Setting up a Regular Java Console Project and JavaFX NetBeans / Eclipse Project for DB Interaction

**Note:** This setup guide assumes that you have already installed Oracle DB 21c Express Edition and configured it according to the instructions in **Appendix A, section A.2 “Installing Oracle Database Express Edition 21c”.**

At the end of this document are instructions for testing out the DB connection in BOTH project types by retrieving data from a table (You can see more instructions in “Chapter 12 of Ezell / Java for IS”.

**Note #2**: Each time you create a new project in NetBeans (for either type) or Eclipse, you’ll have to configure your new project fresh with these settings each time.

# Creating a regular (Maven) console-based Java Database Application:

1. Create a new “Java with Maven” Java Application NetBeans Project.

2. In the “Projects” view, open “Project Files” and then “pom.xml”

3. You’ll notice there is not a dependencies section. You’ll need to add it.  
A screenshot of a computer program

Description automatically generated

4. Add the following right before the end tag for </project>:

<dependency>

<groupId>com.oracle.database.jdbc</groupId>

<artifactId>ojdbc11</artifactId>

<version>21.3.0.0</version>

</dependency>

Here is what it will look like typed in pom.xml:

A screen shot of a computer code

Description automatically generated

If you would like to update the Version number to the latest, you can always get the latest XML above by clicking here <https://mvnrepository.com/artifact/com.oracle.database.jdbc/ojdbc11> and then clicking on the version number you would like to use.

5. Run your project. This will force the NetBeans project to download the OJDBC11 .JAR file needed into your project.

A screenshot of a computer program

Description automatically generated

6. Once it has finished, you should see the .JAR file in your “Dependencies” folder under the Projects view:

A screenshot of a computer

Description automatically generated

7. Now you can add the following imports and class level data fields that will be used:

// Import statements

import java.sql.\*;

import java.util.Scanner;

import oracle.jdbc.pool.\*;

import oracle.jdbc.\*;

import java.util.\*;

// Add these as class-level data fields in your application class

// Class Data Field Level Variables

public static OracleDataSource oDS;

public static Connection jsqlConn;

public static PreparedStatement jsqlStmt;

public static ResultSet jsqlResults;

public static Scanner keyboardIn = new Scanner(System.in);

End result will look like this:

A screenshot of a computer program

Description automatically generated

8. Right click and “Run” your Java application just to make sure no compile errors occur.

9. Now you can add the runDBQuery() method which will make it very easy to interact with the DB. Add this method to your Application as a static application method (“Chapter 6” style):

public static void runDBQuery(String query, char queryType)

{

// queryType - Using the C.R.U.D. acronym

// 'r' - SELECT

// 'c', 'u', or 'd' - UPDATE, INSERT, DELETE

try

{

String URL = "jdbc:oracle:thin:@localhost:1521/XEPDB1";

String user = "javauser"; // From setup instructions

String pass = "javapass"; // From setup instructions

oDS = new OracleDataSource();

oDS.setURL(URL);

jsqlConn = oDS.getConnection(user, pass);

jsqlStmt = jsqlConn.prepareStatement(

query,

ResultSet.TYPE\_SCROLL\_INSENSITIVE,

ResultSet.CONCUR\_READ\_ONLY);

if (queryType == 'r')

jsqlResults = jsqlStmt.executeQuery();

else

jsqlStmt.executeUpdate();

}

catch (SQLException sqlex)

{

System.out.println(sqlex.toString());

}

} // End of runDBQuery() method

This method sends its results up to the jsqlResults object at the data field level, making query results accessible from anywhere in your application class!

Next Page 🡪

# Creating a JavaFX Database Application:

1. Create a new “Java with Maven” NetBeans “Simple JavaFX Maven Archetype (Gluon)” project.

2. Once the project is created, open “pom.xml” under “Project Files” in the “Projects” view in NetBeans

A screen shot of a computer code

Description automatically generated

3. Recommended: Increase the version for JavaFX first. You can find the latest version of the JavaFX Controls dependencies here: <https://mvnrepository.com/artifact/org.openjfx/javafx-controls> Change the version to the latest version (for example, the latest version at the time of this document’s publication was “20-ea+19”) like so:

A screen shot of a computer code

Description automatically generated

Run your project and you will see the .JAR files have updated to the latest versions. Stop your project afterwards.

4. Add the following to your pom.xml as a second dependency tag right below the one for JavaFX:

<dependency>

<groupId>com.oracle.database.jdbc</groupId>

<artifactId>ojdbc11</artifactId>

<version>21.3.0.0</version>

</dependency>

It will look like the following when typed in:

A screenshot of a computer program

Description automatically generated

Run your project again, then stop it once the JavaFX form displays. Confirm that the OJDBC11 Driver has been downloaded:

A screenshot of a computer program

Description automatically generated

5. Add imports and class level datafield objects to interact with the DB:

// Import statements

import java.sql.\*;

import java.util.Scanner;

import oracle.jdbc.pool.\*;

import oracle.jdbc.\*;

import java.util.\*;

// Add these as class-level data fields in your application class

// Class Data Field Level Variables

public static OracleDataSource oDS;

public static Connection jsqlConn;

public static PreparedStatement jsqlStmt;

public static ResultSet jsqlResults;

public static Scanner keyboardIn = new Scanner(System.in);

6. Resolve any errors with the imports. Add the following lines to the module-info.java file, and click “Save All” after editing it:

A screen shot of a computer

Description automatically generated

Run your project again after editing just to make sure no compile errors occur.

7. Add the runDBQuery() method to your JavaFX application as a static application method:

public static void runDBQuery(String query, char queryType)

{

// queryType - Using the C.R.U.D. acronym

// 'r' - SELECT

// 'c', 'u', or 'd' - UPDATE, INSERT, DELETE

try

{

String URL = "jdbc:oracle:thin:@localhost:1521/XEPDB1";

String user = "javauser";

String pass = "javapass";

oDS = new OracleDataSource();

oDS.setURL(URL);

jsqlConn = oDS.getConnection(user, pass);

jsqlStmt = jsqlConn.prepareStatement(

query,

ResultSet.TYPE\_SCROLL\_INSENSITIVE,

ResultSet.CONCUR\_READ\_ONLY);

if (queryType == 'r')

jsqlResults = jsqlStmt.executeQuery();

else

jsqlStmt.executeUpdate();

}

catch (SQLException sqlex)

{

System.out.println(sqlex.toString());

}

} // End of runDBQuery() method

# Testing your Java 🡨🡪 DB Connection for BOTH project types:

Once you have the method added, you can create tables with data in your Oracle DB, then send a query like so (here a table called CUSTOMER exists with data in the database):

try

{

runDBQuery("SELECT \* FROM JAVAUSER.ITEM", 'r');

while (jsqlResults.next())

{

System.out.println(

jsqlResults.getString(1)

+ " "

+ jsqlResults.getString(2));

}

}

catch (Exception ex)

{

ex.printStackTrace();

}

System.out.println("");

The results print to the console:

A white screen with black text

Description automatically generated

# Instructions for an Eclipse IDE Java Maven Console Project:

1. Click “File” -> “New” -> “Project. . .”
2. In the “New Maven Project” dialog box, click “Browse…” to choose a location OR use the default workspace location if you prefer. Click “Next >”.
3. In the “Select an Archetype” dialog box, in the “Filter” text field type org.apache.maven.archetypes to filter the list. Click once to select the “maven-archetype-quickstart” archetype and then click “Next >”.

A screenshot of a computer program

Description automatically generated

1. In the final dialog box on the next screen
   1. For “Group ID:” type com.mycompany (if it is not already present).
   2. For “Artifact ID:” type the desired name for your Maven project.

A screenshot of a computer program

Description automatically generated

* 1. Click “Finish” to finish the setup process.

1. On the “Console”, you may see a prompt of “Y: :”. Place your cursor to the right of this prompt and hit the Enter key to complete the automated setup. You will see more downloading activity after answering this prompt.

A screenshot of a computer

Description automatically generated

1. Ensure the “Project Explorer” window is open (Window -> Show View -> Project Explorer) and find the file “pom.xml” at the bottom. Double-click to open this file.
2. Scroll down in the pom.xlm listing until you find the <dependencies> section (if there is not one, you may have to add it.
3. The Maven Repository Website <https://mvnrepository.com> provides a searchable database for Maven dependency XML. Adding the dependency XML to your project’s pom.xml file will have your project download the appropriate .jar files for 3rd party functionality you wish to add to your Java project.

A screenshot of a computer

Description automatically generated

1. A search for “OJBDC11” yields the following XML:

<!-- https://mvnrepository.com/artifact/com.oracle.database.jdbc/ojdbc11 -->

<dependency>

<groupId>com.oracle.database.jdbc</groupId>

<artifactId>ojdbc11</artifactId>

<version>23.3.0.23.09</version>

</dependency>

1. Notice that it is a singular <dependency> tag. Add this to your pom.xml listing like so:

A screen shot of a computer program

Description automatically generated

1. Click the “Save All” button or use “File” -> “Save All” to save the file. This will force the project to download any .jar files associated with the OJDBC11 dependency you have required. Note in the “Dependencies” section of the Project Explorer that the .jar file has been downloaded:

A screenshot of a computer program

Description automatically generated

1. Now you are ready to follow the steps in the textbook to have your Java code interact with your DB!

# Instructions for an Eclipse IDE **JavaFX** Maven Console Project:

1. Follow the steps in the document “**Appendix A Supplemental Creating a JavaFX Project.docx**” to create JavaFX project in eclipse and update the JavaFX controls version.
2. In the “pom.xml” file you use to update JavaFX to the latest version, add the following as an additional <dependency> section, from the Maven Repository (discussed in the “Console” steps above):

<!-- https://mvnrepository.com/artifact/com.oracle.database.jdbc/ojdbc11 -->

<dependency>

<groupId>com.oracle.database.jdbc</groupId>

<artifactId>ojdbc11</artifactId>

<version>23.3.0.23.09</version>

</dependency>

1. Click the “Save All” Button or use File -> Save All to have your JavaFX project download the associated OJDBC11 .jar file.

A screen shot of a computer program

Description automatically generated

1. You are ready to have your JavaFX project interact with your Oracle DB!

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

NetBeans IDE and NetBeans Platform are based on Apache NetBeans from The Apache Software Foundation and are licensed under Apache License Version 2.0. For more information, please visit <https://netbeans.apache.org/>. Copyright 2017 – 2024 The Apache Software Foundation. Apache, Apache NetBeans, the Apache feather logo and the Apache NetBeans logo are trademarks of [The Apache Software Foundation](https://www.apache.org/).

Eclipse® and the Eclipse logo are trademarks of Eclipse Foundation, Inc. © Copyright Eclipse Contributors and others 2000, 2024. The Eclipse project includes software developed by other open-source projects including the Apache Software Foundation, <https://www.apache.org/>.