SWEN301 Lab - JDBC

Objectives

Learn how to use a highly standardised API to connect to an external data source.

JDBC in one Short Paragraph -- Read this First

JDBC (Java database connectivity) is a framework for Java programs to connect to databases. While it was designed to connect to relational databases (mysql, postgres, oracle etc), it can be used to access a wide range of data formats and storage technologies. At its core, it provides an API to Java programs to send SQL statements (queries, statements to manipulate data) to a database. Integration with external systems is provided via a *driver* that is provided by the database vendor, or by third parties.

Instruction

1. Create a new Java project in the IDE of your choice.
2. In this project, create a folder **db** , and in this folder, create a text file **students.csv**.
3. Copy the following data into this file:

```
ID,FIRSTNAME,LASTNAME,DEGREE,MAJOR
1,Sam,Smith,BE,SE
2,Tom,Taylor,BSc,CS
3,Anne,Ambler,BSc,CS
4,Ben,Berkley,BSc,Math
5,Catherine,Conner,BE,SE
6,Doris,Day,BE,ECSE
```

4. Install the driver:
    a. Download the jar file containing the JDBC driver / DB from https://repo1.maven.org/maven2/net/sourceforge/csvjdbc/csvjdbc/1.0.36/csvjdbc-1.0.36.jar
    b. In your project, copy this in the project folder.
    c. Add this library to the project: (IntelliJ: “Add as library .. ” in the context menu, Eclipse: go to build path settings, CLI > add this to the classpath with the -cp flag)
5. Implement a class **FetchStudents** with the following methods (static methods are ok, you can declare **throws Exception** for all methods to avoid exception handling in your code in order to reduce boilerplate code). These methods should use SQL / the jdbc driver for csv to access the data.
    a. **fetchAll()** -- fetch all student records from **students.csv**, and print details to the console.
    b. **fetchById(int id)** -- fetch all student records for the respective id, and print details to the console. The filtering by id is to be done in the SQL statement, not in the Java code.

c. **fetchByDegreeAndMajor(String degree,String major)** -- fetch all student records for the respective degree and major, and print details to the console. The filtering is to be done in the SQL statement, not in the Java code.

Hints

1. There are plenty of online JDBC tutorials. There is an example on how to use JDBC with the CSV driver on the project web site: https://github.com/simoc/csvjdbc , you can use this as a template.
2. A key issue is to construct the database URL . The token after **"csv:"** is the folder name where the database is located. The file name (students) is the table name that will be used in SQL queries.
3. You can find information on how to write SELECT statements here https://www.w3schools.com/sql/sql_select.asp (and in many other places)
4. Note that when you use conditions to filter results (i.e., SQL WHERE clauses), you need to put values in quotes.
5. In the CSV file, the first row of data defines the column names -- those are the names you will need to reference in the SQL statements.
6. The ResultSet object implements a well-know design pattern, iterator -- but it is a bit different from standard Java iterators.

Limitations

We use JDBC to query CSV files. This is not a "real database", it does not support many features like writing data, transactions, concurrency .. . However, interaction with a "real database" (postgres, mysql, oracle, db2 etc) via JDBC is almost identical to the approach here, the main difference is that you would use a different driver library and a different url to identify the database to connect to.

How is a Driver being Located ?

You may wonder how Java figures out to use the JDBC driver library we added to the classpath to connect to the database. Java uses the plugin-like architecture here. At runtime, it scans the libraries (jar files) in the classpath for "services" they provide -- services are basically interfaces (such as **java.sql.Driver**) which are implemented by some class within the library. You can see this if you check the file **META-INF/services/java.sql.Driver** inside the csvjdbc jar (remember that jars are just zip files). Then frameworks like JDBC can query which drivers are available.

Try to execute the following code snippet that illustrates how this works, and inspect the driver library by unzipping the jar file.

```
java.util.ServiceLoader<java.sql.Driver> drivers = java.util.ServiceLoader.load(java.sql.Driver.class);
for (java.sql.Driver driver:drivers) {
   System.out.println(driver);
}
```