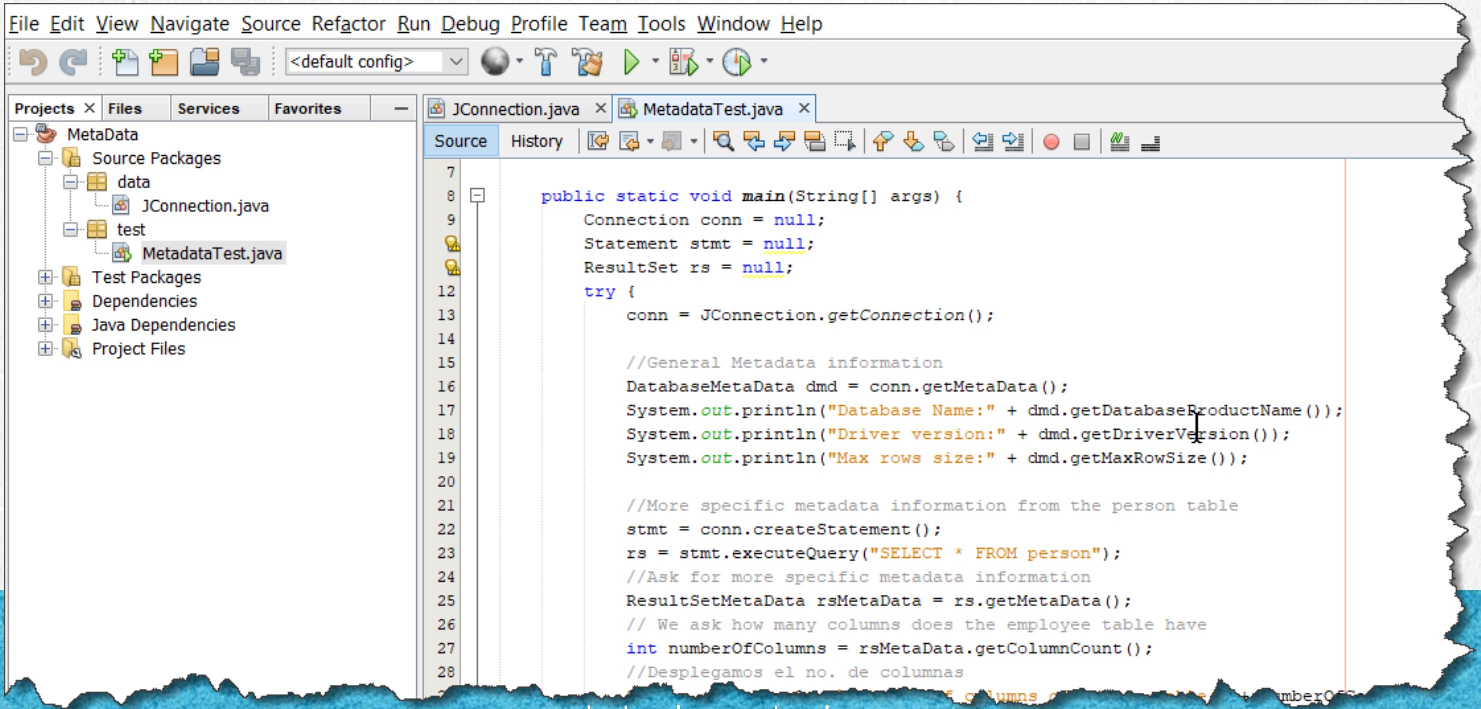
METADATA WITH JDBC

By the expert: Ubaldo Acosta

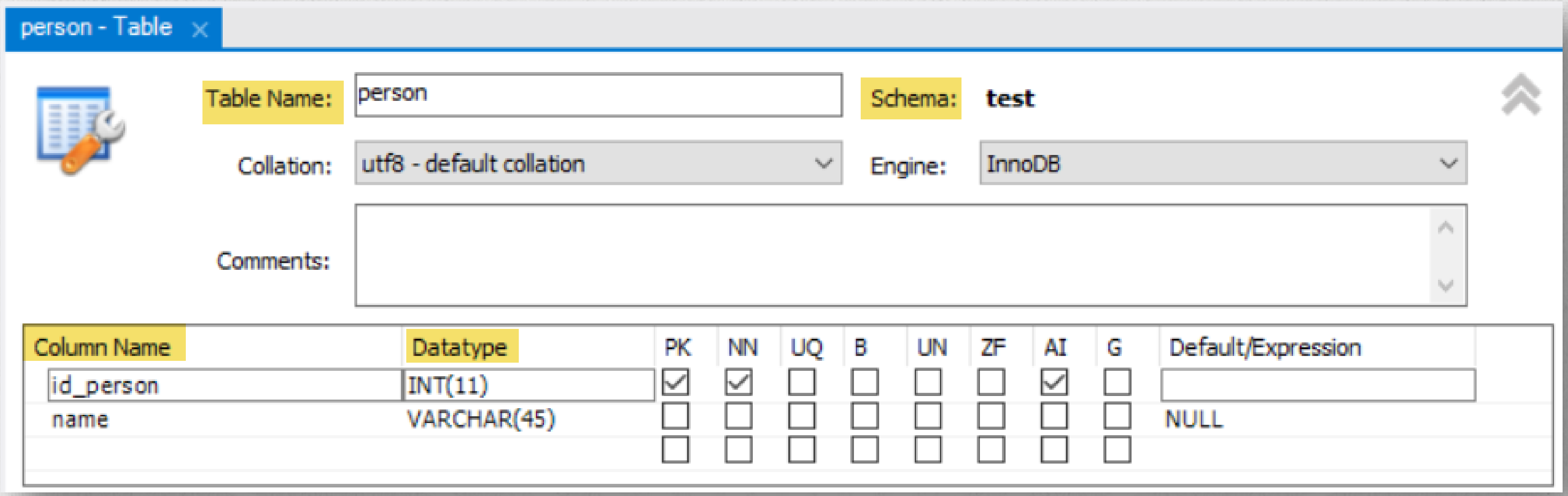
EXERCISE OBJECTIVE

Put into practice the concept of metadata. At the end we should observe the following:



```java
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            conn = JConnection.getConnection();

            //General Metadata information
            DatabaseMetaData dmd = conn.getMetaData();
            System.out.println("Database Name:" + dmd.getDatabaseProductName());
            System.out.println("Driver version:" + dmd.getDriverVersion());
            System.out.println("Max rows size:" + dmd.getMaxRowSize());

            //More specific metadata information from the person table
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT * FROM person");
            //Ask for more specific metadata information
            ResultSetMetaData rsMetaData = rs.getMetaData();
            // We ask how many columns does the employee table have
            int numberOfColumns = rsMetaData.getColumnCount();
            //Desplegamos el no. de columnas
```

SOME MEDATA TO OBTAIN FROM THE PERSON TABLE



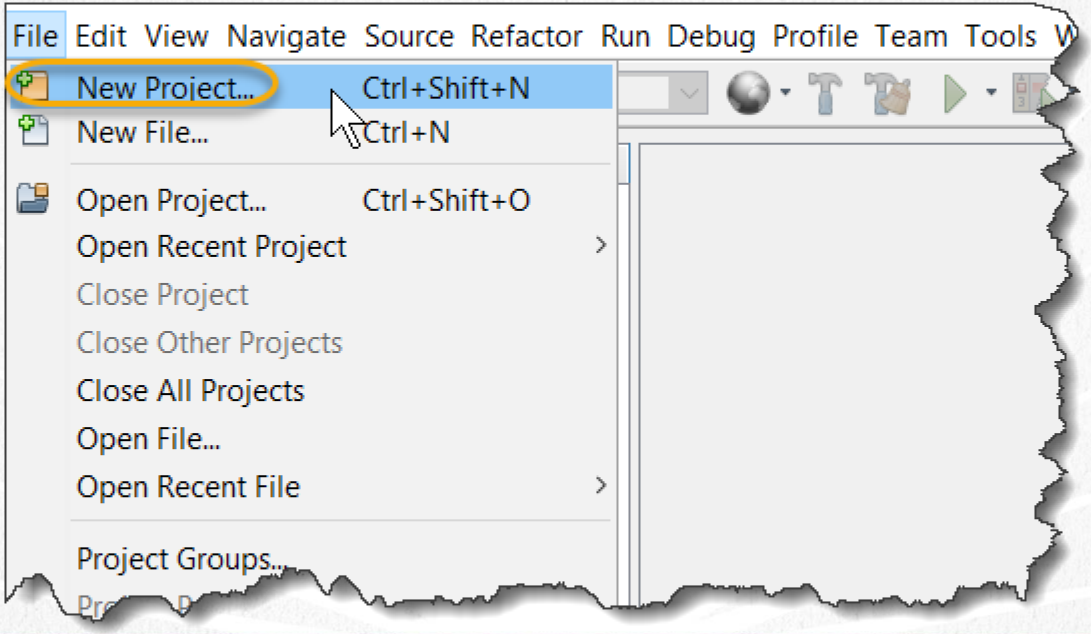
person - Table

Table Name:	person	Schema:	test
Collation:	utf8 - default collation	Engine:	InnoDB
Comments:			

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_person	INT(11)	✓	✓					✓		
name	VARCHAR(45)									NULL

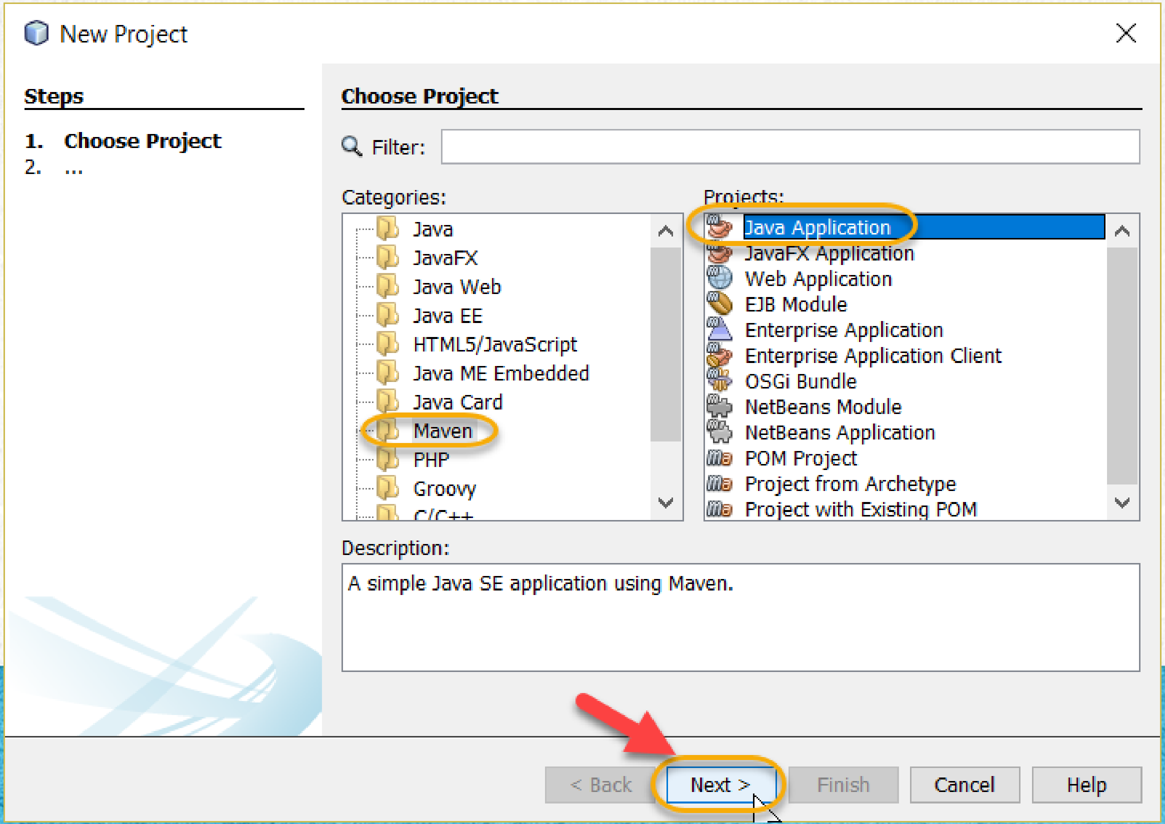
1. CREATE A NEW PROJECT

Create a new project:



1. CREATE A NEW PROJECT

Create a new project:



New Project

Steps

1. **Choose Project**
2. ...

Choose Project

Filter:

Categories:

- Java
- JavaFX
- Java Web
- Java EE
- HTML5/JavaScript
- Java ME Embedded
- Java Card
- Maven
- PHP
- Groovy
- C/C++

Projects:

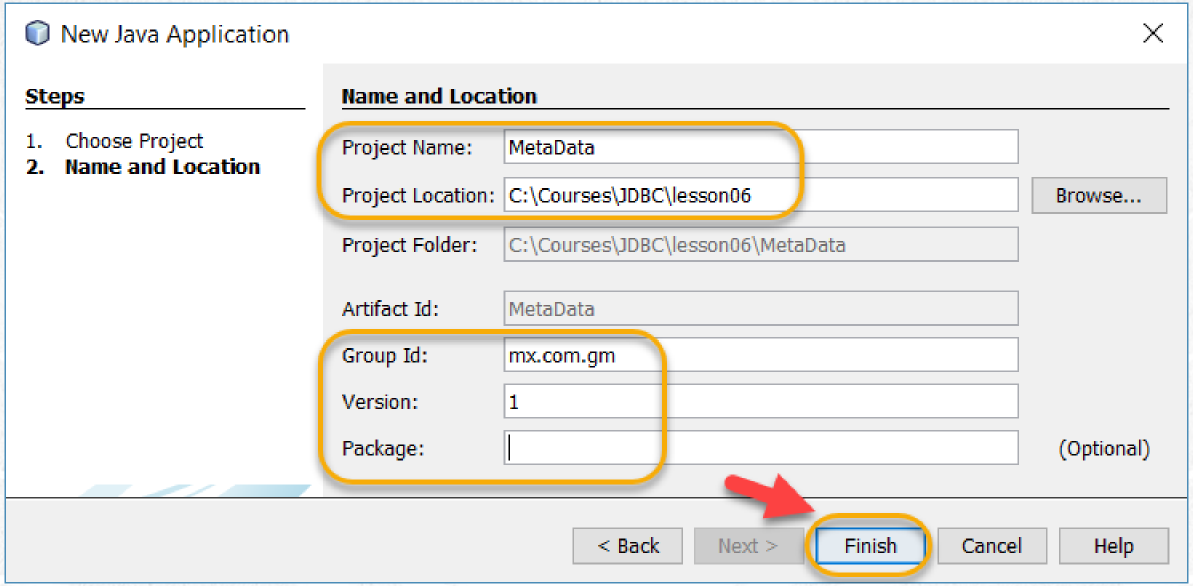
- Java Application
- JavaFX Application
- Web Application
- EJB Module
- Enterprise Application
- Enterprise Application Client
- OSGi Bundle
- NetBeans Module
- NetBeans Application
- POM Project
- Project from Archetype
- Project with Existing POM

Description:

A simple Java SE application using Maven.

1. CREATE A NEW PROJECT

Create a new project:

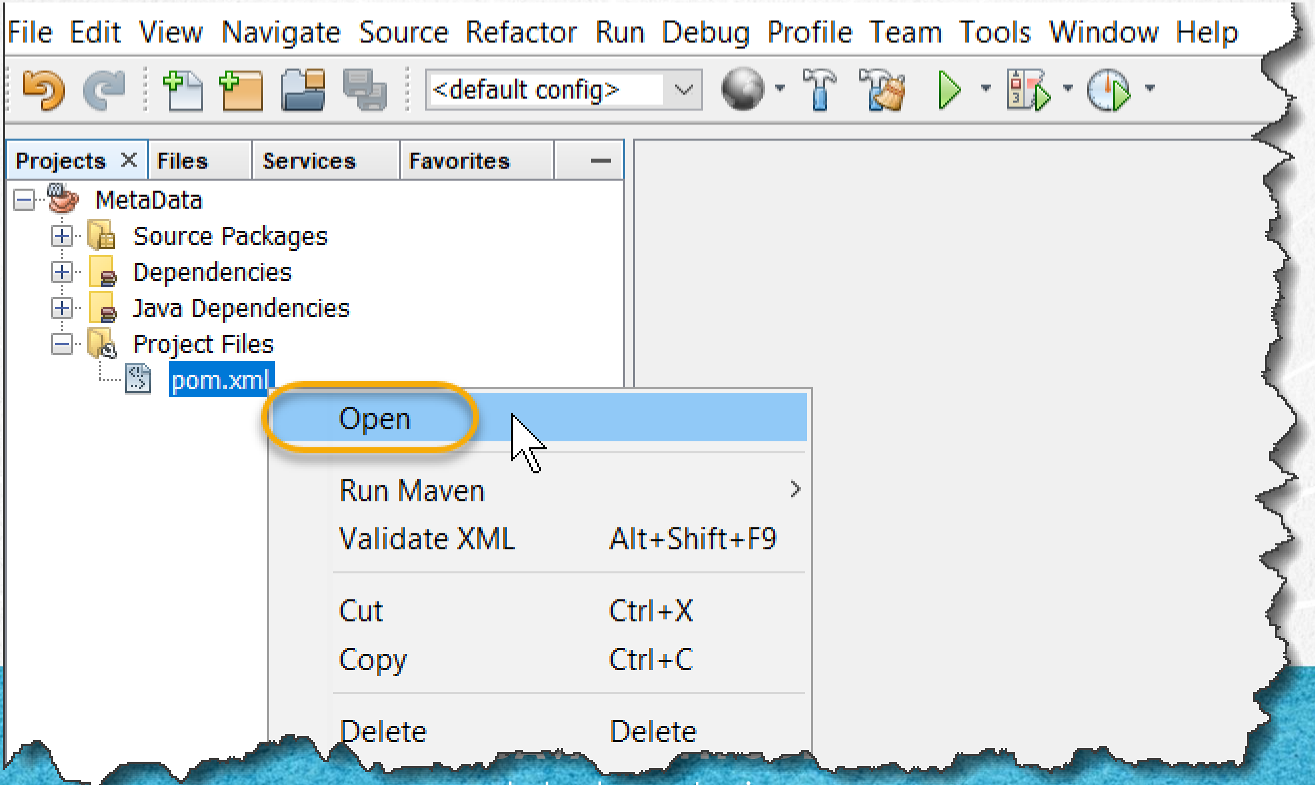


Name and Location

Project Name:	MetaData
Project Location:	C:\Courses\JDBC\lesson06
Project Folder:	C:\Courses\JDBC\lesson06\MetaData
Artifact Id:	MetaData
Group Id:	mx.com.gm
Version:	1
Package:	

2. MODIFY THE POM.XML

Modify the pom.xml to add the mysql.jar:



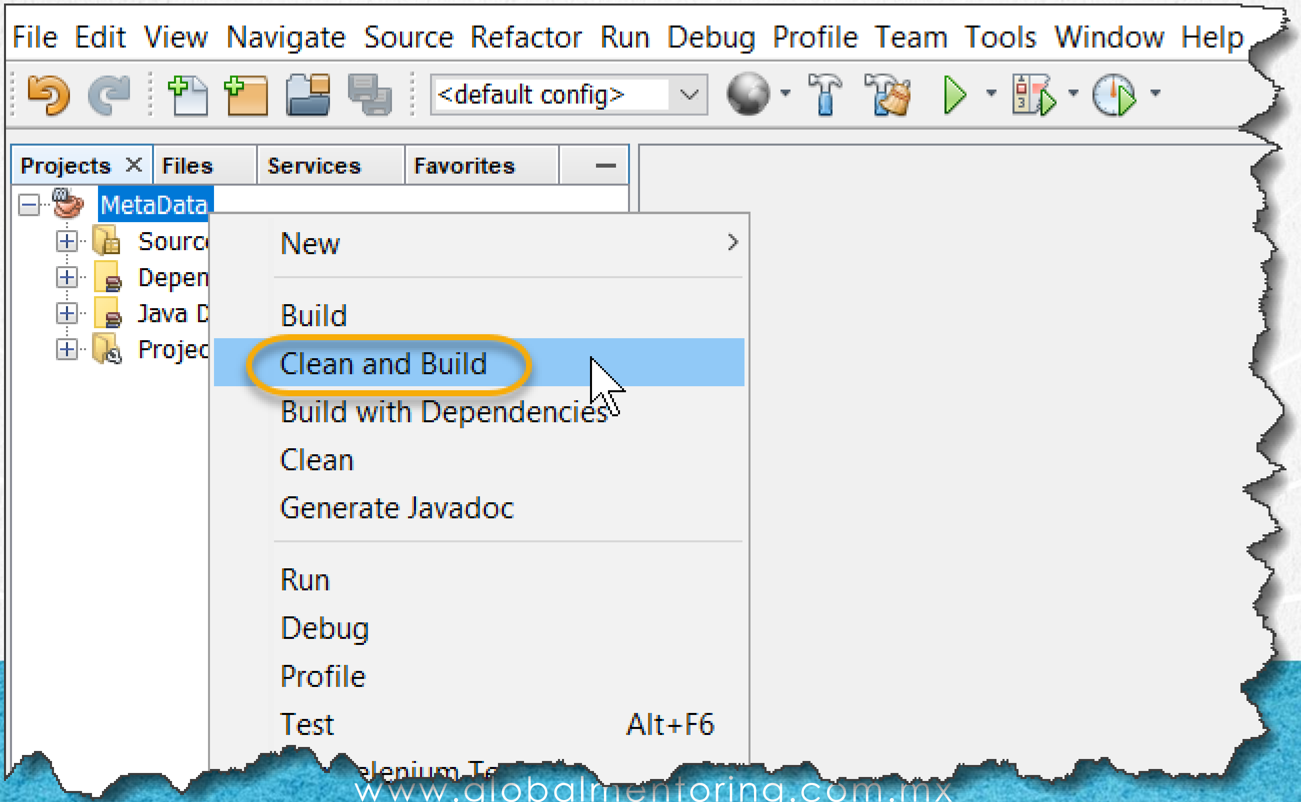
2. MODIFY THE CODE

pom.xml:

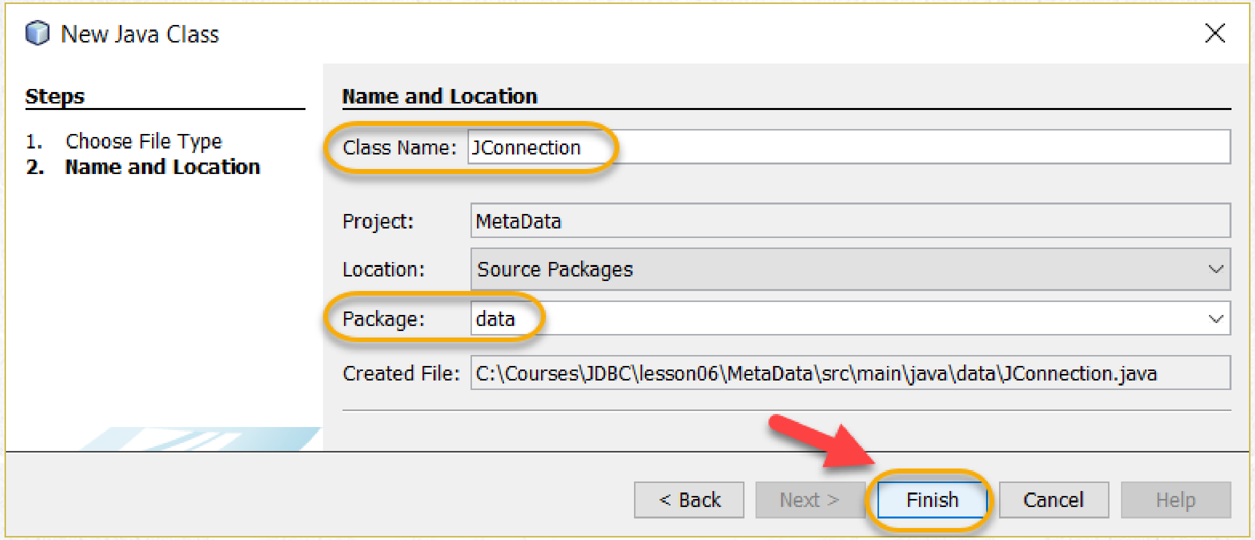
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>mx.com.gm</groupId>
    <artifactId>MetaData</artifactId>
    <version>1</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.46</version>
        </dependency>
    </dependencies>
</project>
```

3. CLEAN & BUILD

Execute the Clean & Build option:



4. CREATE A NEW CLASS


New Java Class

Steps
1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: JConnection

Project: MetaData

Location: Source Packages

Package: data

Created File: C:\Courses\JDBC\lesson06\MetaData\src\main\java\data\JConnection.java

5. MODIFY THE CODE

JConnection.java:

```java
package data;

import java.sql.Connection;
import java.sql.Driver;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class JConnection {

    private static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    private static final String JDBC_URL = "jdbc:mysql://localhost/test?useSSL=false";
    private static final String JDBC_USER = "root";
    private static final String JDBC_PASS = "admin";
    private static Driver driver;
```

5. MODIFY THE CODE

JConnection.java:

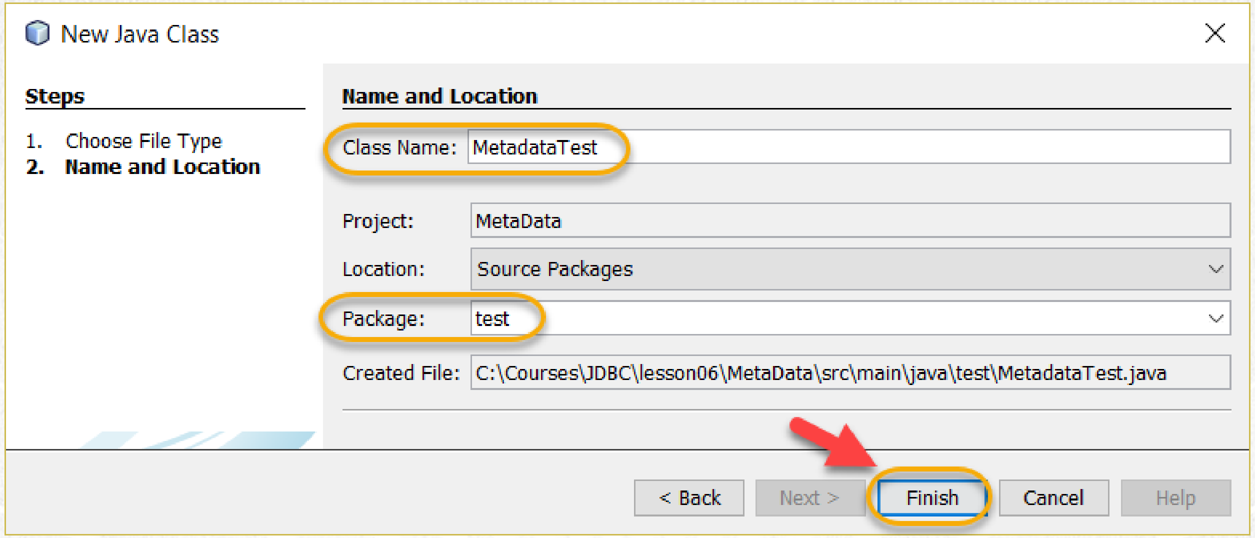
```
    public static synchronized Connection getConnection() throws SQLException {
        if (driver == null) {
            try {
                Class jdbcDriverClass = Class.forName(JDBC_DRIVER);
                driver = (Driver) jdbcDriverClass.newInstance();
                DriverManager.registerDriver(driver);
            } catch (Exception e) {
                System.out.println("Failure to load the JDBC driver");
                e.printStackTrace(System.out);
            }
        }
        return DriverManager.getConnection(JDBC_URL, JDBC_USER, JDBC_PASS);
    }

    //Close the resultSet object
    public static void close(ResultSet rs) {
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException sqle) {
            sqle.printStackTrace(System.out);
        }
    }
```

5. MODIFY THE CODE

JConnection.java:

```
    //Close the PrepareStatement object
    public static void close(PreparedStatement stmt) {
        try {
            if (stmt != null) {
                stmt.close();
            }
        } catch (SQLException sqle) {
            sqle.printStackTrace(System.out);
        }
    }

    //Close the connection object
    public static void close(Connection conn) {
        try {
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException sqle) {
            sqle.printStackTrace(System.out);
        }
    }
}
```

6. CREATE A NEW CLASS



Class Name: MetadataTest

Project: MetaData

Location: Source Packages

Package: test

Created File: C:\Courses\JDBC\lesson06\MetaData\src\main\java\test\MetadataTest.java

7. MODIFY THE CODE

MetadataTest.java:

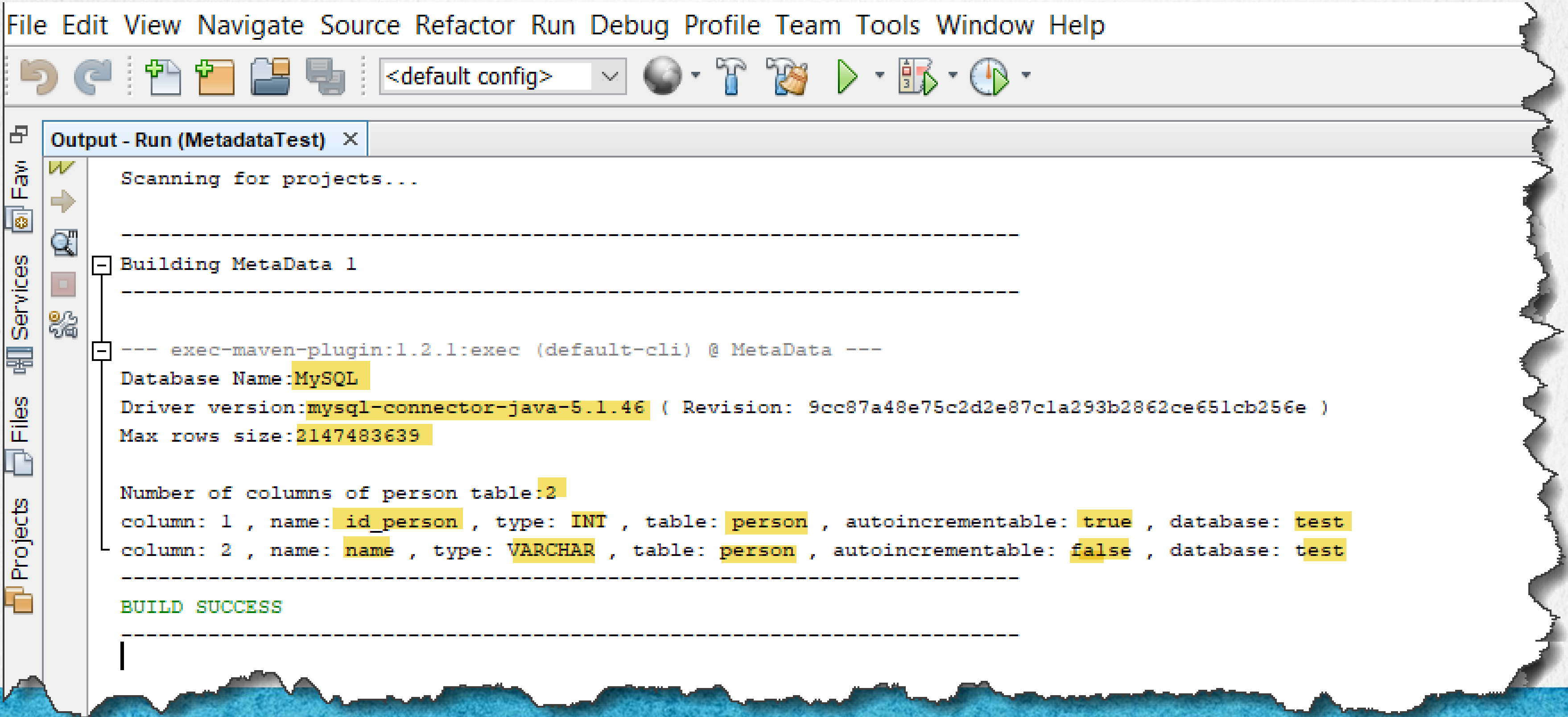
```java
package test;

import data.JConnection;
import java.sql.*;

public class MetadataTest {

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            conn = JConnection.getConnection();

            //General Metadata information
            DatabaseMetaData dmd = conn.getMetaData();
            System.out.println("Database Name:" + dmd.getDatabaseProductName());
            System.out.println("Driver version:" + dmd.getDriverVersion());
            System.out.println("Max rows size:" + dmd.getMaxRowSize());
```

7. MODIFY THE CODE

MetadataTest.java:

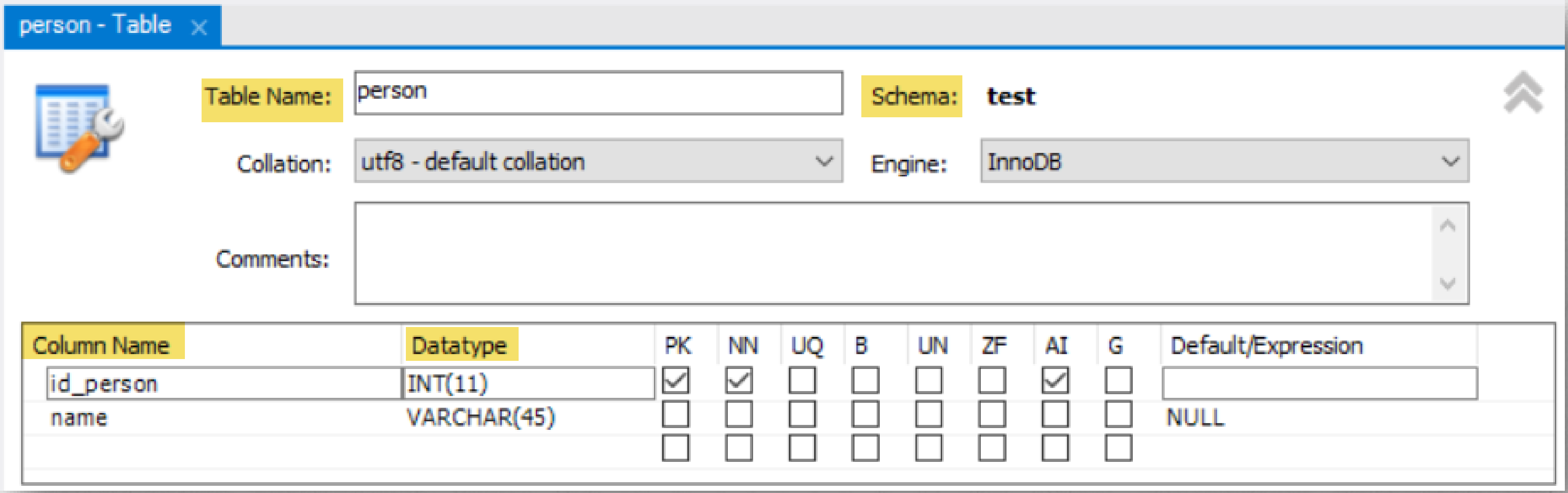
```
            //More specific metadata information from the person table
            stmt = conn.createStatement();
            rs = stmt.executeQuery("SELECT * FROM person");
            //Ask for more specific metadata information
            ResultSetMetaData rsMetaData = rs.getMetaData();
            // We ask how many columns does the person table have
            int numberOfColumns = rsMetaData.getColumnCount();
            //Display the number of columns of person table
            System.out.println("\nNumber of columns of person table:" + numberOfColumns);
            for (int i = 1; i <= numberOfColumns; i++) {
                System.out.print("column: " + i);
                System.out.print(" , name: " + rsMetaData.getColumnName(i));
                System.out.print(" , type: " + rsMetaData.getColumnTypeName(i));
                System.out.print(" , table: " + rsMetaData.getTableName(i));
                System.out.print(" , autoincrementable: " + rsMetaData.isAutoIncrement(i));
                System.out.print(" , database: " + rsMetaData.getCatalogName(i));
                System.out.println("");
            }
        } catch (SQLException e) {
            e.printStackTrace(System.out);
        } finally {
            JConnection.close(conn);
        }
    }
}
```

8. EXECUTE THE PROJECT


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - Run (MetadataTest)
Favi
Services
Files
Projects

```
Scanning for projects...

------------------------------------------------------------------------
Building MetaData 1
------------------------------------------------------------------------

--- exec-maven-plugin:1.2.1:exec (default-cli) @ MetaData ---
Database Name:MySQL
Driver version:mysql-connector-java-5.1.46 ( Revision: 9cc87a48e75c2d2e87c1a293b2862ce651cb256e )
Max rows size:2147483639

Number of columns of person table:2
column: 1 , name: id_person , type: INT , table: person , autoincrementable: true , database: test
column: 2 , name: name , type: VARCHAR , table: person , autoincrementable: false , database: test
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

9. VERIFY THE RESULT



person - Table

Table Name: person

Schema: test

Collation: utf8 - default collation

Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_person	INT(11)	✓	✓					✓		
name	VARCHAR(45)									NULL

EXERCISE CONCLUSION

With this exercise we have put into practice the concept of Metadata using the JDBC API and the MySql database.

The metadata of a database table can help us to know information dynamically from a database table and thus be able to create Java programs that find out this type of information dynamically, for example, when a base table of Data is created dynamically by another process of our system.

ONLINE COURSE

JAVA WITH JDBC

By: Eng. Ubaldo Acosta