Maven Practical

Objectives

The objective of this practical session is to configure a Maven POM file and run goals to build a project.

Reference Material

This practical session is based on the material covered in the *Intro to Maven* chapter. Additional information can be found in Maven documentation.

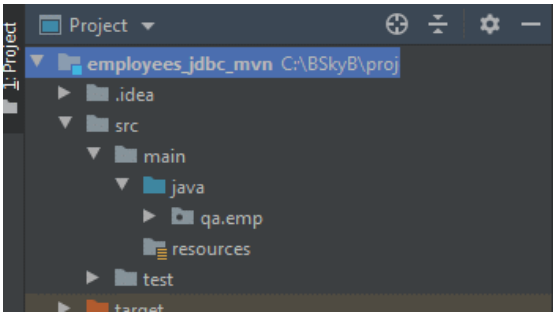
Overview

In this practical you will build the JDBC project using Maven. Recall that with JDBC, the driver class must be specified as a library. This will become a dependency in Maven.

Practical: Configuring the POM to use a MySql Driver

Setting up the Maven Project

1. The JDBC exercise used a GUI (`EmployeeFrame`) to on top of a DAO class (`EmployeeControllerJDBC`), which contained data access code.
2. In IntelliJ, create a new Maven project with **File > New > Project... > Maven**. Click **Next**. Choose a folder and then **Finish**.
3. The resulting project contains the conventional Maven project structure, with folders like `src/main/java`, which is where you should copy the source code from your previous exercise.

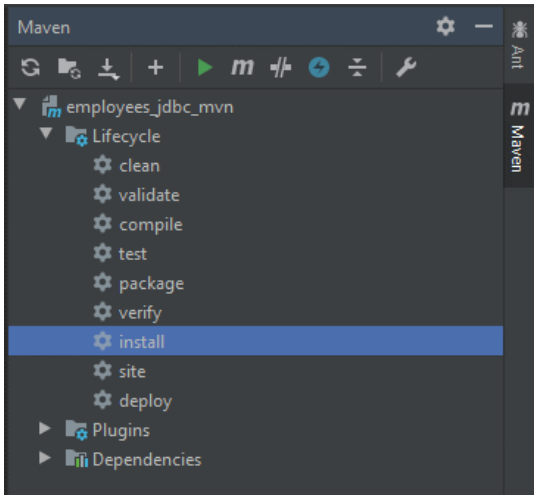


4. Locate the file `pom.xml`. This is the central Maven configuration file, which contains dependencies and plugins used in building the project.
5. The `<dependencies>` tags contain the registration of libraries, in this case for the MySql driver. You can search for the appropriate tags online, for example by Googling 'maven mysql'. This will most likely take you to the `mvnrepository.com` site. You can choose a new version of the database driver JAR mysql-connector. It is usual to select a new version, but maybe safest is to choose one that has a reasonable number of downloads. With open-

source libraries, new releases happen frequently, and bugs sometimes occur. The tags you need will be something like:

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.20</version>
</dependency>
```

6. Save the POM file. After a few seconds, the JAR should be visible under External Libraries. Sometimes, you need to give IntelliJ a nudge to fetch the JAR. If you seen a link 'Enable Auto Import', then this will help. Alternatively, the Maven tab (on the right) has a **Lifecycle > install** option, which builds the project and puts all JAR dependencies in the local repository.



7. The project should run as before.
8. Note that you should be able to build and run your application from a terminal, ignoring IntelliJ. You use the Maven commands to build:

```
> mvn install
```

and then find the JAR file in a `target` folder:

```
> java -cp emp-1.0-SNAPSHOT.jar qa.emp.EmployeeMain
```

9. This should launch the GUI as before.
10. Note that you can also write JUnit tests under the test folder. In order for Maven to find the JARs for JUnit, you can add the dependencies:

```
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.13</version>
<scope>test</scope>
</dependency>
```

11. When you build the project using the Maven tools (or in a terminal), it will run the tests and report on their results.