SWEN90007 - Software Design and Architecture

Semester 2 – 2020 – Additional Learning Resource for Students

Data Source Layer

This how-to document will help you:

- To work with Database engine, and to connect your application to a DB server.
- To create databases, tables, and populate tables.

Introduction:

PostgreSQL, also known as **Postgres**, is a free and open-source relational database management system (RDBMS) emphasizing extensibility and technical standards compliance. It is designed to handle a range of workloads, from single machines to data warehouses or Web services with many concurrent users. https://www.postgresql.org/download/

What is JDBC?

JDBC stands for **Java Database Connectivity**, which is a standard Java API for database independent connectivity between the Java programming runtime and a wide range of databases.

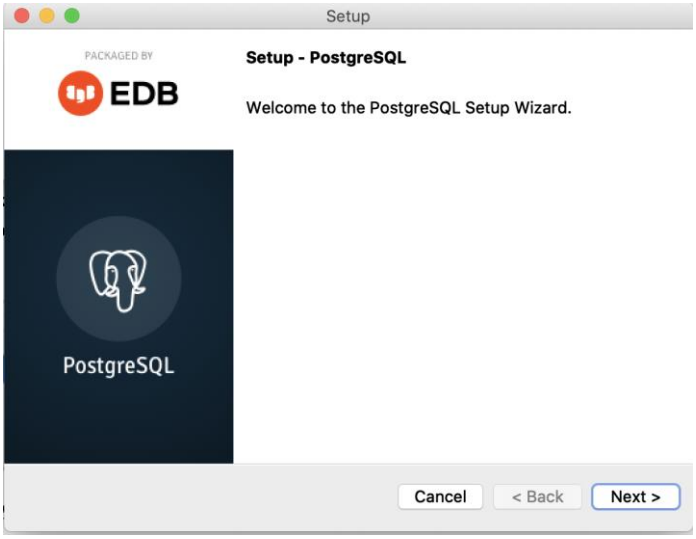
The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

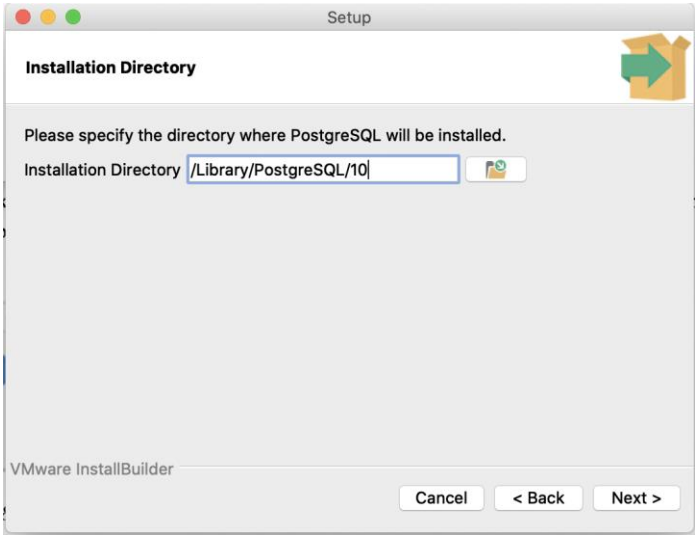
- Making a connection to a database.
- Creating SQL statements.
- Executing SQL queries in the database.
- Viewing & Modifying the resulting records.

Configurations:

1. Installing database server

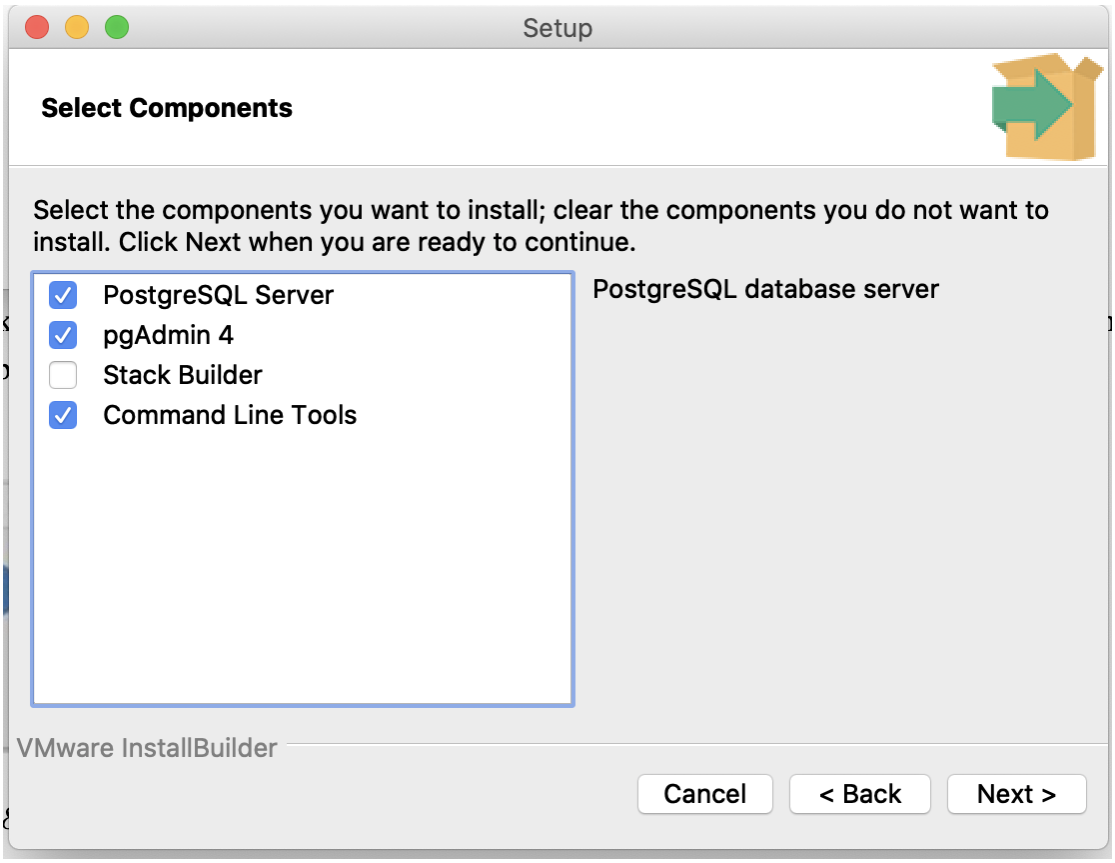
- Download the installer PostgreSQL Database [version 10.14]: https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
- **Note**: PostgreSQL can also be installed on macOS using **Homebrew**
- Step 1. Double click on the installer file, an installation wizard will appear and guide you through multiple steps where you can choose different options that you would like to have in PostgreSQL.
- Step 2. Click the Next button



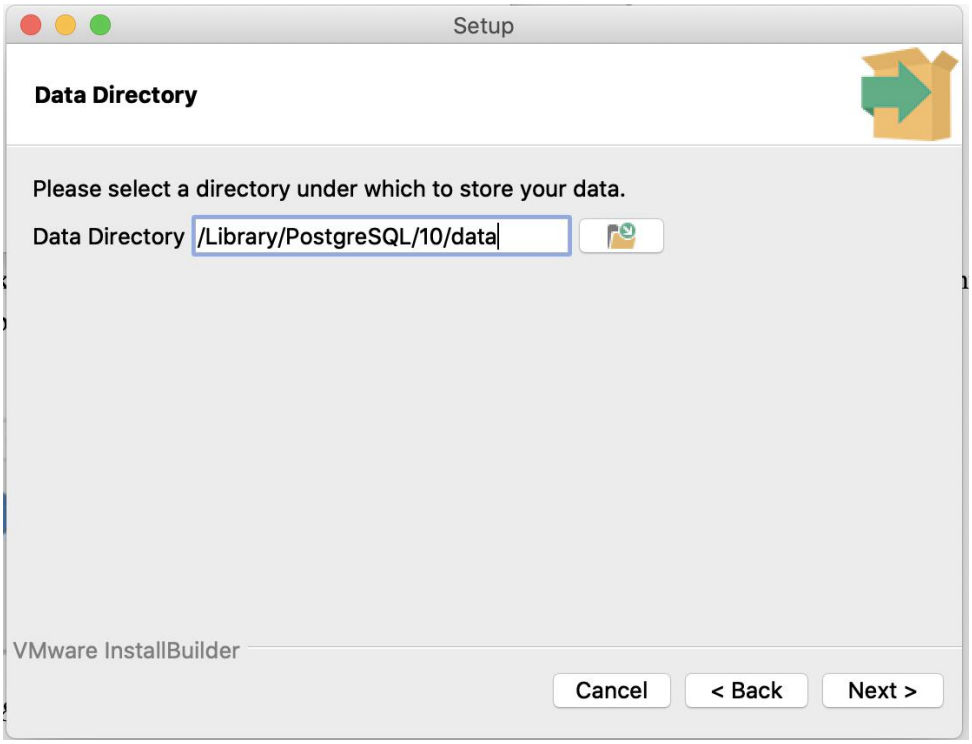


- Step 3. Select software components to install:
  - The PostgreSQL Server to install the PostgreSQL database server
  - pgAdmin 4 to install the PostgreSQL database GUI management tool.
  - Command Line Tools to install command-line tools such as psql, pg_restore, etc. These tools allow you to interact with the PostgreSQL database server using the command-line interface.
  - Stack Builder provides a GUI that allows you to download and install drivers that work with PostgreSQL.

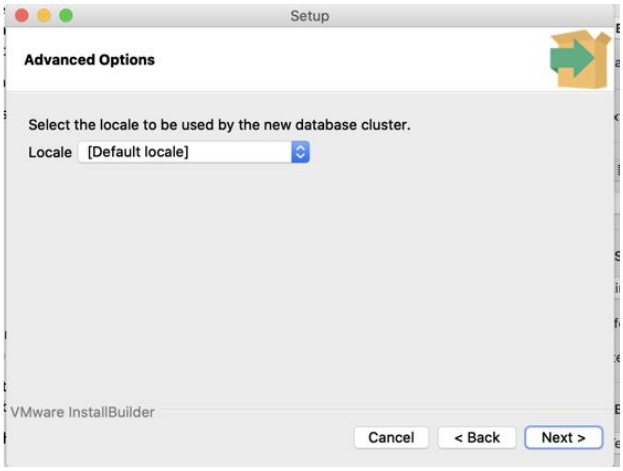
For the tutorial, you don’t need to install Stack Builder so feel free to uncheck it and click the Next button to select the data directory:

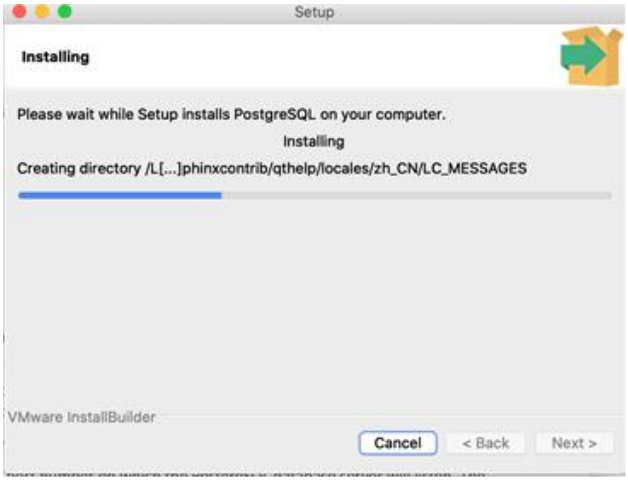


- Step 4. Select the database directory to store the data or accept the default folder. And click the Next button to go to the next step:



- Step 5. Enter the password for the database superuser (*postgres*)
- Step 6. Enter a port number on which the PostgreSQL database server will listen. The default port of PostgreSQL is 5433. Then, continue to the end of installation.





Congratulation! you've successfully installed PostgreSQL database server on your local system.

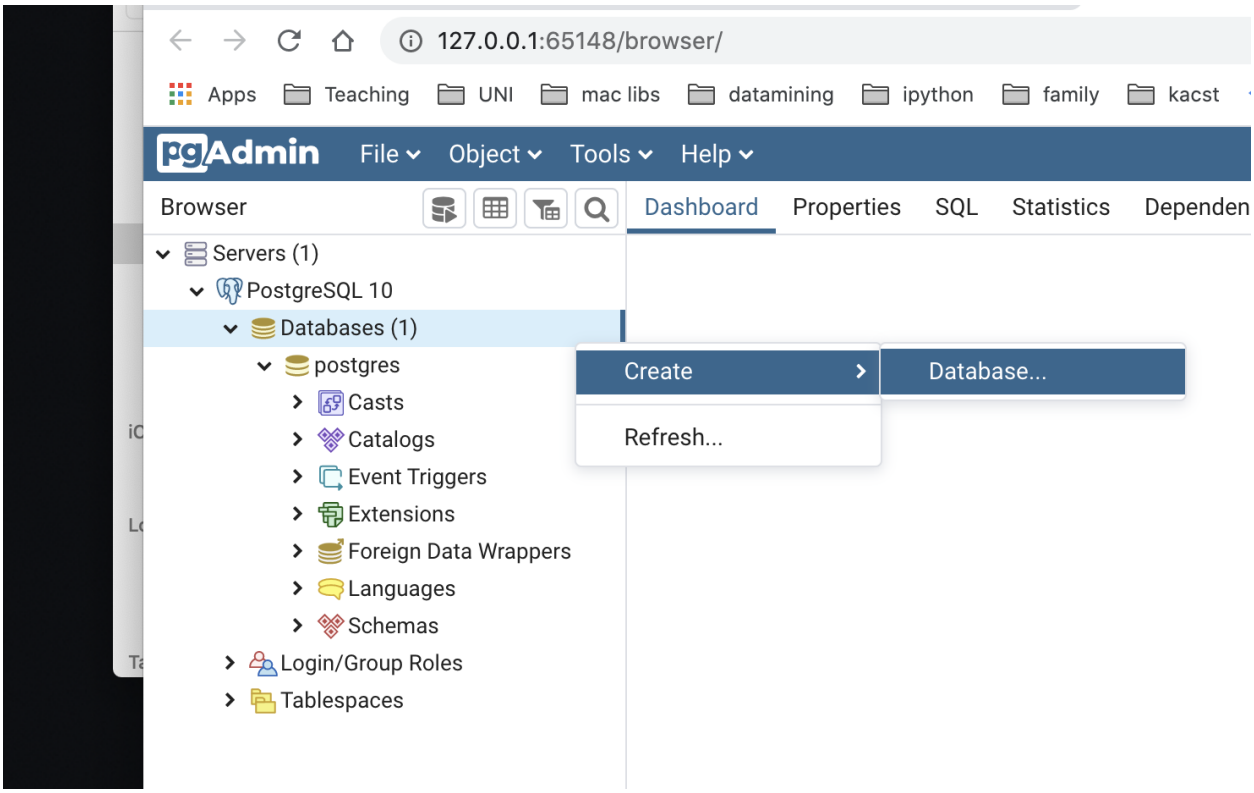
Note: when you finish your project and deploy it to Heroku, you do not need the local DB server. You are going to need to connect your application to the server on Heroku.

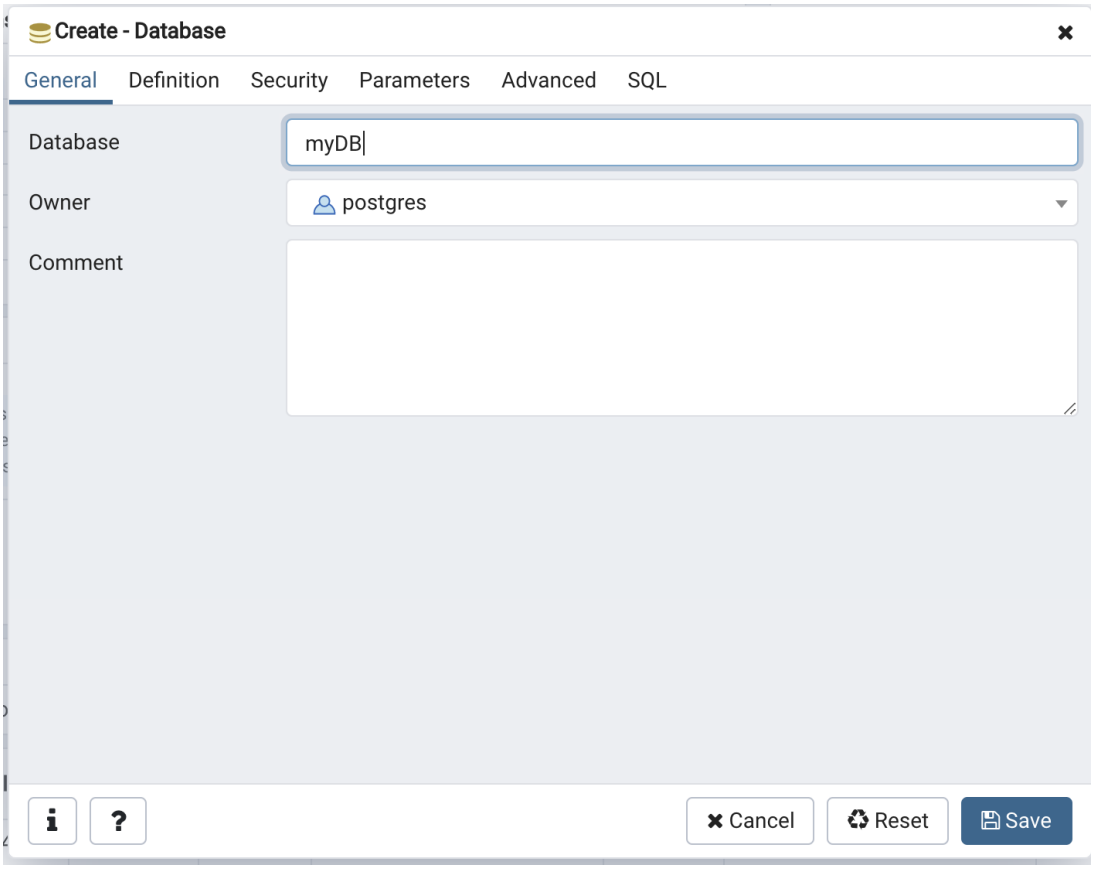
2. Create Database in PostgresSQL server

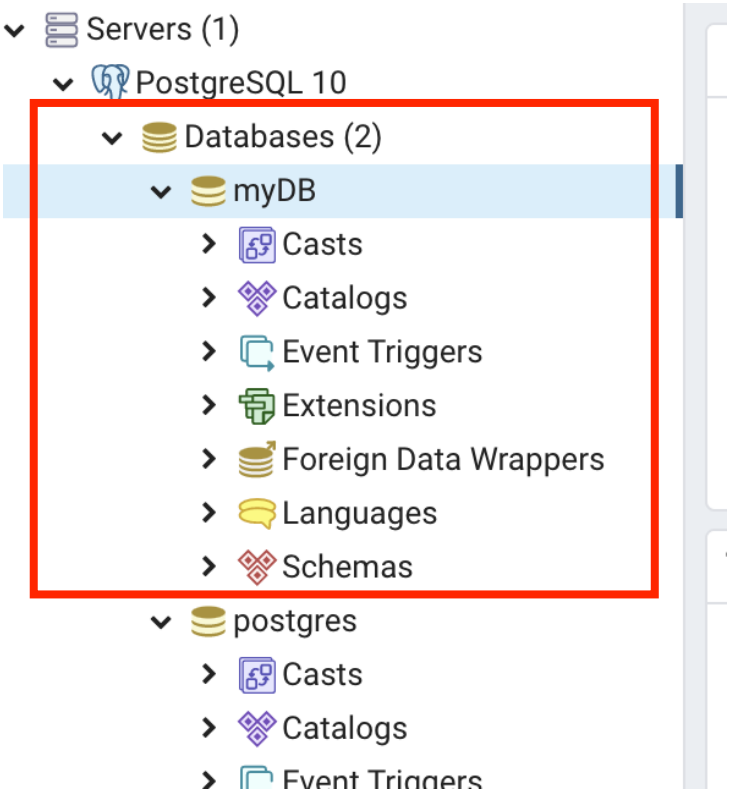
There are several ways to create your database on the PostgreSQL server from any client application e.g., ***psql*** and ***pgAdmin***.

Using pgAdmin: The pgAdmin application allows you to interact with the PostgreSQL database server via an intuitive user interface.

- First, launch the **pgAdmin** application.
- The pgAdmin application will launch on the web browser,
- Right-click on Databases => create => Database; as shown in the following picture:

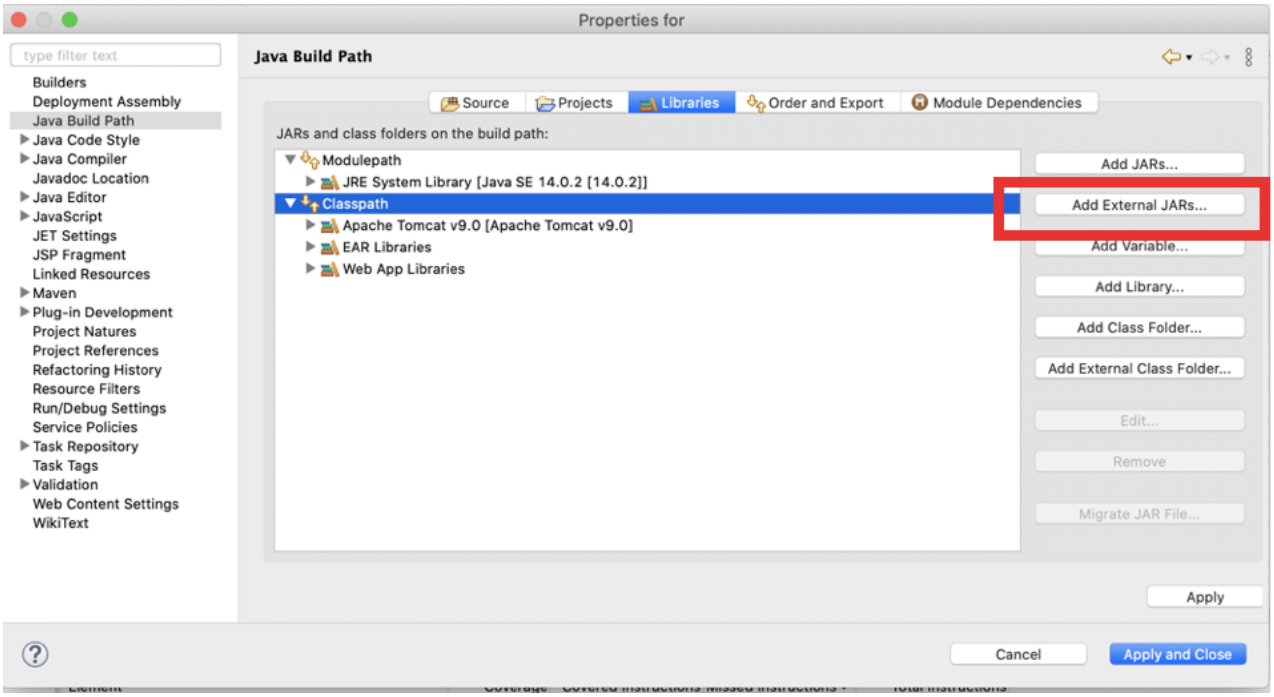


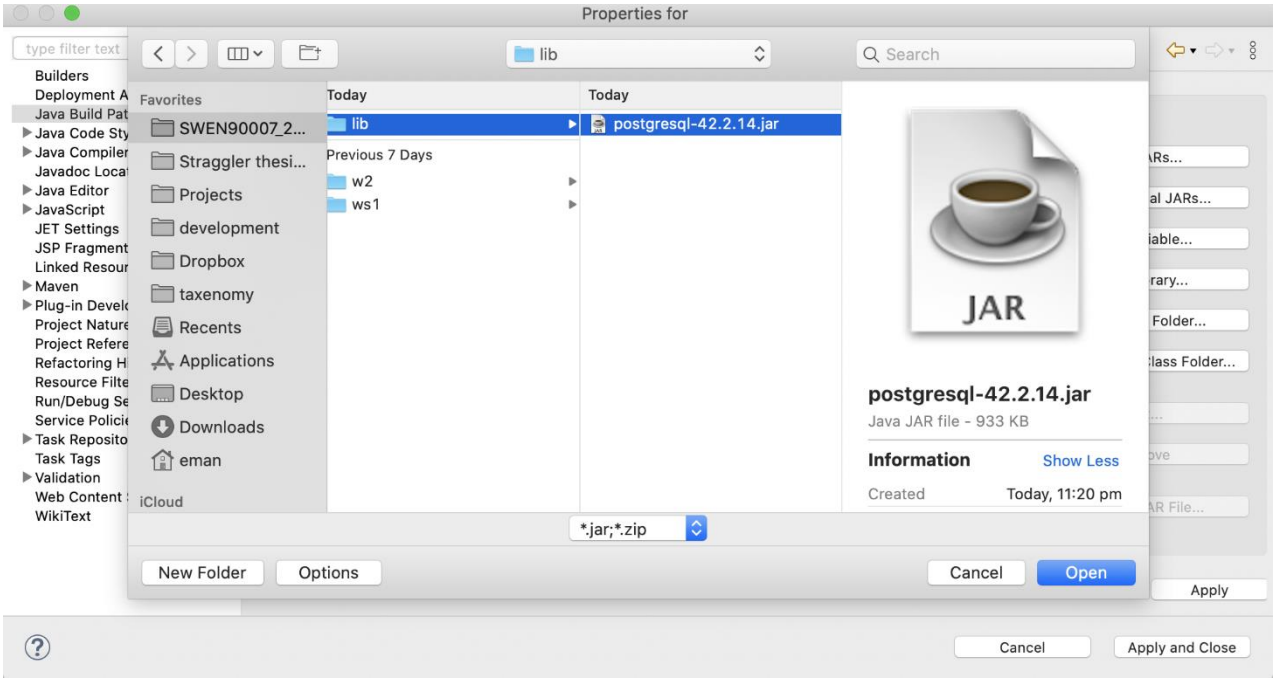

Create - Database
General
Definition
Security
Parameters
Advanced
SQL
Database
myDB
Owner
postgres
Comment
Cancel
Reset
Save

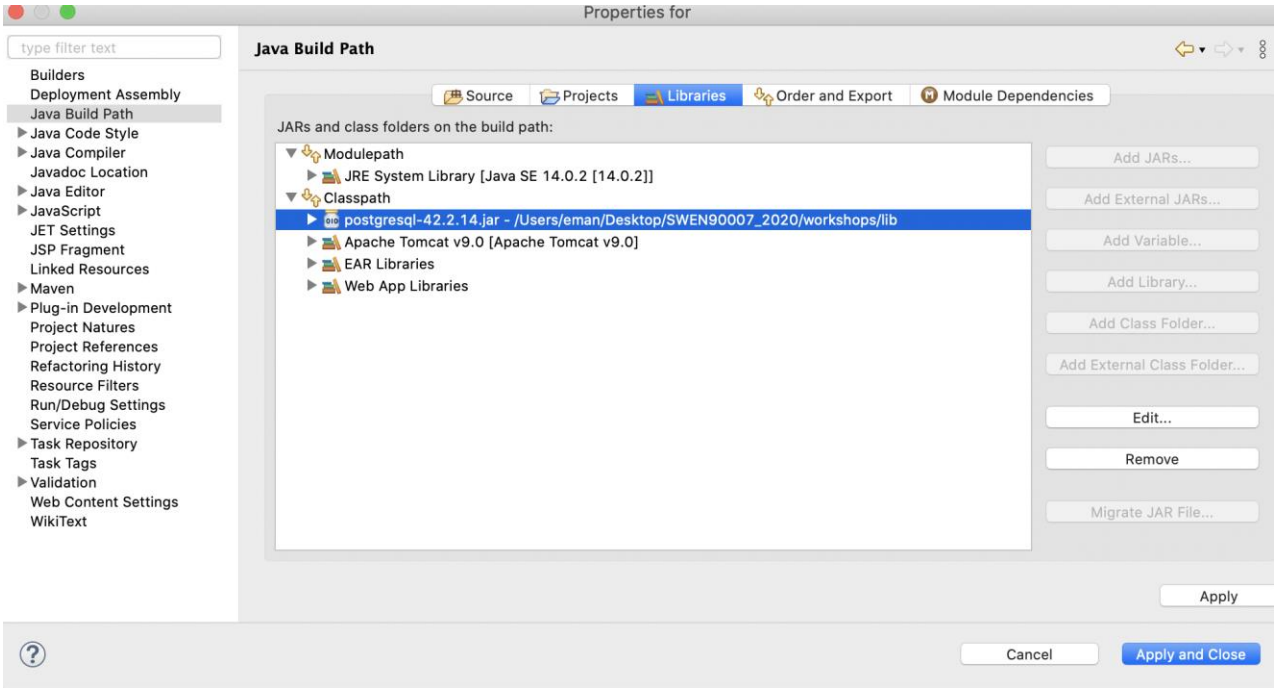

Servers (1)
PostgreSQL 10
Databases (2)
myDB
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas
postgres
Casts
Catalogs
Event Triggers

3. Connecting to Postgres Database server with Eclipse

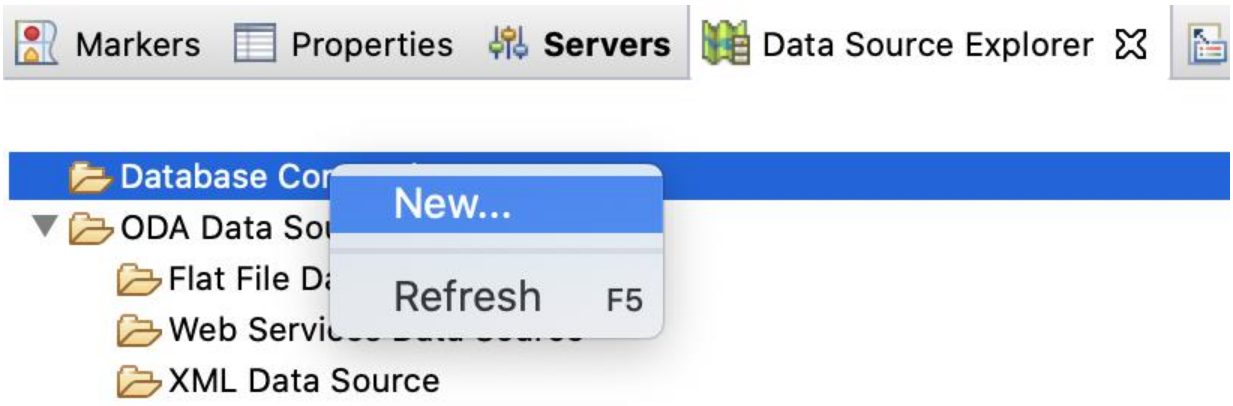
- First, download the Postgres JDBC driver: https://jdbc.postgresql.org/download/postgresql-42.2.14.jar
- In Eclipse, add downloaded jar file “postgresql-(VERSION).jdbc.jar” in your class path, or you can use it along with -classpath option; as the following:



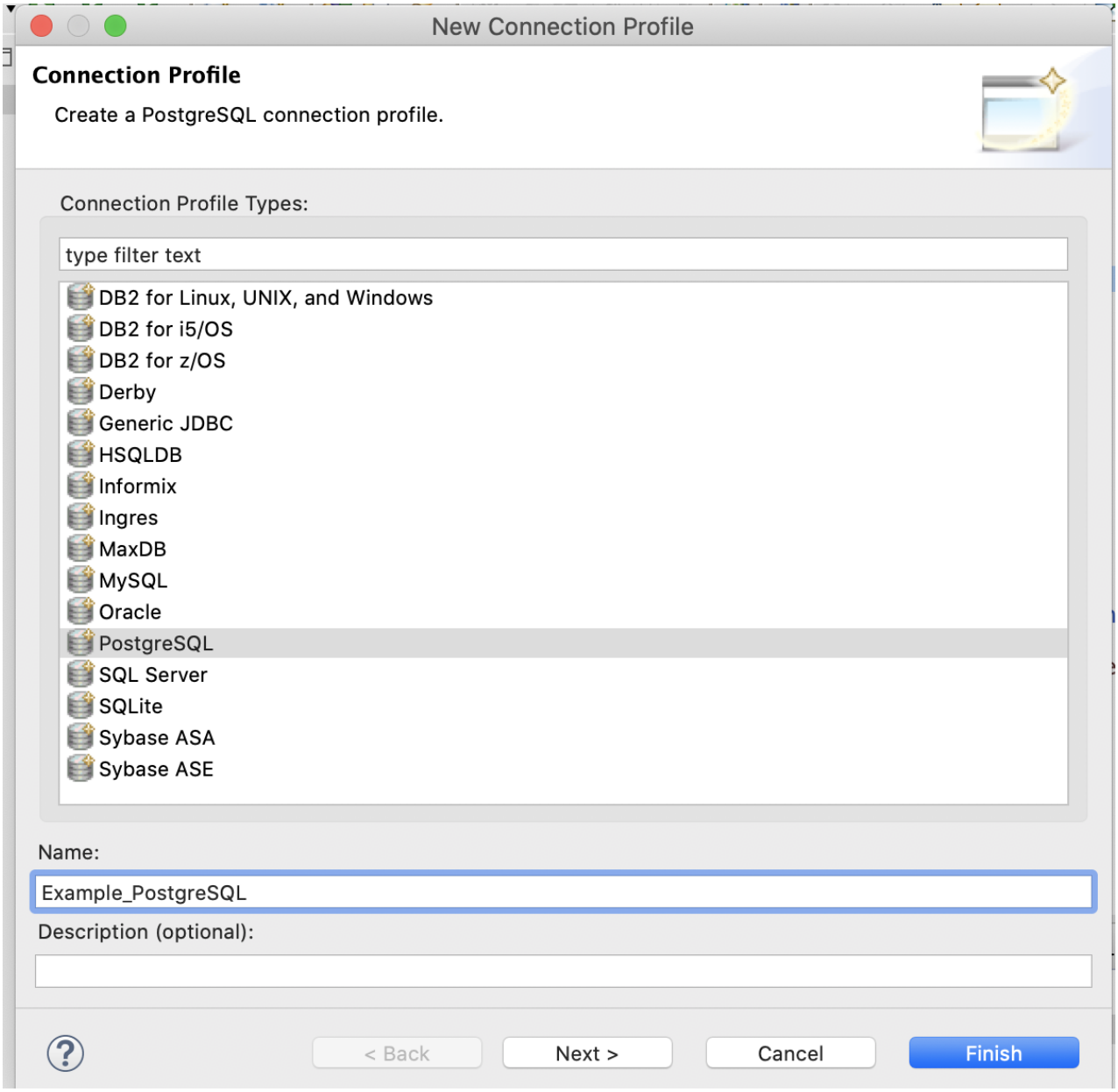


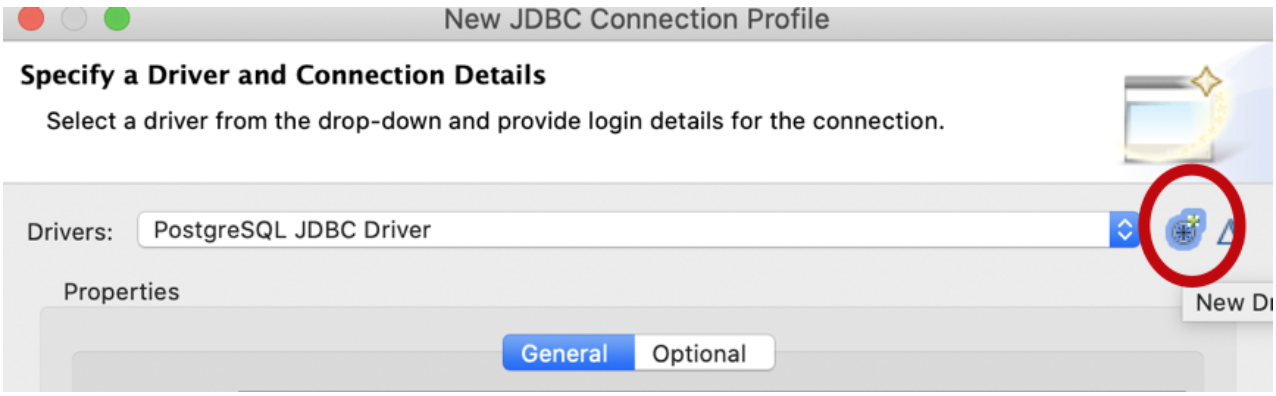


- Change the Eclipse perspective to ”**Database Development Perspective**” ;
- Create a new database connection;
- Right click on “Database Connections” and choose New

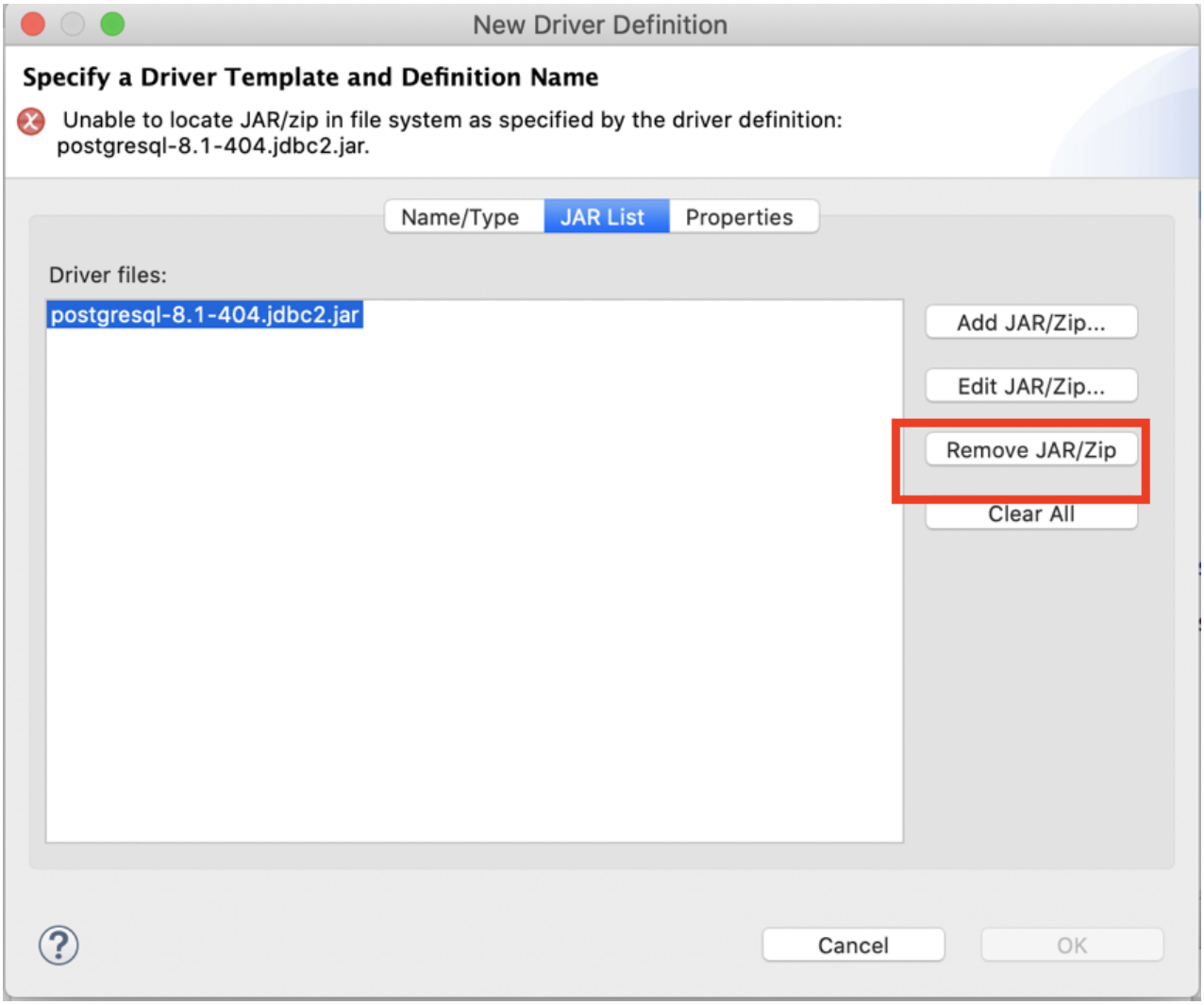


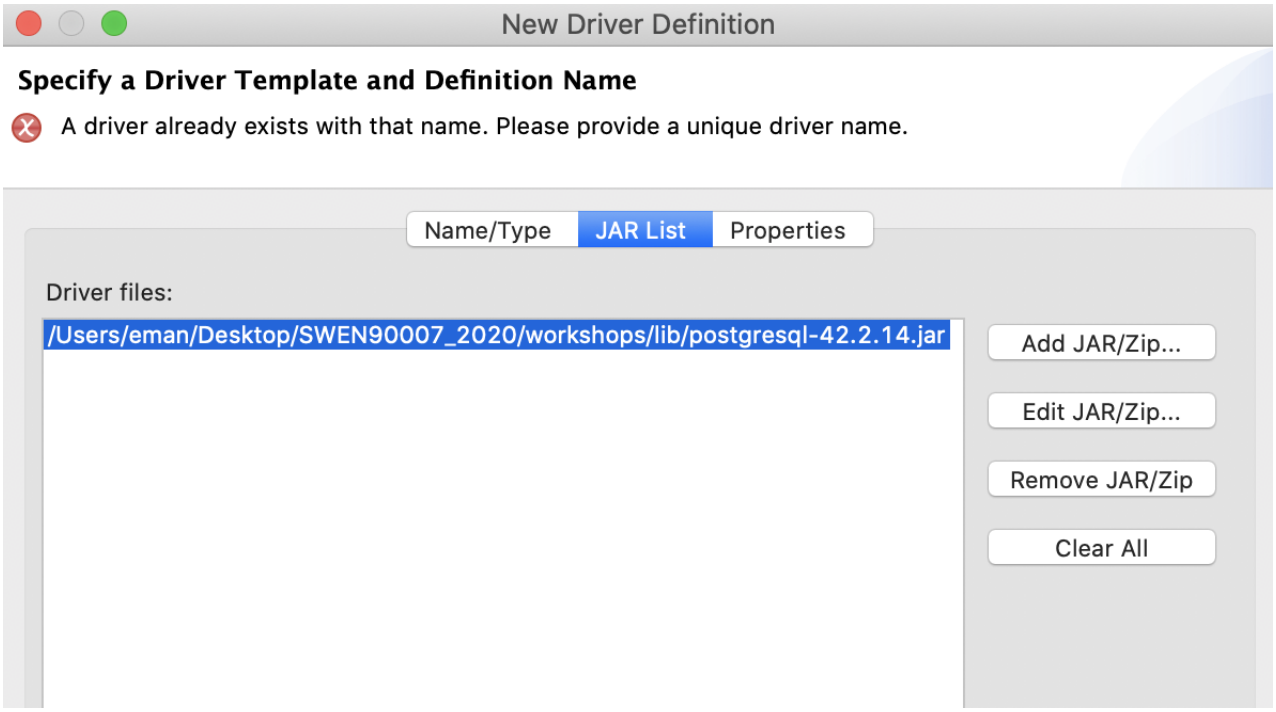
- Select the ***PostgreSQL*** database and give a name to the connection; Choose a new driver definition and choose "**PostgreSQL**


New Connection Profile
Connection Profile
Create a PostgreSQL connection profile.
Connection Profile Types:
type filter text
DB2 for Linux, UNIX, and Windows
DB2 for i5/OS
DB2 for z/OS
Derby
Generic JDBC
HSQLDB
Informix
Ingres
MaxDB
MySQL
Oracle
PostgreSQL
SQL Server
SQLite
Sybase ASA
Sybase ASE
Name:
Example_PostgreSQL
Description (optional):
< Back
Next >
Cancel
Finish

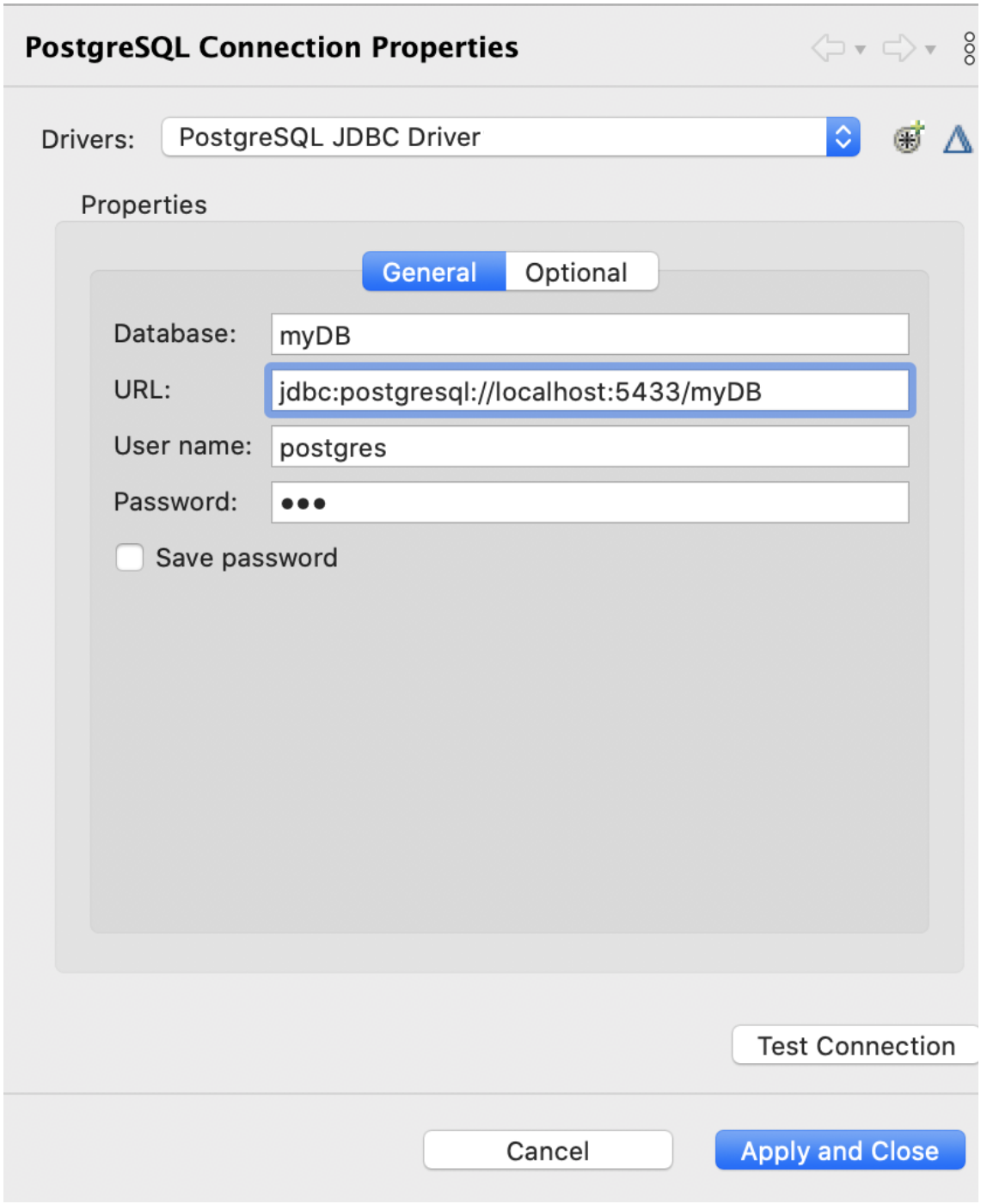

New JDBC Connection Profile
Specify a Driver and Connection Details
Select a driver from the drop-down and provide login details for the connection.
Drivers:
PostgreSQL JDBC Driver
Properties
General
Optional

- Select Drivers for the connection:
    - Click on the drivers
    - Select JAR List, remove old jar
    - Add JAR
    - Select the JAR file (downloaded previously)
    - Click OK



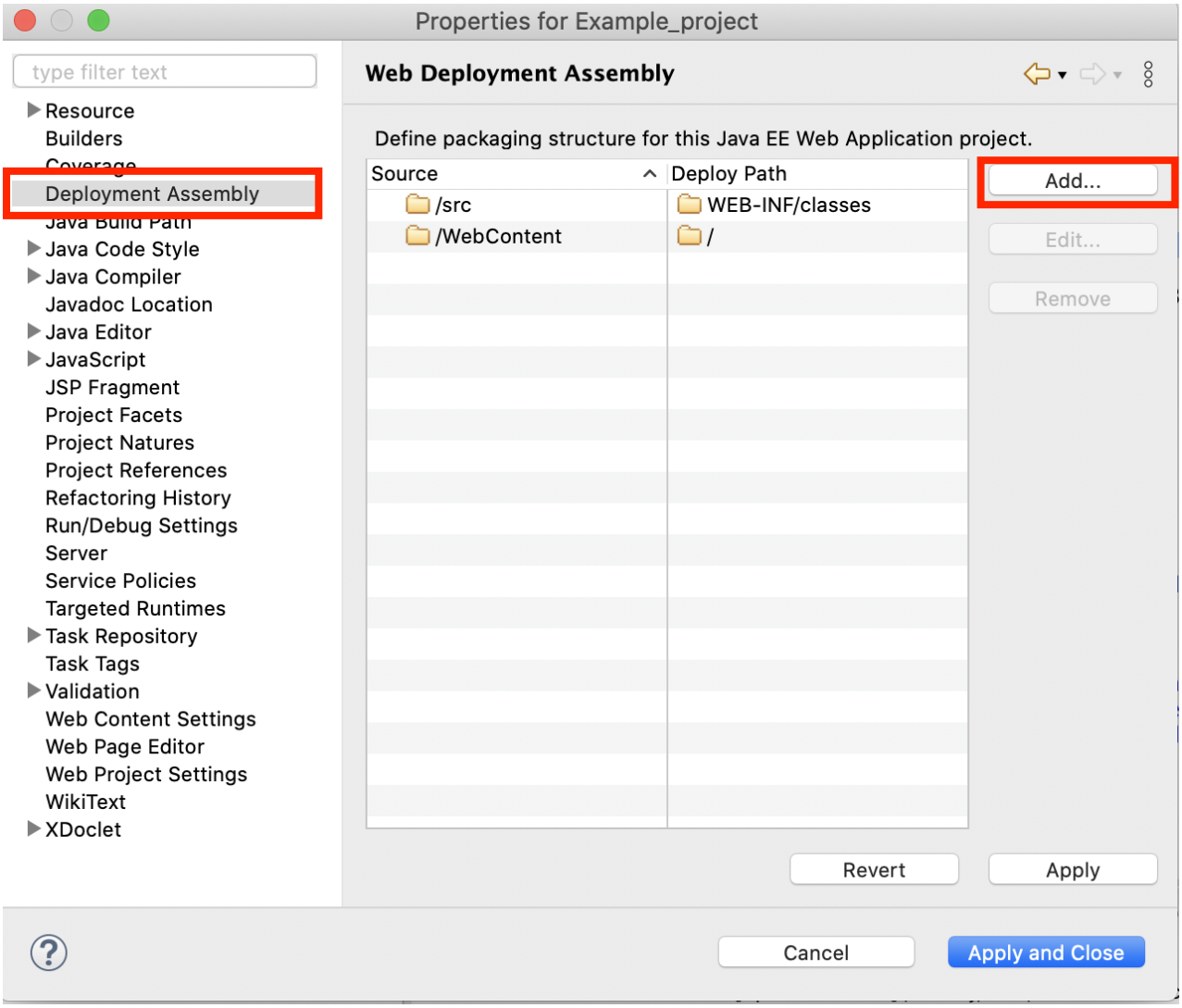


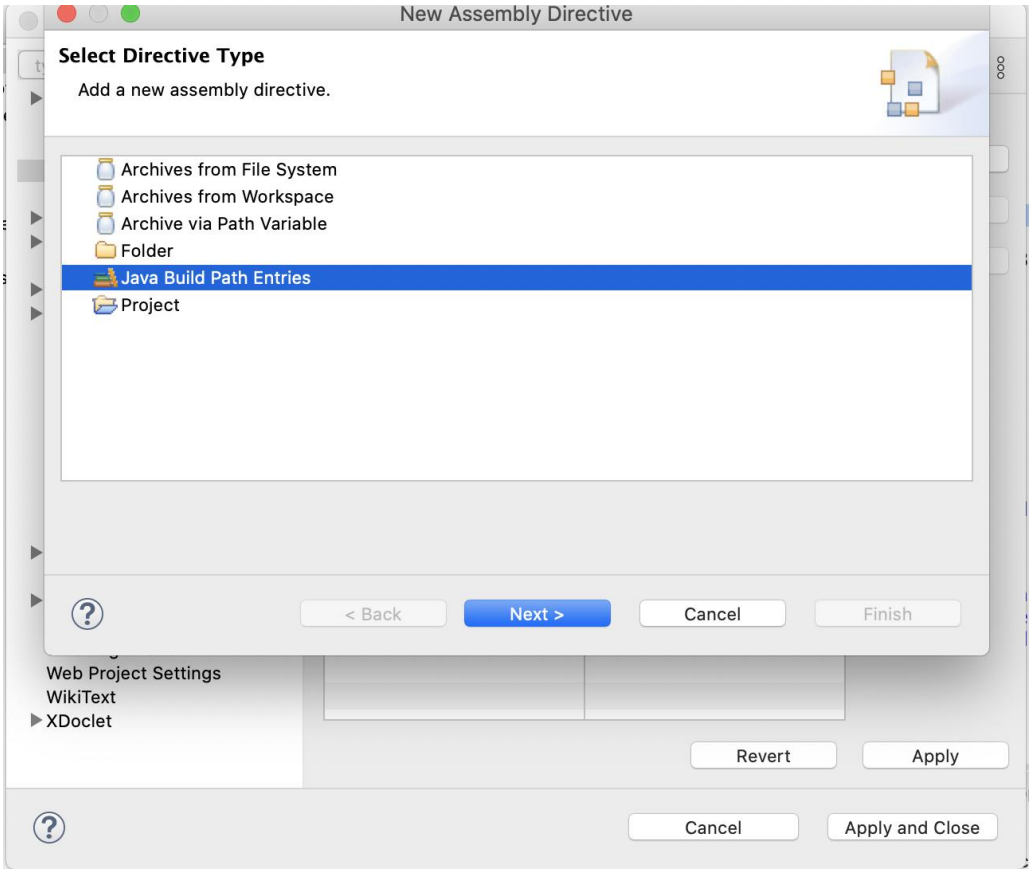
- Enter the database details (i.e., port, database name, username, password) for the connection.
- Test the connection.

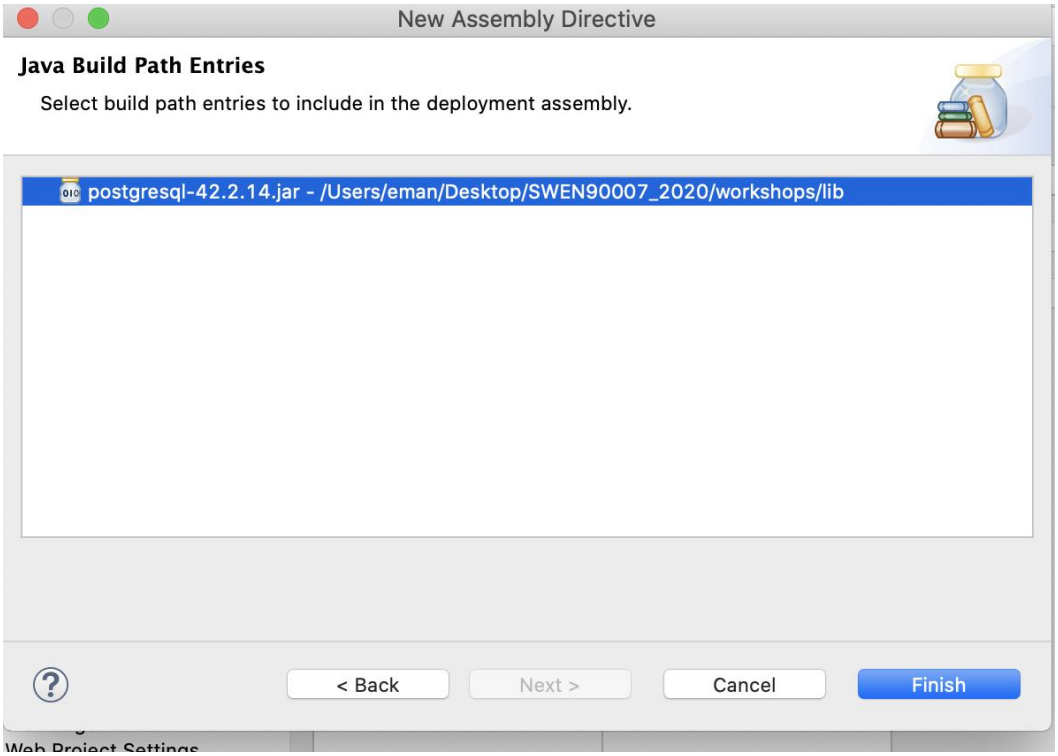

PostgreSQL Connection Properties
Drivers:
PostgreSQL JDBC Driver
Properties
General
Optional
Database:
myDB
URL:
jdbc:postgresql://localhost:5433/myDB
User name:
postgres
Password:
Save password
Test Connection
Cancel
Apply and Close

4. Add library to the build path of the project.

- Add postgresql-(VERSION).jdbc.jar to the build path of our “Example_project” project.
- Go to “properties” of your project. Select “Deployment Assembly” -> add -> ***Java Build Path Entries***. This makes sure the library will be included in the deployment artefact.



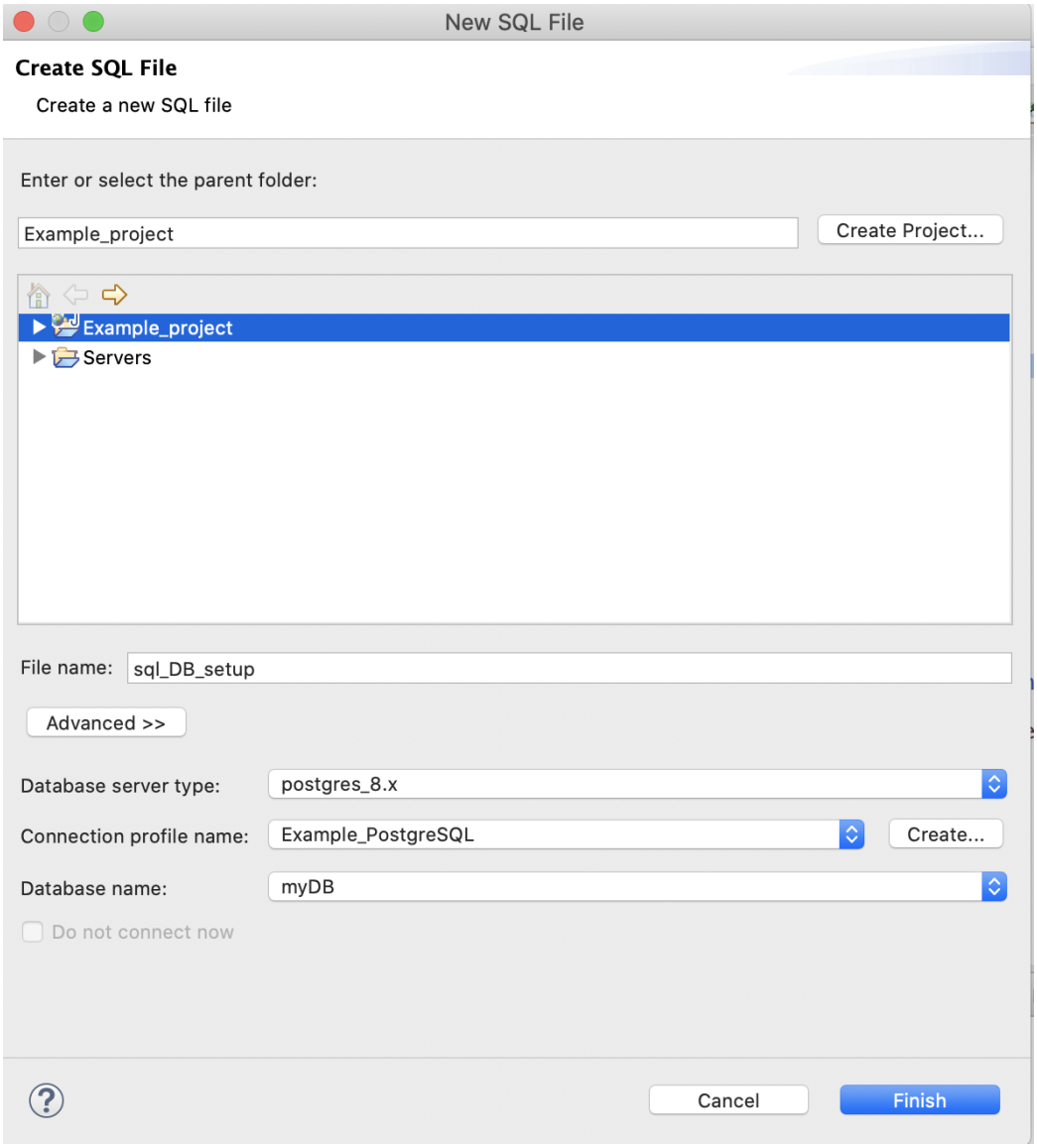

New Assembly Directive
Select Directive Type
Add a new assembly directive.
Archives from File System
Archives from Workspace
Archive via Path Variable
Folder
Java Build Path Entries
Project
< Back
Next >
Cancel
Finish
Web Project Settings
WikiText
XDoclet
Revert
Apply
Cancel
Apply and Close


New Assembly Directive
Java Build Path Entries
Select build path entries to include in the deployment assembly.
postgresql-42.2.14.jar - /Users/eman/Desktop/SWEN90007_2020/workshops/lib
< Back
Next >
Cancel
Finish

□ Setup the Database:

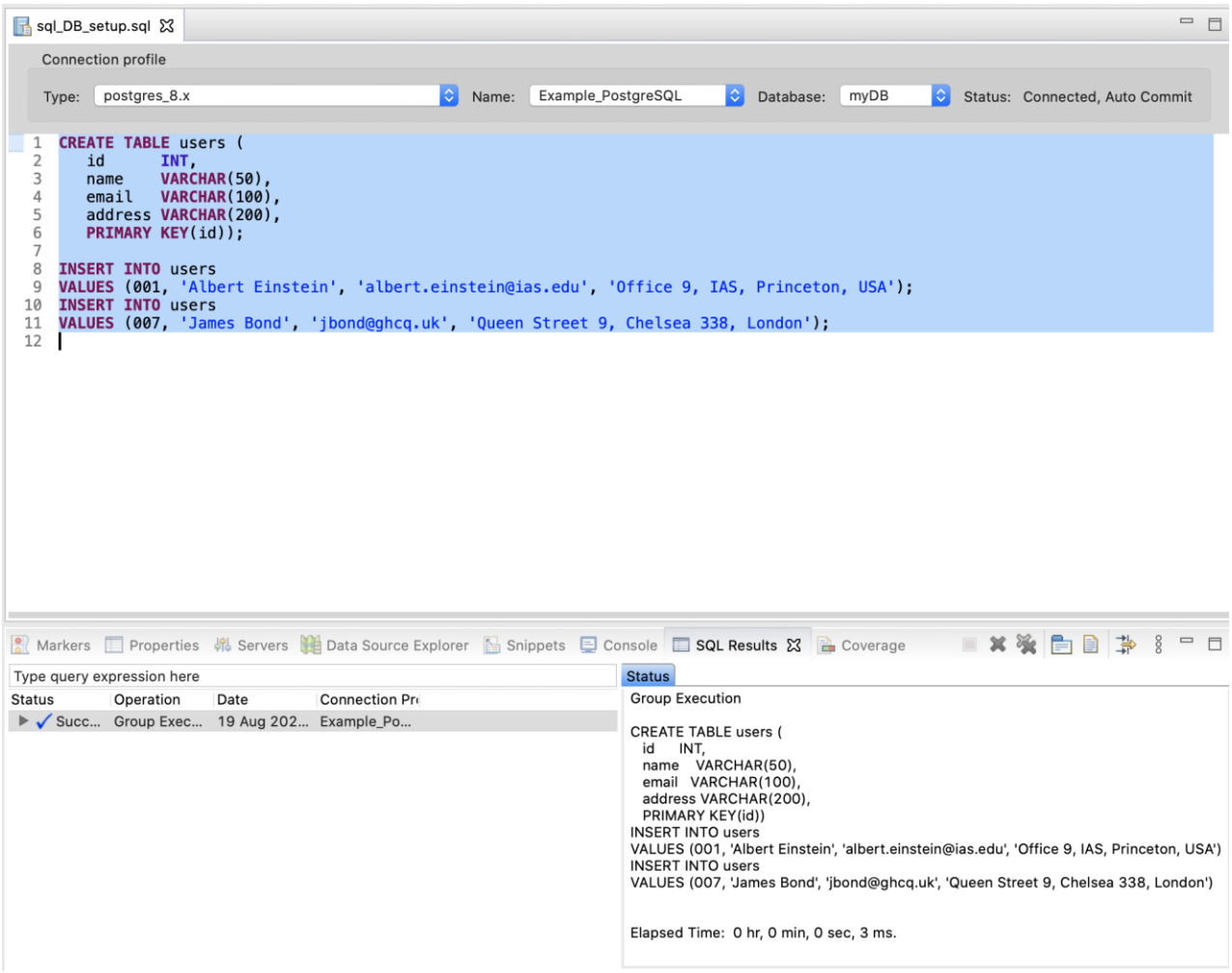
Create tables, and populate tables

- Under our project, create SQL file named: " `sql_DB_setup`" , with the following details
- Type the sql statements below, then save; This is to create "users" table only.



- Right click on the file “Execute SQL files”.

Note, this sql file is not complete. It is just an example of creating table and populate it. You should design the database tables and write your statements.



Now your database is ready to use.

JDBC Connection to the PostgreSQL Database Server

To establish a connection to the PostgreSQL database server, you call the getConnection method of the DriverManager class. This method returns a Connection object.

The following connect() method connects to the PostgreSQL database server and returns a Connection object

```
/**
     * Connect to the PostgreSQL database
     *
     * @return a Connection object
     */
    public Connection connect() {
        Connection conn = null;
        try {
            conn = DriverManager.getConnection(url, user, password);

            if (conn != null) {
                System.out.println("Connected to the PostgreSQL server
successfully.");
            } else {
                System.out.println("Failed to make connection!");
            }

        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }

        return conn;
    }
```

The complete program for connecting to PostgreSQL database server is as follows:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class JDBCPostgreSQLConnection {
    private final String url = "jdbc:postgresql://localhost:5433/myDB";
    private final String user = "postgres";
    private final String password = "123";
    /**
     * Connect to the PostgreSQL database
     * @return a Connection object
     */
    public Connection connect() {
        Connection conn = null;
        try {
            conn = DriverManager.getConnection(url, user, password);

            if (conn != null) {
                System.out.println("Connected to the PostgreSQL server
successfully.");
            } else {
                System.out.println("Failed to make connection!");
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
        return conn;
    }

    public static void main(String[] args) {
        JDBCPostgreSQLConnection app = new JDBCPostgreSQLConnection();
        app.connect();
    }
}
```

So we can connect to the PostgreSQL database server successfully.