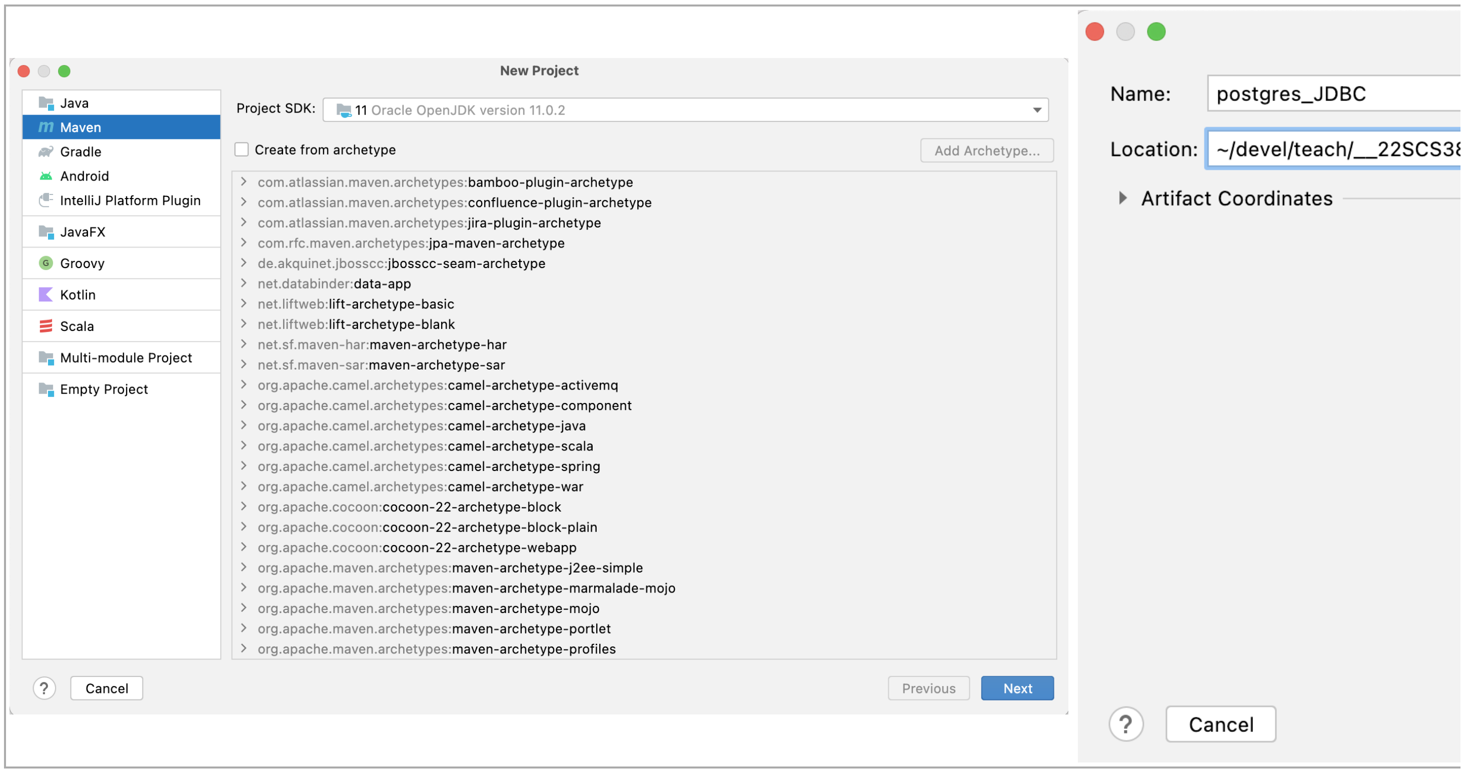
CS 3810: PostgreSQL JDBC Driver

This how-to illustrates how to connect and run SQL queries in Java using the **PostgreSQL JDBC Driver (https://jdbc.postgresql.org/)**. It uses IntelliJ IDE and Maven. The driver is basically a bridge between Java JDBC (Java Database Connectivity) API and a (native) PostgreSQL database driver. The JDBC API documentation is available **here (https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html)**. **Maven (https://maven.apache.org/)** is a software management tool that takes care of issues such as compiling, testing, documenting, packaging, module dependency troubleshooting, among other things. There is no need to install Maven because IntelliJ comes with it.

Step 1: Create Maven Project in IntelliJ



Project SDK: 11 Oracle OpenJDK version 11.0.2

Create from archetype

com.atlassian.maven.archetypes:bamboo-plugin-archetype
com.atlassian.maven.archetypes:confluence-plugin-archetype
com.atlassian.maven.archetypes:jira-plugin-archetype
com.rfc.maven.archetypes:jpa-maven-archetype
de.akquinet.jbosscc:jbosscc-seam-archetype
net.databinder:data-app
net.liftweb:lift-archetype-basic
net.liftweb:lift-archetype-blank
net.sf.maven-har:maven-archetype-har
net.sf.maven-sar:maven-archetype-sar
org.apache.camel.archetypes:camel-archetype-activemq
org.apache.camel.archetypes:camel-archetype-component
org.apache.camel.archetypes:camel-archetype-java
org.apache.camel.archetypes:camel-archetype-scala
org.apache.camel.archetypes:camel-archetype-spring
org.apache.camel.archetypes:camel-archetype-war
org.apache.cocoon:cocoon-22-archetype-block
org.apache.cocoon:cocoon-22-archetype-block-plain
org.apache.cocoon:cocoon-22-archetype-webapp
org.apache.maven.archetypes:maven-archetype-j2ee-simple
org.apache.maven.archetypes:maven-archetype-marmalade-mojo
org.apache.maven.archetypes:maven-archetype-mojo
org.apache.maven.archetypes:maven-archetype-portlet
org.apache.maven.archetypes:maven-archetype-profiles

Name: postgres_JDBC

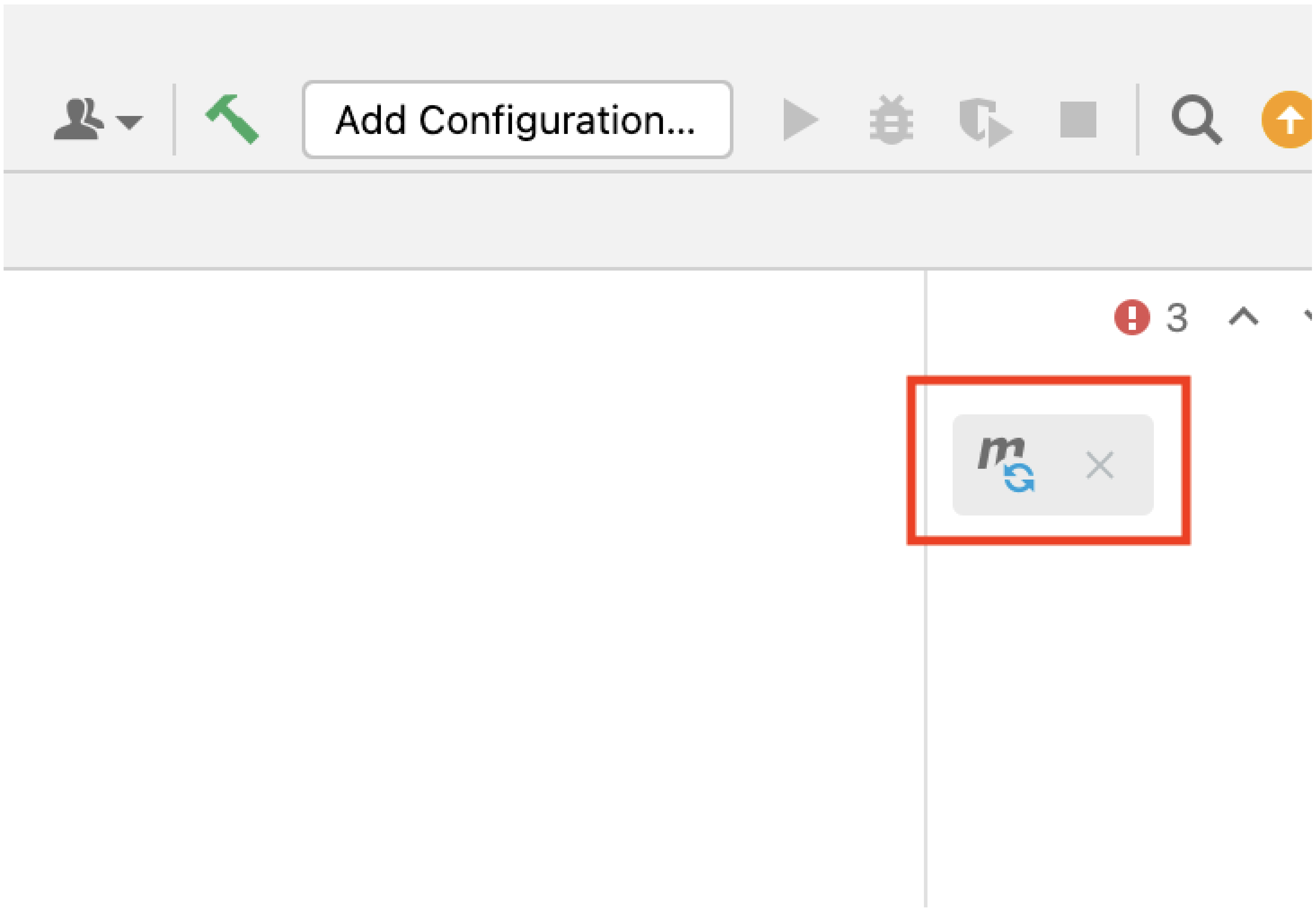
Location: ~/devel/teach/__22SCS38

Step 2: Add the PostgreSQL JDBC Driver Dependency

A Maven project has a special XML file named "pom.xml." POM stands for Project Object Model and it contains information about the project and configuration details used by Maven to build the project. Add the following "dependencies" tag to instruct Maven that our project needs the postgreSQL JDBC driver to be able to be built. Make sure you add the tag into the "project" tag.

```
    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>42.3.3</version>
        </dependency>
    </dependencies>
```

Click on the "Load Maven Changes" tiny button that appears on the upper-right corner. Wait for the library to be dowloaded and incorporated into your project.



Step 3: Connect to a Database

Create a new class named "HelloPostgres" and type in the following code. Make sure you have database "hr" created (use psql program to verify that).

```
import java.sql.Connection;
import java.sql.DriverManager;

public class HelloPostgres {

    public static void main(String[] args) throws Exception {
        String server = "localhost";
        String database = "hr";
        String user = "hr_admin";

        // you should NEVER hard-code passwords like shown in this example in a production environment!!!
        String password = "135791";
        String connectURL = "jdbc:postgresql://" + server + "/" + database + "?user=" + user + "&password
=" + password;
```

```
        // connects to the database
        Connection conn = DriverManager.getConnection(connectURL);
        System.out.println("Connection to Postgres database " + database + " was successful!");

        // closes the connection
        System.out.println("Bye!");
        conn.close();
    }
}
```

Run the program and verify that you are able to connect to postgres. If everything works you should get a message saying:

```
Connection to Postgres database hr was successful!
Bye!
```

Step 4: Run a Simple Query

Add the following lines of code right-after the connection but before the call to close.

```
        // run a simple query
        Statement stmt = conn.createStatement();
        String sql = "SELECT id, name FROM employees";
        ResultSet resultSet = stmt.executeQuery(sql);

        // navigate through the rows of the result set
        while (resultSet.next()) {
            int id = resultSet.getInt("id");
            String name = resultSet.getString("name");
            System.out.println(id + ", " + name);
        }
```

You will have to add the following import statements as well:

```
import java.sql.ResultSet;
import java.sql.Statement;
```

Run your code again. You should be able to see the rows from table Employees.