Java Database Connectivity API

- JDBC
- Driver
- Connection
- Statement
- ResultSet
- Examples

Before we get started

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

- MySQL Community Server
- MySQL Connector: Connector/J

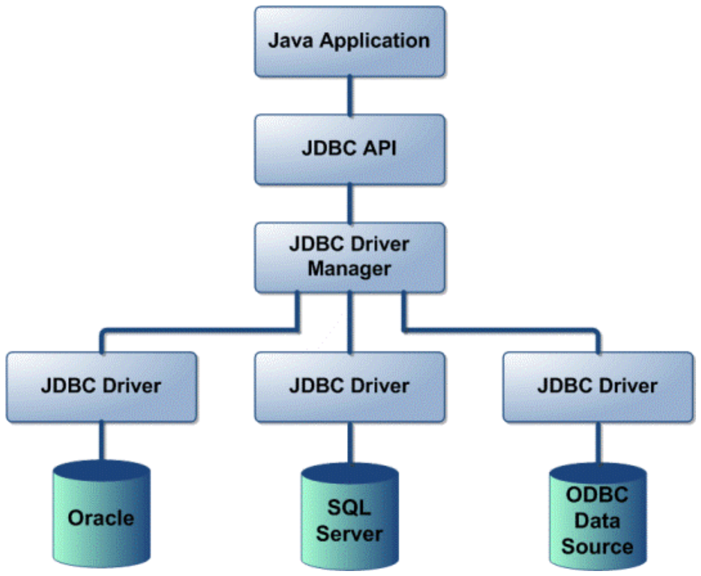
+ new maven project with dependency

```
<dependency>
	<groupId>mysql</groupId>
	<artifactId>mysql-connector-java</artifactId>
	<version>6.0.6</version>
</dependency>
```

Java Database Connectivity

JDBC is an API, which

- defines how a Java application may access a database.
- provides methods to query and update data in a database.
- is oriented towards relational databases.



JDBC Driver

JDBC is an API, so to work with real DBMS you need an implementation of this API.

MySQL Connector: Connector/J is an implementation of JDBC API for MySQL

java.sql.Driver is an interface in java library

com.mysql.jdbc.Driver in an implementation of java.sql.Driver in Connector J.

To work with BD you need a Driver:

Driver mysqlDriver = (Driver) Class.forName("com.mysql.jdbc.Driver").newInstance()

java.sql.DriverManager

DriverManager is a manager for drivers.

Each driver must be registered before use.

```
DriverManager.registerDriver(driver);
```

java.sql.Connection

A connection (session) with a specific database.

SQL statements are executed and results are returned within the context of a connection.

```
Driver driver = (Driver) Class.forName("com.mysql.jdbc.Driver").newInstance();
DriverManager.registerDriver(driver);
Connection connection = DriverManager.getConnection(“jdbc:mysql://localhost:3306/db_example?
user=tully&password=tully”);
```

Statements

Update statements: CREATE, DELETE, INSERT…

Query statements: SELECT

Interfaces:

- java.sql.Statement

  used for executing a static SQL statement and returning the results it produces.
- java.sql.PreparedStatement

  represents a precompiled SQL statement.
- java.sql.CallableStatement

  used to execute SQL stored procedures.

So, to make a request to DB you need a Statement object.

```
Statement stmt = connection.createStatement();
stmt.execute(“select * from user”);
stmt.close();
```

java.sql.ResultSet

A table of data representing a database result set, generated by executing a statement that queries the database.

A ResultSet object maintains a cursor pointing to its current row of data.

Initially the cursor is positioned before the first row.

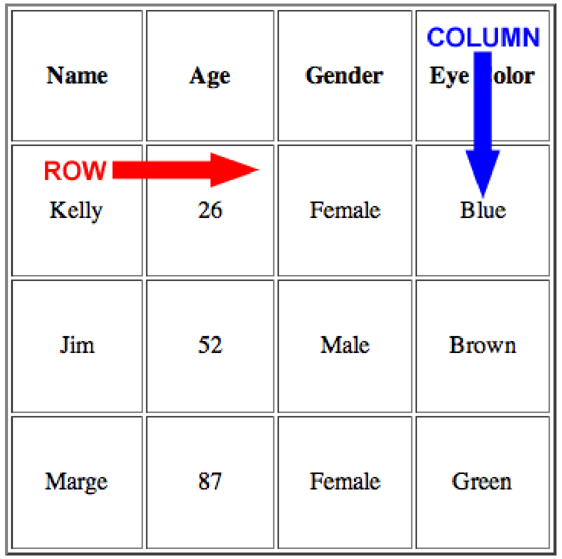
The next() method moves the cursor to the next row.

When there are no more rows in the ResultSet object it returns false.

How to use ResultSet object

To move from row to row:

- next()
- previous()
- isLast()



Name	Age	Gender	Eye olor
Kelly	26	Female	Blue
Jim	52	Male	Brown
Marge	87	Female	Green

To read from the column in the current row:

- By name: getBoolean(String name), getLong(String name)…
- By index: getBoolean(int index), getLong(int index)…

How to close the statement

Statement is AutoCloseable

You can create it inside try(…) block

```
try (Statement stmt = connection.createStatement()) {
        stmt.execute(“update user set age=100 where id=1”);
}
```

ResultSet object will be closed with Statement object