Programare avansata pe obiecte – laborator 7

***Andreea Ciocan*** *andreea.ciocan@endava.com*

# JDBC

JDBC (Java Database Connectivity) is an API for connecting and executing queries on a database.

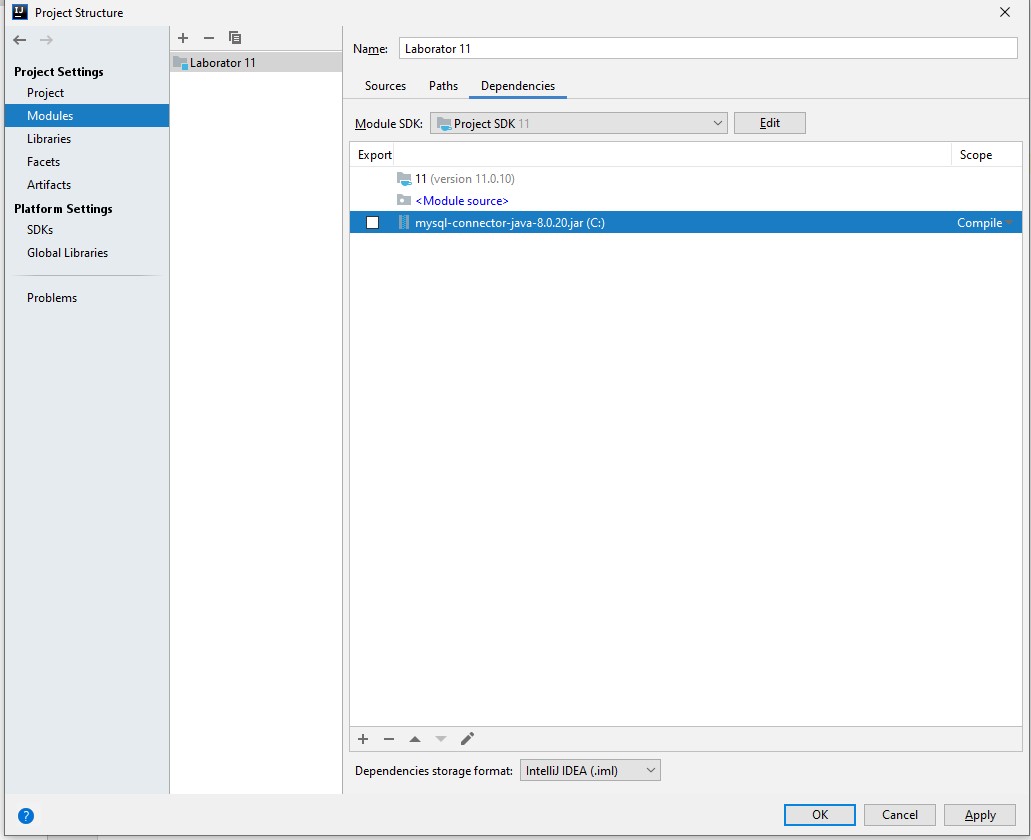
JDBC can work with any database if proper drivers are provided.

## Connecting to a Database

To connect to a database, we simply must initialize the driver and open a database connection.

**Registering the Driver:** We're using a MySQL database, we need the mysql-connector-java dependency: <https://mvnrepository.com/artifact/com.mysql/mysql-connector-j/8.0.33>

(Ctrl + Alt + Shift + S -> “+” from below or Alt+Insert and select downloaded jar)



## Creating the Connection

We will assume that we already have a MySQL server installed and running on localhost (default port 3306).

We will also need to create the database and user.

## Send SQL instructions to the database

We can use instances of type **Statement**, **PreparedStatement** or **CallableStatement**. These are obtained using the **Connection** object.

After executing a query, the result is represented by a **ResultSet** object, which has a structure like a table, with lines and columns. The ResultSet uses the **next()** method to move to the next line.

* **Statement** - The Statement interface contains the essential functions for executing SQL commands.  It is generally used for general**–**purpose access to databases and is useful while using static SQL statements at runtime.
  + **Implementation:**Once the Statement object is created, there are three ways to execute it.
    - **boolean execute(String SQL):** If the ResultSet object is retrieved, then it returns true else false is returned. Is used to execute [SQL DDL](https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/) (Data Definition Language) statements or for dynamic SQL.
    - **int executeUpdate(String SQL):** Returns number of rows that are affected by the execution of the statement, used when you need a number for INSERT, DELETE or UPDATE statements.
    - **ResultSet executeQuery(String SQL):** Returns a ResultSet object. Used similarly as SELECT is used in SQL.
* **PreparedStatement** - PreparedStatement objects contain precompiled SQL sequences. They can have one or more parameters denoted by a question mark.
* **Implementation:**Once the PreparedStatement object is created, there are three ways to execute it:
  + **execute():**This returns a boolean value and executes a static SQL statement that is present in the prepared statement object.
  + **executeQuery():**Returns a ResultSet from the current prepared statement.
  + **executeUpdate():**Returns the number of rows affected by the DML statements such as INSERT, DELETE, and more that is present in the current Prepared Statement.
* **CallableStatement** – used to call stored procedures o Setting input parameter values for the stored procedure is done like in the PreparedStatement interface, using setX() methods
  + If the stored procedure has output parameters, we need to add them using the registerOutParameter() method
  + **Implementation**: Once the callable statement object is created

**execute**() is used to perform the execution of the statement.

For the CallableStatement example to work, we need to create the stored procedure into the database (lab example):

*DELIMITER //*

*CREATE PROCEDURE insertPerson(OUT id int, IN name varchar(30), IN age double) BEGIN*

*INSERT INTO persons(name,age)*

*VALUES (name,age);*

*SET id = LAST\_INSERT\_ID();*

*END//*

*DELIMITER ;*

## Closing the Connection

When we are no longer using, it is necessary to close the connection to release database resources.