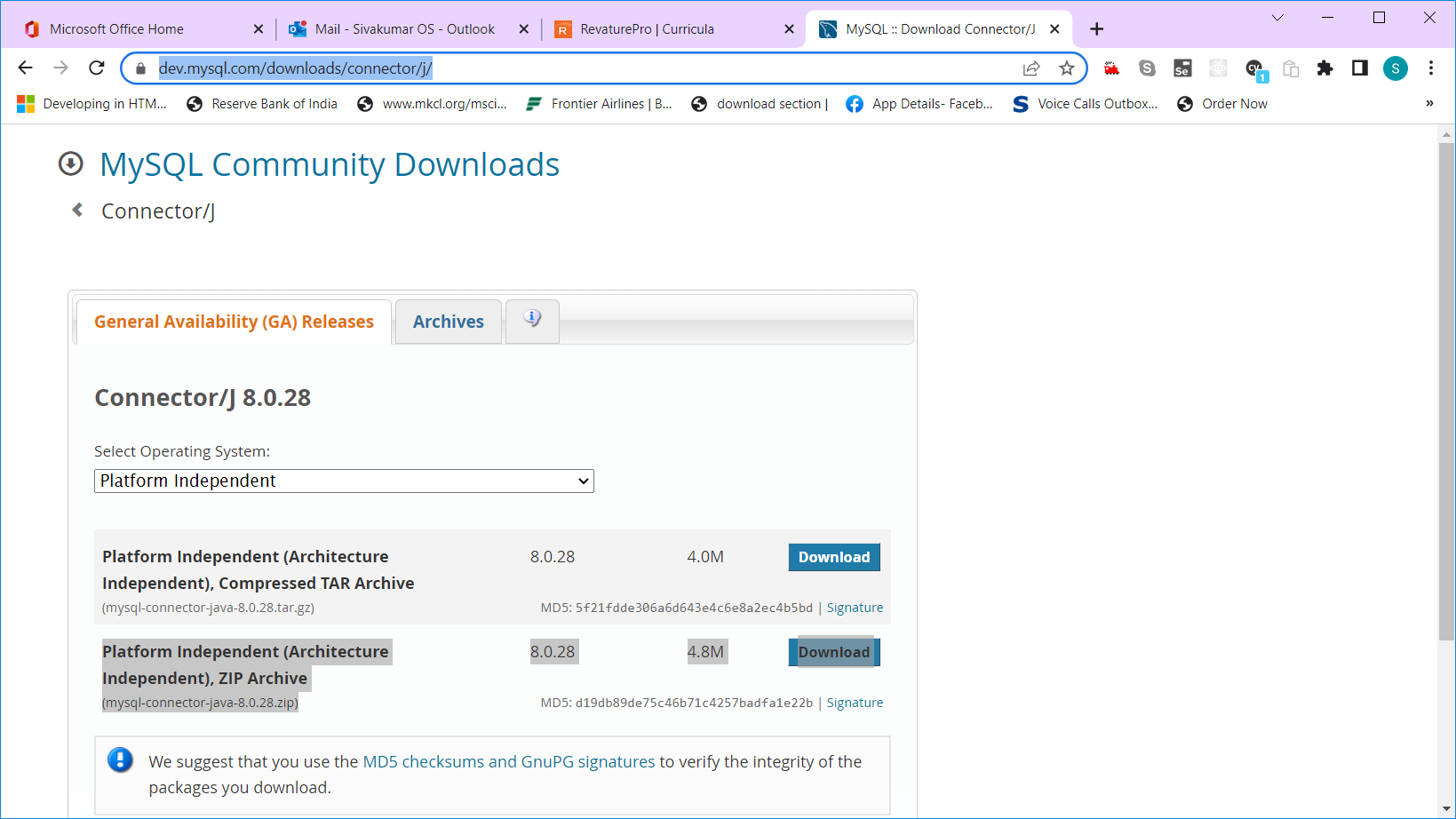
In the Libraries tab, click on “Add External Jars” button

From the following path - C:\Program Files (x86)\MySQL\Connector J 8.0 add the mysql-connector8.x.jar file(Driver Jar)

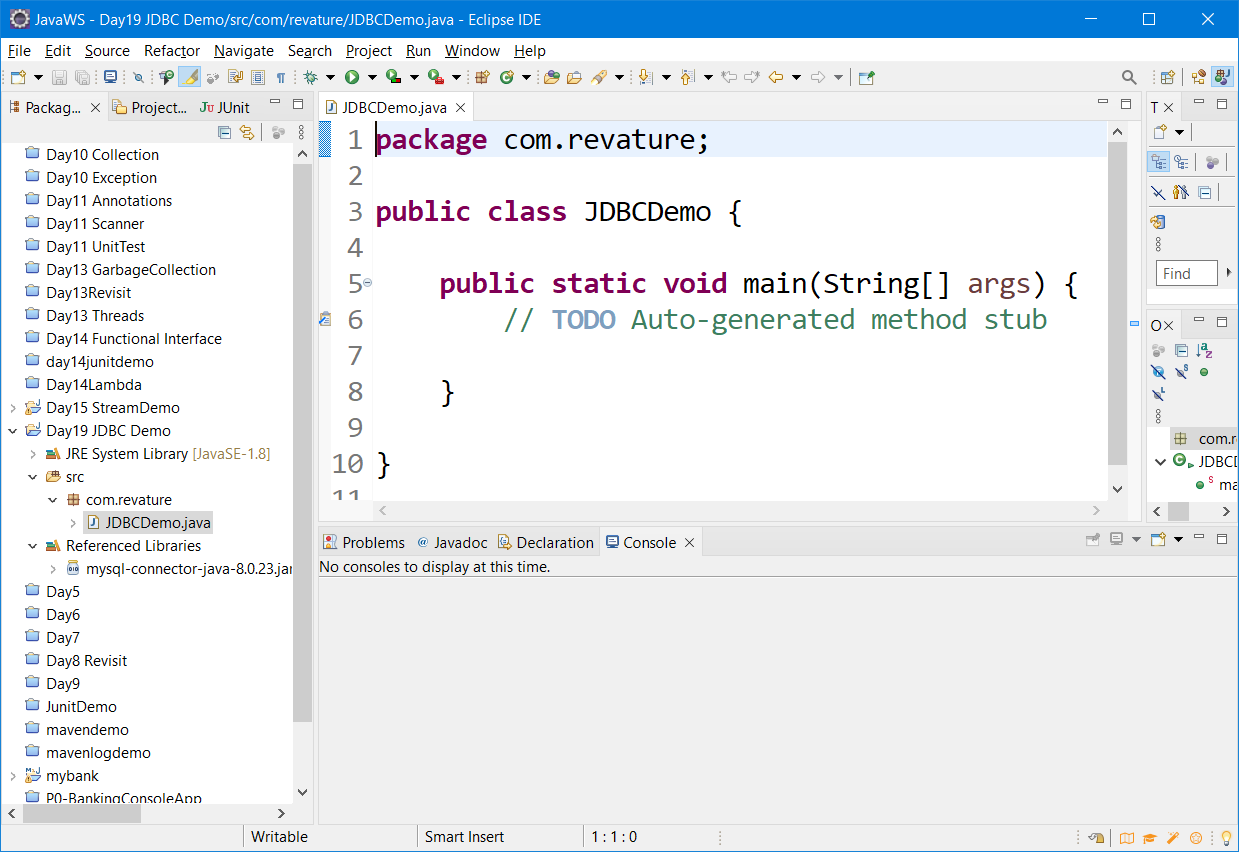


<https://dev.mysql.com/downloads/connector/j/> (If MySQL Driver Jar is not available in Programfiles(x86) folder, then download it from the official site. )

|  |  |  |  |
| --- | --- | --- | --- |
| **Platform Independent (Architecture Independent), ZIP Archive** | 8.0.28 | 4.8M | [**Download**](https://dev.mysql.com/downloads/file/?id=509728) |
| (mysql-connector-java-8.0.28.zip) |  |  |  |



Add a new Class “JDBCDemo” with main method in the SRC folder



CRUD – Create/Read/Update/Delete

Create – Need an Object

Read – ReadAll() – List of Objects will be returned & ReadById(int id) – will return a single object

Update – Need id (To update a particular record/row) and updated object

Delete – Need id (To delete a particular record/row)

Create – addObject() / add() – will not return anything

Read – ReadAll() & ReadById()

Update

Delete

5 methods for 4 operations in Database.

Statement & PreparedStatement are interfaces defined in java.sql package.

These are all the important interface used to execute any SQL Queries

PreparedStatement – Is also called as PrecompiledStatement. – If we are doing same operation many times, then preparedStatement will help you to increase the performance.

Database Transactions/Operations are costly process. Bcos it consumes more time.

Statement interface is best suitable for executing simple queries without parameters.

PreparedStatement is best suitable for executing queries with parameters.

CREATE SCHEMA `p0` ;

CREATE TABLE `p0`.`p0\_user` (

`id` INT NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(45) NULL,

`last\_name` VARCHAR(45) NULL,

`username` VARCHAR(45) NULL,

`password` VARCHAR(45) NULL,

`user\_type` VARCHAR(45) NULL,

PRIMARY KEY (`id`));

CREATE TABLE `p0`.`transactions` (

`id` INT NOT NULL AUTO\_INCREMENT,

`from\_accountId` VARCHAR(45) NULL,

`to\_accountId` VARCHAR(45) NULL,

`amount` DOUBLE NULL,

`transaction\_type` VARCHAR(20) NULL,

`timestamp` TIMESTAMP(2) NULL,

PRIMARY KEY (`id`));

CREATE TABLE `p0`.`account` (

`account\_id` INT NOT NULL AUTO\_INCREMENT,

`owner\_id` INT NULL,

`balance` DOUBLE NULL,

`account\_type` VARCHAR(45) NULL,

`approved` TINYINT NULL,

PRIMARY KEY (`account\_id`));